THE UNIVERSITY OF MELBOURNE

Agents for Logistics: A Provisional Agreement Approach

Don Perugini

February 2006

Submitted in total fulfilment of the requirements

of the degree of Doctor of Philosophy

Department of Computer Science and Software Engineering

The University of Melbourne

Victoria, Australia

Abstract

Logistics planning is vital in the military, involving the provision of supplies and transport to support military operations. Operations research, artificial intelligence and (intelligent) agents have been used to address logistics, and thus transportation scheduling. Most of these approaches are applied to *traditional* (military) logistics. Organisations that perform logistics typically own the assets required in logistics plans, and thus can control their behaviour and have visibility of information that governs their actions. Some logistics domains are relatively constrained as tasks required to achieve logistics goals, and who will perform them, is known *a priori*. Additionally, transportation scheduling usually considers *local* transportation, where a single transportation asset can perform the complete route of a transportation task.

The military are changing the way they do business, primarily due to deregulation, outsourcing and a push towards Network Centric Warfare (or Ubiquitous Command and Control). This results in a complex open market environment in which organisations must cooperate and obtain services from other self-interested organisations in order to form flexible and agile enterprises (or coalitions) to achieve their logistics (or business) goals. Therefore, *modern* military logistics planning is performed in a decentralised, dynamic and open environment. It involves the exchange of services, via contracting, among *many* service providing organisations and *many* service acquiring organisations, i.e. many-to-many interaction. The domain becomes unconstrained, as tasks required to achieve logistics goals, and who will perform them, depends on the indeterminate organisations (and their services) that are available at the time. In addition, the military transportation requirements are *global*. Transportation assets may be able to transport only part of the route and quantity of a transportation task, which must be planned in the complex environment described above.

The aim of this thesis is to investigate the suitability of (BDI) agents for modern logistics planning, and the development of a protocol, the *Provisional Agreement Protocol* (PAP), to facilitate the complex agent interaction required for planning and task allocation. PAP is used to implement an agent system that can address our global transportation scheduling problem.

Multi-Agent Logistics Tool (MALT) is an agent system that we propose to automate aspects of military logistics planning. BDI agents in MALT model organisations' logistics processes, interactions (protocols) and expertise. Agents were found to be suitable for modern logistics planning, for a variety of reasons. Agents provide a suitable conceptual model to easily map organisations' processes, interaction and expertise into agents. Both use similar ideas and concepts, e.g. thought of in terms of goals and beliefs, and are both decentralised. Agents can be easily developed to respond and recover from failure, and respond and react to changes that may occur in the environment. This increases the software system's robustness. BDI agents are suited to highly constrained domains, e.g. organisational processes and protocols are known *a priori*. This was the case within *individual* organisations in our logistics domain, although social processes (distributed plans) to achieve an organisation's goals are unconstrained. PAP, a constrained protocol, was developed to find these unconstrained distributed plans. We present a methodology for modelling organisations' processes, interactions and expertise, and allowing agents to recover from failure and react to changes, using BDI agents (implemented in ATTITUDE agent architecture). Agents are also able to automate, and perform quickly, tasks such as calculations, information gathering and checking of constraints. Agents' modular design and decoupling from other agents (due to decentralisation) makes MALT easy to maintain. Therefore, agents provide a *practical* approach to address the modern logistics planning domain. Components of MALT were successfully applied to DARPA's international CoAX demonstration, performing the logistics planning of a medical evacuation from a ship.

PAP is an extension of Fischer *et al.*'s Extended Contract Net Protocol, overcoming shortfalls with previous contract net approaches when applied to our domain. PAP facilitates planning and task allocation in decentralised, dynamic and open environments, with agents involved in a many-to-many interaction through contracting. Agents using PAP are able to form flexible and agile enterprises (or coalitions) to achieve their business goals. PAP requires less communication (under certain circumstances) and has greater planning flexibility than previous contract net approaches. It is able to perform a decentralised depth-first search (with a dynamic search tree), which from our knowledge, has not been previously done. PAP is relatively consistent with commercial contracting undertaken in business. Since it allows deliberation before forming full contracts, it may reduce the number of broken contracts. It does not fall into livelock (i.e. will converge to either a solution or no solution) or deadlock, and by forcing distributed processing it has beneficial scalability properties.

PAP was applied to the combinatorial auctions and global transportation scheduling domains. PAP was able to facilitate the common static single auction case, as well as the novel single dynamic and multiple (dynamic) auction cases. PAP is shown to have benefits over current (centralised) one-shot approaches, including: bidders not required to submit all their bids and dependencies to a single bidder; reduced communication if bidders possess many bids; and an improved solution through PAP's ability to interact with a changing environment during the auction (planning) process. In the multiple auction case, PAP performed better when resources (bids per auction) were plentiful since competition, and thus the tragedy of the commons phenomena, was reduced. PAP's backtracking facility was shown to be able to find a better solution than the first greedy solution with single static auctions, but was detrimental with multiple auctions, primarily due to environment dynamics. This seems counterintuitive as one would always expect backtracking to provide a better solution.

Our global transportation scheduling implementation overcomes shortfalls with Fischer *et al.*'s transportation implementation when applied in our domain because it allows partial routes, in addition to partial quantity, bids. This allows our implementation to address a greater range of transportation problems than current approaches. Our implementation allows a more informed bid pricing than previous approaches, and uses a bid evaluation function that reduces communication. Experimental results show that our implementation worked well with the scenarios used – producing similar plans quicker (ideally) than doing it manually, which is how it is currently done in practice (in military training). Even if our implementation did not perform much better, planners would still benefit from automating this tedious and complex task, freeing their time for other important tasks. Issues of *commitment*, *communication*, *time* and *partial observability* complicates planning, which are a result of decentralisation (types of domains that PAP was developed for). This causes difficulty in deliberating and making informed decisions.

Declaration

This is to certify that:

i. the thesis comprises only my original work towards the PhD except where indicated in the preface,

ii. due acknowledgement has been made in the text to all other material used,

iii. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signed, ____

Don Perugini

February 2006.

Preface

The work in this thesis has not been published elsewhere, except as noted below.

Papers and the technical report in (Perugini and Fabian 2001; Perugini, Lambert et al. 2002; Perugini, Wark et al. 2003) are utilised in chapters 1 - 3. Papers (Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004) are the utilised in chapters 4, 5 and 7, and paper (Perugini, Lambert et al. 2005) is utilised in chapter 6.

All papers are co-authored by my supervisors, Leon Sterling, Adrian Pearce and Dale Lambert.

Ontology analysis using EXPLODE in section 3.1.4 was performed jointly with Maia Hristozova.

The medevac agent used in DARPA's international CoAX demonstration (discussed in section 3.2.5) was developed with the assistance of Steven Wark and Andrew Zschorn.

Acknowledgements

I would like to thank my three supervisors, Leon Sterling and Adrian Pearce from The University of Melbourne, and Dale Lambert from the Defence, Science and Technology Organisation (DSTO). I appreciate your patience, guidance and support throughout my PhD. I would also like to thank DSTO for their financial support and resources.

Thankyou to all my colleagues at DSTO (Human Systems Integration Group and Distributed Enterprises Group) and The University of Melbourne (Intelligent Agent Laboratory). From DSTO, I would like to thank Steven Wark, Andrew Zschorn and Ingrid Fabian for their technical assistance with ATTITUDE and helpful discussions; Don Gossink, Dominique Estival and Chris Nowak for their supervision and feedback; Chris Drymalik, Mofeed Shahin and Peter Smet for computing resources and technical assistance to run my simulations; and higher management, Michael Webb, Jason Scholz, Rudi Vernik, Alan Burgess, Neil Bryans and Ian Heron for their continuing support.

I would like to thank members of the Intelligent Agent Laboratory for their constructive comments and suggestions during my regular week long visits to The University of Melbourne. Thanks to Susannah Hicks for useful discussions during our coinciding visits to Melbourne.

Thanks also to the administration and technical staff at The University of Melbourne (Department of Computer Science and Software Engineering) and DSTO (Command and Control Division).

I would also like to extend my appreciation to researchers in the area for their helpful feedback and discussions. These include Ed Durfee, Gal A. Kaminka, Jeffrey S. Rosenschein, Michael Wooldridge and Liz Sonenberg during AAMAS04 doctoral mentoring program; Milind Tambe during AAMAS03 doctoral consortium; Kevin Leyton-Brown for assistance with the CATS software; and anonymous reviewers for their useful feedback and suggestions.

A special thanks to my parents, parent in-laws, family and friends for their support.

Last but not least, I would like to thank my wonderful wife Michelle. Her continual support, useful discussions and feedback, and love, gave me the strength and motivation to be the best that I can be. I love you very much.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

Australian military logistics planning[1] is primarily concerned with the supply and transport of resources to support a military operation. For example, in order to deploy a force element (e.g. army unit) for a military operation, logistics planners must determine the supply requirements to sustain them throughout the operation, such as food and fuel. Logisticians must then plan the transportation of the force element and the supplies from their origin to their required destination. Most of the planning is currently performed manually in logistics training exercises[2], which is a complex, tedious and time consuming process. It involves many calculations, satisfying many constraints, and having to deal with many combinations of possible options (plans) to achieve logistics goals. Logistics could make or break a military operation. If, for example, logistics constraints are not satisfied (e.g. a runway is too short for a cargo plane to land on), this could result in the logistics plan, and hence the military operation, to become infeasible. Therefore, military logistics planning is a vital part of military operations. A software support system that could automate aspects of logistics planning could potentially lead to the improvement of the logistics planning process (e.g. time to find a plan) and resulting logistics plans. Additionally, a support system could benefit the logistics planners by automating some of these demanding logistics processes.

One of the most demanding components of military logistics planning is transportation scheduling. We refer to the military transportation scheduling domain as *global transportation scheduling* because a large quantity of resources must be transported large distances – on a *global* scale. A single transport asset may only be able to transport part of the quantity of a resource, only part of the distance (or *route*). Therefore, the domain allows transhipment (or drop and swap), enabling resources to be transferred between

[1] This thesis is concerned with planning at the operational level of command in the Australian Defence Force (ADF).

[2] The author was involved in military logistics training in 2000 in which these observations were discovered – see next chapter.

transportation assets during its journey from its source to its destination. Additionally, the resources to be transported are divisible and the transportation assets may transport a portion of the resources. The transportation domain is multi-modal, as any mode of transport (i.e. air, sea or land transportation assets) may be used to transport the resources. The global transportation scheduling problem is computationally complex, and is not regularly addressed in the literature.

To add to the complexity of military logistics planning, and hence global transportation scheduling domains, the environment is *decentralised*, *dynamic* and *open*. Traditionally, the military primarily used their own assets to achieve their logistics goals. The military typically had full control over, and had visibility of information regarding their asset's behaviour. Solutions to these logistics problems are typically centralised, for example, using Operations Research approaches (Cohen 1985; Ahuja, Magnanti et al. 1993; Bramel and Simchi-Levi 1997; Carter and Price 2000; Kozan and Ohuchi 2002; Kress 2002). All information regarding the assets is gathered and processed by a single decision making entity (that may contain distributed, or multiple, processors). The resulting plan is then imposed onto the assets (or their operators) to be carried out. The military have now changed the way they conduct military operations – or do business. Due to increasing deregulation, outsourcing and coalition operations[3], the military must acquire services from other organisations in order to achieve their logistics (or business) goals. As a result, the military may no longer have control over assets that are used to achieve their logistics goals, nor access to information that governs their behaviour (how they do things), information regarding the organisations' local plans (what they intend to do) and all the services that they can provide (what they can do). Organisations may want to keep this information private because, for example: it is proprietary, commercial in confidence or classified; to protect others privacy, they may not want to release information regarding services that they intend to provide for certain organisations; or they may not want to provide a particular service that is available because it is not in their best interest. Additionally, the concept of Network Centric Warfare (NCW) (DoD-US 2006), or what

[3] Military operations involving military organisations from more than one nation

we refer to as Ubiquitous Command and Control (UC2) (Lambert 1999; Lambert and Scholz 2005), pushes for the military to move from a rigid hierarchical organisational structure to a flexible decentralised structure. Organisations and sensors are connected via networks, and allocate tasks and gather information in a decentralised manner.

Modern logistics must therefore be performed in a decentralised open market. Organisations must cooperate with other self-interested service providing organisations in order to obtain the required services to achieve their logistics goals. Service providing organisations make their own (self-interested) decisions on how their assets and resources are utilised, and hence provide only those services that they wish to provide. Organisational cooperation involves the exchange of services, where many service providing organisations[4] may be cooperating with many service acquiring organisations, resulting in a *many-to-many* setting as shown in Figure 1. Issues such as contracting (legal agreements) for the exchange of services are present. Centralised approaches are not well suited to this decentralised domain.

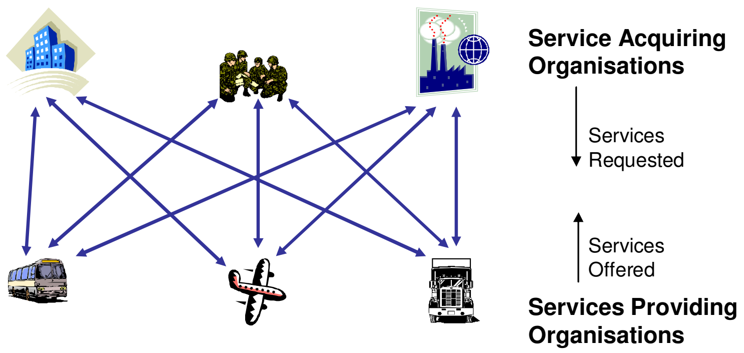

Service Acquiring Organisations
Services Requested
Services Offered
Services Providing Organisations

Figure 1. Many-to-many setting (open market) in our decentralised, dynamic and open logistics domain. Nodes represent organisations and arrows represent their interaction.

The logistics environment is dynamic and open because organisations may enter or leave the system at anytime, and their goals and capabilities (services that they can provide) may change during the planning process. Services may be dynamic because, for example, an organisation may be negotiating to provide the same service(s) to multiple

[4] Service providing organisations in our domain have fixed prices/valuations for their services, and aim to have their services allocated by service acquiring organisations.

organisations concurrently. Therefore, if one organisation aquires the service, it may no longer be available for other organisations, or may conflict with other services that can be provided. Logistics goals may change because, for example, the military operation that the logistics plan is for may change, or a logistics goal may become unachievable. Therefore, the old goal needs to be retracted and a new goal created.

Logistics in the commercial sector can have similar characteristics to our military logistics domain. Rather than supply and transport resources to support a military operation, the commercial sector supplies and transports resources to support their business goals. Commercial transportation is increasingly becoming global due to globalisation. Deregulation and outsourcing in the commercial sector is very common, and thus is also decentralised, dynamic and open. Although this thesis uses military logistics as the application domain, the same concepts and logistics solutions within the thesis, apply to logistics in the commercial sector.

Agents (or intelligent agents) are distributed intelligent software entities that are defined to at least have characteristics such as being (Wooldridge and Jennings 1995; Wooldridge 2002): *autonomous* – they may act on their own and have control over their actions and internal state, and therefore may act in a self-interested manner; *reactive* – they perceive their environment and respond to changes that occur within it; *proactive* – they exhibit goal-directed behaviour, and take the initiative in order to satisfy their goals; and *social* – they may interact with other agents in the environment. Agents have been applied to many application domains that include logistics and e-commerce, such as: supply chain logistics; auctions; coalition formation; *traditional* military logistics planning; transportation scheduling; task allocation; and manufacturing (see next chapter). As with our logistics domain, these application domains are also inherently distributed, complex and/or dynamic in nature, and are usually involved in an e-commerce environment (exchanging services between organisations). It is reasonable to assume, therefore, that agents may be a suitable candidate for addressing our particular logistics and transportation scheduling domains. Agents can represent and model and automate organisational processes, expertise and interactions, and perform the required planning. But how this may be done at the detailed agent level using the BDI agent paradigm (Bratman, Israel et al. 1988; Rao and Georgeff 1991; Rao and Georgeff 1991; Rao and

Georgeff 1995), when to use the BDI paradigm, and the suitability of agents for *modern* military logistics, has not been previously addressed.

In our domain, agents (i.e. organisations) may have a goal to achieve, which they cannot achieve alone. The agent must obtain services from other agents (i.e. *allocate tasks* to other agents) in order to achieve the goal. The set of services obtained, where each service has an associated agent that performs it, is a distributed plan to achieve the goal. The distributed plan required to achieve a goal may not be known *a priori* (i.e. domain is *not constrained*), primarily because it is dependent on the available services that are provided by other agents in the society. These services may not be known due to the decentralised, dynamic and open environment. The dynamic and open nature of the domain results in an indeterminate number and type of agents (and hence indeterminate number and type of services available) to enter or leave the system at anytime. Hence, available services continually change, particularly throughout the planning process. The decentralised environment may result in the agent not having knowledge of all the other agents' available services, since this information may be kept private. The available services may only be revealed when they are asked for, in which case the agents only provide services that they are willing to perform, based on their self-interested goals.

As a related issue, it may not be practicable for all of the agents in the society to send all, or many, of their possible services to a centralised planning agent for processing[5]. In addition to the information regarding all possible services being potentially private, there may also be a large number of services to communicate (as in our transportation scheduling domain, chapter 7). This may result in extensive communication and computation for a single agent to consider the large quantity of services. It may also be difficult to define the complex dependencies between the large numbers of services. For example, using one particular service may result in other services becoming unavailable, or result in their price changing. The dependencies between services may also be private information. Additionally, due to the domain dynamics, services sent for processing may

[5] The term centralised here is slightly different to that above. Distributed agents may act in a decentralised manner, and decide for themselves what services that are willing to provide. These services can be submitted to a centralised agent for processing. In the term centralised above, a centralised agent determines services others will provide.

quickly become outdated, as some services may no longer be available, and new services may become available, during the planning process. Finally, in the case of multiple service acquiring organisations (or agents), they may be reluctant for a centralised mediator to perform the planning for them in order to find the most efficient set of services from service providing organisations to achieve all their individual goals. Organisations may not be confident that the mediator will make decisions in their best interests, and feel they could potentially obtain a better solution on their own. This is why organisations often act independently in the real world.

To enable agents to perform the planning required in our logistics planning and global transportation scheduling domains, agents must be able to perform distributed planning and task allocation in a decentralised, dynamic and open environment, within a many-to-many setting, and involve contracting. Agents must undergo a *decentralised planning process* in order to determine a suitable set of services to be assembled (the distributed plan) in order to achieve its goal. Services are extracted by asking other agents for them, and the services may change during the planning process. Planning is complicated by the issue of *partial observability*, where the agent performing the planning has incomplete knowledge regarding services that could achieve its goal. Protocols that allow agents to perform the planning and task allocation with these domain complexities have not been previously developed.

Aim and Scope of this Study

The aim is to check the suitability of (BDI) agents for the *modern* military logistics environment, and develope an agent protocol to facilitate organisational planning and task allocation, in a decentralised, dynamic and open environment, within a many-to-many (open market) setting.

The *modern* military logistics planning and global transportation scheduling problems will be addressed using agent technology, and through the development of a new agent protocol. The suitability of using agents in these domains will be investigated. We discuss how and when to use the BDI agent approach to model and automate organisational logistics (business) processes, expertise and interactions, at the detailed agent level, in order to obtain the required agent characteristics (e.g. reactivity and pro-activeness). An agent protocol is developed to facilitate required complex agent planning and task allocation, which is relatively consistent with current organisational interaction. The protocol will be applied to the global transportation scheduling problem, as well as combinatorial auction problems. An agent implementation that addresses our complex global transportation scheduling problem is presented.

A Guide to this Thesis

Chapter 2 will provide the relevant background. It includes a discussion on our logistics and transportation domains, a brief introduction to agent technology and the BDI-based ATTITUDE agent architecture, and literature on agent logistics systems and related agent literature related to our problem domain. Chapter 3 discusses the Multi-Agent Logistics Tool (MALT), which is an agent-based software support tool for automating aspects of Australian military logistics planning. Only a small component of the complete vision of MALT has been currently developed (using ATTITUDE), such as agents involved in DARPA's international CoAX demonstration. We investigate the use of the BDI paradigm for implementing agents to model organisations' processes, expertise and interactions. In chapter 4 we look at current protocols for decentralised planning or task allocation, such as Fischer *et al.*'s Extended Contract Net Protocol (ECNP) (Fischer and Kuhn 1993; Fischer, Muller et al. 1996), and their shortfalls for our domain. Chapter 5 introduces the Provisional Agreement Protocol (PAP), which was developed to facilitate our required distributed agent planning and task allocation. PAP generalises the approaches discussed in chapter 4. In chapter 6 we apply PAP to combinatorial auctions, in order to evaluate the protocol. We present benefits of the protocol to ECNP and traditional approaches to combinatorial auctions, and show that it can address the novel dynamic and multiple simultaneous combinatorial auction problems. Chapter 7 discusses the implementation of agents for the global transportation scheduling domain using PAP. Our implementation can address a greater range of transportation problems, and allows a more informed pricing approach, than current approaches. Chapter 8 concludes, in particular discussing the suitability of agents to our modern logistics planning and global transportation scheduling domains.

Chapter 2

2 Background

In this chapter, we will provide a brief overview of our military logistics domain and traditional approaches to logistics and transportation scheduling. We then introduce agents and discuss current agent research related to the development of a logistics support system for our domain – the *modern* military logistics domain.

2.1 Modern Military Logistics

2.1.1 Logistics Definition

The concept of logistics is applied to both the defence and the commercial sector (Blanchard 1992). A general definition is:

"[Logistics is] the process of planning, implementing, and controlling the efficient, effective flow and storage of goods, services, and related information from point of origin to point of consumption for the purpose of conforming to customer requirements." (Bramel and Simchi-Levi 1997)

Basically, logistics is used to get the right thing at the right place at the right time (Bramel and Simchi-Levi 1997). Logistics includes transportation, material flow and handling, product distribution, purchasing and inventory control, warehousing, packaging, manufacturing management and customer service (Blanchard 1992; Bramel and Simchi-Levi 1997; 2000).

The definition of military logistics from U.S. doctrine (DoD-US-Doc 2000) is:

"Logistics is the science of planning and carrying out the movement and maintenance of forces. In its most comprehensive sense, those aspects of military operations which deal with:

- *design and development, acquisition, storage, movement, distribution, maintenance, evacuation, and disposition of material;*

- *movement, evacuation, and hospitalization of personnel;*
- *acquisition or construction, maintenance, operation, and disposition of facilities;*
- *acquisition or furnishing of services."*

The primary focus of military logistics is to ensure that the continuous and timely flow of material, facilities, personnel and ancillary services is maintained throughout an operation. It provides commanders with the correct resources, in the required quantities, on time and suitably configured, necessary to complete their operation.

Kress (Kress 2002) defines military logistics by relating it to a production process, where the inputs to the production process are the means (manpower and machines) and resources (raw materials and services):

"Logistics: A discipline that encompasses the resources that are needed to keep the means of the military process (operation) going in order to achieve its desired outputs (objectives). Logistics includes planning, managing, treating and controlling these resources." (Kress 2002)

Although definitions and terms vary, the logistics concepts are the same. Logistics (military and commercial) is a broad concept involving a wide variety of functions used to support a military or business operation by managing its physical resources – i.e. managing the space and time issues of the operation.

There are three levels to logistics (Bramel and Simchi-Levi 1997; Kress 2002) – although the commercial sector uses the terms tactical and operational where military uses the terms operational and tactical, respectively. To avoid confusion, we will use the military terminology:

- *Strategic level* – long term logistics planning, i.e. logistics decisions that have a long lasting effects. This includes supply chain design and resource acquisition, particularly decisions regarding the number of, location and capacities of warehouses and manufacturing plants, or the flow of material through the logistics network.

- *Operational level* – medium term logistics planning, i.e. logistics decisions that are updated once every month, quarter or year. Includes production and distribution planning, resource allocation, inventory policies and transportation strategies.
- *Tactical level* – short term logistics planning, i.e. refers to the day-to-day decisions. Includes resource scheduling and routing. This occurs daily and in real-time.

Our military logistics domain is focussed primarily at the operational level, but there may be overlap with the tactical level.

2.1.2 Military Logsitics

The author attended a military logistics planning course at the Australian Defence Force Warfare Centre (ADFWC) in order to gain an appreciation of the military logistics planning process, and to investigate the use of agents to support it. Details of our logistics domain discussed in this section, and within the thesis, were obtained at the course. Due to the sensitivity of the information, not all details of our logistics domain can be discussed in this thesis. However, these details are not necessary to explain the concepts within the thesis.

Military planning commences with operational planners forming a plan, called a campaign plan, to achieve a military goal, such as resolving a conflict. It is the role of the logistics planners in this process to realise the campaign plan. Logistics planners need to ensure that the force elements (military units involved in the operation) can be deployed to their required destination, and are supported throughout the campaign by providing them with supplies required to achieve the campaign, such as food, water, fuel, ammunition, equipment, maintenance and medical assistance. In order to achieve this, the logistics planners must determine what resources (e.g. equipment, supplies and services) are required to support the operation, which in turn requires many calculations that are repeated and prone to error if performed manually. A schedule of supplies and transportation is then formed to: acquire the required supplies; and transport the supplies and the force elements to their destination. In forming a logistics plan, logistics planners must satisfy many constraints. For example, force elements must be deployed within the campaign time constraints, supplies must be delivered periodically before they run out, logistics planners must ensure: that transport assets can get to their destination (e.g. can the specific aircraft land on the runway, can the ship dock at the port, are the roads suitable for trucks to travel on, etc.); that weather will not affect the mode of transport selected or the quantity of supplies required (e.g. personnel require more water in hot weather); and that the force elements have enough fuel to support their campaign. Only one unsatisfied and overlooked constraint could result in an infeasible logistics plan, and hence infeasible campaign plan. To assist in checking these constraints and make informed decisions in its plans, logistics planners must gather required information from distributed sources. Information gathered includes the type of supplies required by force elements and their consumption rates (quantity of supplies consumed per day), infrastructure information (e.g. length of a runway, facilities at a sea port, type of roads), weather at the deployment destinations, and supplies and transportation assets available. In chapter 3 we further discuss the logistics planning process.

We refer to the transportation scheduling problem that the logistics planners must undertake as the global transportation scheduling problem (Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004) (see chapter 7). A large amount of resources must be transported on a global scale, and thus a single transportation asset can transport only a part of the quantity of resources only part of the distance (route). Therefore, during transportation from source to destination, a resource may need to be transferred, or drop-and-swap, between two or more transportation assets.

At the logistics planning training course, the logistics planning process was performed manually by the logistics planners, which is a tedious, complex and time consuming process. There is a trade off between the quality of a logistics plan [6] and the time to form a logistics plan. Due to the complexity of the problem and tight time constraints, the quality usually suffers, particularly as it forces the planner to speculate on values, satisfaction of constraints and information, rather than rigorously verify them in order to

[6] Quality is measured by the accuracy, feasibility and optimality (or efficiency) of the logistics plan.

conserve time. To further complicate matters, the large number of constraints and planners forming plans for various military organisations (e.g. Army, Air Force and Navy force elements) which comprise their own logistics “business” processes and expertise, makes it difficult for planners to possess knowledge of all this information. At the training course, this information was not always documented and was obtained from geographically distributed personnel associated with different organisations using, for example, phone calls or email. It was not always easy getting access to the person with the required information.

2.1.3 Modern Logistics Environment

To add to the complexity of the logistics planning process, the modern logistics environment is becoming *decentralised*, *dynamic* and *open*. Traditionally, the military primarily used their own assets and resources, which they have control over and visibility of information regarding their behaviour, to achieve their logistics requirements. The military have now changed the way they conduct military operations – or do business. Due to increasing deregulation, outsourcing and coalition operations (military operations involving military organisations from more than one nation), the military must acquire services from other organisations in order to achieve their logistics (or business) goals. As a result, the military may not have control over assets and resources that are used to achieve their logistics goals, nor access to information that governs their behaviour (how they do things), information regarding the organisations’ local plans (what they intend to do) and all the services that they can provide (what they can do). Organisations may want to keep this information private because, for example: it is proprietary, commercial in confidence or classified. To protect others’ privacy, they may not want to release information regarding services that they intend to provide for certain organisations; or they may not want to provide a particular service that is available as it is not in their best interest. Logistics must therefore be performed in a decentralised, open market, cooperating with other organisations that make their own self-interested decisions on how their assets and resources are utilised, and hence provide only those services that they wish to provide, in order to obtain the required services to achieve the logistics goals.

The environment is dynamic and open because organisations may enter or leave the system at anytime, and their goals and capabilities (services that they may provide) may be continually changing during the planning process. Capabilities may be dynamic because, for example, an organisation X may be cooperating and providing services to the military and other organisations concurrently. Therefore, X's capabilities that are available for the military may change during their cooperation to exchange services as other organisations obtain services from X at the same time. Logistics goals may change because, for example, the campaign plan may change, requiring a new logistics plan to realise it, or a logistics goal may not be able to be achieved, and hence the old logistics goal must be retracted and a new logistics goal created.

2.1.4 Motivation for a Logistics Support System

Logistics planners have a difficult task planning in the complex modern logistics environment, particularly if performed manually as in the logistics planning course. Great benefit can be attained with the development of a military logistics support system that can automate aspects of modern logistics planning. The logistics support system can potentially improve the current logistics planning process and allow the formation of better plans, i.e. high quality plans produced faster than manual plans. Additionally, it can remove logistics planners from menial tasks, allowing them to focus their effort on other tasks, such as refining a logistics plan, rather than trying to form a single logistics plan on time.

2.2 Traditional Approaches to Logistics and Transportation Scheduling

Logistics and transportation scheduling covers a broad area, and thus there is a plethora of literature on the subjects. There are a few issues with much of the current literature in relation to our modern military logistics and transportation problem, which generally fall under the research area of Operations Research (OR) (Cohen 1985; Ahuja, Magnanti et al. 1993; Bramel and Simchi-Levi 1997; Carter and Price 2000; Kozan and Ohuchi 2002; Kress 2002) and Artificial Intelligence (e.g. classical planning and constraint satisfaction)

(Fikes and Nilsson 1971; Allen, Hendler et al. 1990; Hendler, Tate et al. 1990; Currie and Tate 1991; Petrie 1992; Russell and Norvig 1995; McDermott 1996; Weld 1999; Refanidis, Bassiliades et al. 2001; Laborie 2003). Solutions to the problems are typically centralised and assume a static environment. By centralised we mean the problem is solved by a *single decision making entity*, which may use a distributed approach in solving it, with multiple processes where each processor solves a part of the problem, but does not have *control* over its own decisions (it just performs calculations as instructed). Centralised approaches to logistics and transportation problems may not be appropriate for our decentralised, dynamic and open environment. The distributed organisations (and information sources) may be reluctant to release all their information to a centralised system for processing. The amount of information that must be communicated could also be extensive. Having a centralised system decide on the actions that these organisations should perform is a form of "control", which the organisations may not appreciate. Additionally, the decisions made may not be in the interests of the organisations. Due to the dynamic nature of our domain, it may be difficult for organisations to keep a centralised system up-to-date with changes in their information. Any changes may require the centralised system to restart planning. Also, these approaches usually do not consider intra-organisational interaction, such as contracting for services, which are required while planning.

Most of the literature on logistics and transportation scheduling focus on specific problems. These include inventory management, vehicle routing and pick up and delivery, network flow, job scheduling, supply distribution, queuing and forecasting (Cohen 1985; Ahuja, Magnanti et al. 1993; Bramel and Simchi-Levi 1997; Carter and Price 2000; 2002). Although many of these problems are associated with organisations within our logistics system, at the system level most differ to our logistics and transportation scheduling problems.

In the General Pickup and Delivery Problem (GPDP), a set of routes are constructed for a fleet of vehicles in order to satisfy transportation requests (Savelsbergh and Sol 1995). Each vehicle has a given capacity, a start and an end location. The transportation request specifies the load size, the pickup locations and delivery locations. Instances of the GPDP are the Pickup and Delivery Problem (PDP) where the transportation request specifies a single pickup and single delivery location, and all vehicles depart and return to a central depot. The Dial-a-Ride Problem (DARP) is a PDP where all load sizes are equal to one, since the resources to be transported are usually people. The Vehicle Routing Problem is a PDP in which all origins or destinations in the transportation request are located at the central depot.

Pasquier *el al.* investigate dynamic transportation scheduling using a Blackboard-based approach (Pasquier, Quek et al. 2001), which is a dynamic version of the GPDP. The Blackboard (Erman, Hayes-Roth et al. 1980; Nii 1989; Corkill 1991) centralises problem related information and Knowledge Sources access the information to assist in building the solution. Knowledge Sources are independent problem solvers, performing roles such as route planning, request managing and fleet managing.

In these transportation problems, only one vehicle (transportation asset) performs the complete route of the transportation request. Therefore, there is no drop-and-swap (or transhipment) of a resource between transportation assets along the route. Our global transportation scheduling problem requires drop-and-swap as transportation assets may not be able to perform the complete route due to the large distances. Our global transportation scheduling problem is equivalent to the GPDP with a single pickup and single delivery location, or the (multiple vehicle) PDP without the requirement of departing and returning to a central depot, both with time constraints. As with the problem addressed in (Pasquier, Quek et al. 2001), we require planning in a dynamic environment.

The greyhound scheduling problem allows drop-and-swap between transportation assets (Dean and Greenwald 1992; Dean and Greenwald 1992). Transportation assets schedules are fixed (which routes and when they travel are predetermined), similar to a commercial bus, train and airline. Our global transportation scheduling problem requires that transportation assets schedules are flexible.

Haghani and Oh address the multi-commodity, multi-modal network flow problem with time windows, which is equivalent to our global transportation problem (Haghani and Oh 1996). They consider drop-and-swap between transportation assets, where others that address similar problems do not (Haghani and Oh 1996). They develop suitable heuristics to solve the problem, but the solution is centralised and assumes a static environment.

Montana *et al.* use genetic algorithms for military transportation scheduling (Montana, Brinn et al. 1998; Montana, Bidwell et al. 1999). They investigate the grouping of trucks into convoys and the selection of routes for the convoy to travel along. Becker and Smith present a scheduler that allocates military aircraft to transportation tasks (Becker and Smith 2000). Both the aforementioned approaches are centralised and assume that transport assets can perform the complete transportation task or route.

Moynihan *et al.* (Moynihan, Raj et al. 1995) and Hameri and Paatela (Hameri and Paatela 1995) have developed a simulation system for strategic logistics. Our logistics domain is at the operational level of logistics and not the strategic level of logistics (see previous section). Sandoz (Sandoz 1990) also looks at simulation, but for operational level (U.S.) military logistics. We are investigating planning systems rather than simulation systems.

2.3 Agents and the BDI Architecture

In this section, we introduce agents and the Belief-Desire-Intention (BDI) agent architecture. An overview of ATTITUDE, which is a BDI-based multi-agent programming language, will be presented. ATTITUDE was used to develop the agents in this thesis.

2.3.1 Agents

Agents should have at least the following properties (characteristics) (Wooldridge and Jennings 1995; Wooldridge 2002):

- *Autonomy* – operates with little or no assistance from humans and other external sources, and has control over its actions and internal state.
- *Social Ability* – interacts with other agents (software and human).
- *Reactivity* – should be able to perceive the environment that they are in and respond in a timely fashion to situations that occur within it.
- *Pro-Activeness* – agents should exhibit goal directed behaviour, and therefore plan and perform actions to achieve their goals, and responds to failure in a plan/action by devising a new plan to achieve the goal, or revise its current goals.

Therefore agents are intelligent autonomous programs that can interact and react to their environment, and cooperate with other agents, to achieve their tasks. Agents themselves decide whether they will assist or provide information to other agents, if it is in their interest to do so. Agent systems that have more than one agent are generally referred to as Multi-Agent Systems (MAS). Agents may have other characteristics, such as mobility and the ability to learn. Other useful introductory references on agents and their applications include (Shoham 1993; Russell and Norvig 1995; Nwana 1996; Nwana and Ndumu 1996; Nwana and Wooldridge 1996; Jennings, Sycara et al. 1998; Weiss 1999).

2.3.2 BDI agents and Procedural Approaches to Agent Planning

The BDI architecture was originally proposed by Bratman *et al.* (Bratman, Israel et al. 1988) in order to develop practical software agents. It is based on the philosophical model of human practical reasoning (Bratman 1987) where an individual's (intelligent) behaviour can be described by their mental attitudes – their beliefs, desires and intentions. If an individual, or agent, believes that it is hungry and there is food on the table, and it desires (has a goal) to no longer be hungry, then one would expect the agent to act in order to satisfy it desires. It may satisfy its desire, i.e. commit to achieving its desire which may be one of its many goals, by forming an intention to eat the food on the table, and thus performs actions (a plan) in order to achieve the intention.

The BDI agent architecture was formalised by Rao and Georgeff (Rao and Georgeff 1991; Rao and Georgeff 1991; Rao and Georgeff 1995) who implemented one of the first BDI systems, called the Procedural Reasoning System (PRS), and its successor dMARS (Rao and Georgeff 1995; Rao and Georgeff 1995). Agents have an architecture that contains components which are similar to belief, desire and intention mental attitudes. Agents have beliefs about the world and desires, which are goals that they would like to achieve. In order to achieve these desires, agents form intentions which are a commitment to achieving a desire. An important element of the BDI architecture is its procedural approach to agent planning (Georgeff and Lansky 1986). Agents have a plan library that contains "recipes" informing the agent how to achieve its intentions. The plans may contain primitive actions for the agent to perform or further intentions that need to be achieved before the agent can continue with the plan.

To achieve its intentions, the agent must perform means-ends analysis and determine which plans are suitable in achieving the intention. Each plan contains a context that determines the conditions (e.g. beliefs) required for the plan to run. For example, to achieve the intention of eating the food on the table, if the food is soup, then agent will select the plan with a "food is soup" context rather than a "food is steak" context – so it can use a spoon to eat rather than a knife and fork. There may be more than one suitable plan that can achieve the intention. In order to select an appropriate plan, an agent may use meta-plans, which are plans used to determine which plan to execute, or use utilities for plans, so that the plan with the greatest utility will execute.

Having plans defined *a priori* in a plan library assists in developing practical agents, particularly for real-time applications. Agents do not need to determine plans from *first-principles* – i.e. find a plan to achieve its intentions (or goals) from scratch. First-principles planning is performed by searching all/many possible combinations of actions to find the sequence of actions that can achieve the agent's intention. This can take considerable time even for simple problems, particularly if trying to find the optimal plan. For real-time applications, the agent may not have this time to search for a suitable plan. Planning from first-principles is not always necessary. Plans to achieve goals in many real world situations can be determined *a priori*, to a certain level of abstraction. For example, when one attempts to start a car, one does not plan from scratch in order to determine the best way to achieve this goal. The fixed plan of *"open the driver door of the car, get in, take out the car key, insert the key into the key hole, turn the key until the car starts"* to achieve the goal is known by most drivers. Plans may be at a high level of abstraction, and thus the components in the plan may not be primitive actions that the agent can perform, but are intentions that the agents must achieve by selecting sub-plans.

Having plans predefined by agents is a procedural programming approach to the agent planning problem, which is used in programming languages such as C, C++, Fortran, Pascal and Basic (Ralston and Reilly 1993). A procedural program is written specifying, step by step, the list of instructions on how to achieve some goal. The procedural approach allows resource (time) bounded agents to decide on which plans to execute and execute the plans rather than spend time forming the plans from scratch. In the next chapter, we explain which type of application domains the procedural approach is suited to.

Other BDI-based agent languages have been developed since PRS and dMARS. These include ATTITUDE (Lambert 1999; Lambert 1999; Lambert 2003), Jack (Busetta, Howden et al. 2000) and Jam (Huber 1999). ATTITUDE is used to develop the agents described in this thesis.

2.3.3 ATTITUDE

ATTITUDE (Lambert 1999; Lambert 1999; Lambert 2003) is an agent programming language based on the BDI architecture, with consideration from Nelson (Nelson 1982). It incorporates research into multi-agent reasoning, contextual reasoning, and reasoning under uncertainty. ATTITUDE uses the term desires, intentions and routines where Bratman *et al.* (Bratman, Israel et al. 1988) and Rao and Georgeff (Rao and Georgeff 1991; Rao and Georgeff 1995) uses the terms intentions, desires, and plans, respectively. ATTITUDE agents have beliefs about the world [7]. They must satisfy some primary goal or intention by forming desires which are sub-tasks or desirable states that the agent wishes to occur, moving the agent closer to its intention. Intentions and desires are attained using cognitive routines (we will use the term plan from now on) comprising a set of instructions that instruct the agent how to accomplish its desires or intentions. ATTITUDE also allows declarative reasoning by storing inference rules into the agent's knowledge base.

ATTITUDE uses propositional attitude expressions as programming instructions to achieve these desires and intentions. A propositional attitude has the form

[subject] [attitude] that *[propositional expression]*

where:

[subject] denotes an individual whose mental state is being characterised (eg. Fred, Harry, etc),

[7] These are akin to knowledge bases, databases or local memory.

[attitude] is the subject's dispositional attitude toward that claim about the world (eg. believe, desire, fears, expects, anticipates, etc.),

[propositional expression] is describing some propositional claim about the world (eg. it is raining, the sky is blue, today is Monday, etc.).

Examples of formalised propositional attitudes include "FC_IG_Agent2 *believes* (rain Tuesday)", "Supply_Fuel_Agent4 *expects* that (take trucks fuel)", and "Bde_Agent *desires* that (deploy Bde)".

ATTITUDE adapts propositional attitude expressions to form propositional attitude instructions. For example, the propositional attitude instruction "Transport_Agent1 *believe* that (rain Tuesday)", will instruct the agent called Transport_Agent1 to store into its knowledge base the fact (rain Tuesday). The propositional attitude instruction may originate from Transport_Agent1 itself, or by another agent, such as an IG agent providing Transport_Agent1 with weather information – but it will be up to Transport_Agent1 whether it will believe or use this information once it is stored. The propositional attitude instruction "I *desire* (deploy FE)", will result in the agent that is executing the command to itself attempt to satisfy its desire to deploy their FE. The agent will attempt to find a plan in its plan library that can achieve this goal, and will execute it. The propositional attitude instruction "Ship_Transport_Agent3 *anticipate* (breakdown ship) then *also desire* (replan transport cargo)", will result in the agent anticipating the event (breakdown ship) while continuing to perform its current tasks. If the agent does get a break down (e.g. the fact (breakdown ship) enters its knowledge base), then the agent will attempt to find a plan that will satisfy its desire to replan the transport of goods that it was supposed to transport, and execute it concurrently with the tasks that it is already performing. There are many attitudes that can be used to control the actions of the agent, including: *believe, ask if believe, not believe, desire, also desire, match, expect*, *anticipate, sense* and *effect*.

The flow of execution in ATTITUDE is determined by the success or failure of propositional attitude instructions, and ATTITUDE contains several control structures to manage this. For example, the *kleene star operator* ("*") will continuously execute a section of the plan until execution fails. The *join operator* ("^") will concatenate a sequence of two or more propositional attitudes instructions to form a plan, or part thereof. It succeeds when all instructions inside the join operator succeed, i.e. logical AND. The *exclusive union operator* ("|") is used when there are alternatives to achieve a goal (also known as the XOR operator). It uses two or more plans, which the agent will attempt to execute in order and succeed when one of the plans executes successfully. The *guarded exclusive union operator* ("#") can be used to simulate an if-then-else. A "guarded" statement (instruction or plan) is executed, and if it succeeds, its corresponding plan is executed. If the guarded statement fails, then the next guarded statement in the guarded exclusive union operator, and its corresponding plan, is attempted, and so on. If the guarded statement succeeds and its corresponding plan fails, or none of the guarded statements succeed, then an error statement is executed.

Note that that in ATTITUDE, a token preceded by a question mark '?' is a variable. The ATTITUDE instruction "match (?name don)" will result in the variable ?name becoming associated with the string 'don'. ATTITUDE code will be presented in the next chapter.

2.4 Agent Approaches to Logistics and Transportation Scheduling

The use of agent technology for logistics (Timm, Schleiffer et al. 2002) and transportation (Bazzan, Klugl et al. 2004) has received greater attention in recent years. We present an overview of some current systems and literature in these domains.

2.4.1 General Logistics

The Defense Advanced Research Projects Agency's (DARPA) UltraLog (DARPA(b) 2000), which is a follow on project from the Advanced Logistics Project (Brinn and Carrico 2000; DARPA 2000; Adali and Pigaty 2003), uses agents for military logistics to perform dynamic planning, monitoring and replanning. Logistics processes of organisations within the logistics environment are modelled using agents in order to automate the logistics planning process. Agents are developed using an agent architecture called Cougaar (Cougaar 2005), which is not BDI-based, but has some similarities (DARPA 2000). Satapathy *et al.* use agents for military logistics, in a system called

Distributed Intelligent Architecture (or Agents) for Logistics (DIAL) (Satapathy, Kumara et al. 1998). Agents decompose logistics tasks and distribute these tasks to agents that can achieve them. DIAL allows the integration of current military logistics planning systems by interfacing them with agents. Collectively, agents form a coherent logistics plan. CDM Technologies, Inc., have developed a military logistics support system called the Joint Forces Collaborative Tool Kit (JFCT) (CDM Technologies 2005). JFCT contains a rich knowledge base (ontology) of the logistics domain. Agents reason about the knowledge using CLIPS (CLIPS 2005) or Jess (Jess 2005) rule-based engines.

The aforementioned logistics tools consider the traditional logistics environment. These are generally more constrained (agent social structure and roles are predefined), cooperative (agents perform tasks allocated to them without considering their own interests) and perform some logistics functions in a centralised manner (e.g. assume organisations'/assets' information and services are accessible by the agent). We focus on investigating the suitability of agents to the *modern* (military) logistics environment, where the above assumptions may not hold.

Moore *et al.* use agents for logistics replanning (Moore, Kumara et al. 1996). They assume that an initial logistics plan exists, and agents are used to alter components the plan in order to accommodate events that may occur, without having to re-run the complete planning process again. We focus on the initial logistics plan formation.

Tate *et al.* (Tate, Drabble et al. 1994; Tate, Drabble et al. 1995) propose O-Plan2, which incorporates agents with their (centralised) O-Plan classical planning framework (Currie and Tate 1991), and apply it to the logistics domain. There are three agents: a job assignment agent – the user specifies the task to be performed; the planning agent – receives the task and forms a plan to achieve the task; and an execution agent – carries out the detailed tasks specified by the planner. The planning is primarily performed by the planning agent, which is centralised, and thus not well suited to our logistics domain.

Karageorgos *et al.* apply agent technology and the holonic paradigm to address planning and scheduling in virtual manufacturing enterprises, integrating logistics and production planning across companies (Karageorgos, Mehandjiev et al. 2003). It uses a Nested Contract Net Protocol, an extension of the Contract Net Protocol (CNP) (Smith 1980), to allocate manufacturing and transportation tasks to agents. Tasks propagate along chains of agents at various levels of abstraction in the problem domain. In addition to the task allocation problem, our work focuses on the planning problem of which tasks are required to achieve a logistics goal, which is dependent on the services available by agents in the society.

2.4.2 Transportation Scheduling

Davidsson *et al.* present a comprehensive survey on agent-based literature on transportation (and traffic) (Davidsson, Henesey et al. 2004). Fischer *et al.* use agents for decentralised transportation scheduling in their **M**odeling **A**utonomous Coope**R**ating **S**hipping Companies (MARS) system (Fischer and Kuhn 1993; Fischer, Muller et al. 1996). Agents in MARS represent shipping companies and trucks. Transportation tasks are distributed from shipping companies to trucks using the Extended Contract Net Protocol (ECNP), which is an extension of the Contract Net Protocol (CNP) (Smith 1980). ECNP enables shipping companies to achieve a transportation request by finding a transportation plan (or task decomposition) in a decentralised manner and allocating these tasks to trucks (further details in chapter 4). Task allocations using ECNP are usually sub-optimal. Simulated trading procedure, which realises a market mechanism, is used by shipping companies to optimise their plans by buying and selling tasks.

Fischer *et al.* use the MARS system and holonic concepts for transportation scheduling in its TELETRUCK system (Burckert, Fischer et al. 2000; Fischer, Funk et al. 2001). Elements required for truck transportation services, such as trucks, drivers and trailers, are represented as agents (or holons). To achieve a transportation task, the correct combination of these elements must cooperate and merge to form a larger (vehicle) holon – i.e. form an *enterprise* or coalition (see sections 2.5.1 and 2.5.6, respectively). The holon formation is coordinated by a Plan'n'Execute Unit (PnEUs) agent. The PnEUs agent also cooperates with the shipping companies using ECNP to provide transportation services derived by its formed holon.

Sandholm also uses a contract net approach for transportation scheduling and routing (Sandholm 1993). Bouzid apply agents for online transportation scheduling, where trucks accept orders while executing other orders (Bouzid 2003). A Fuzzy Temporal

Characteristic Function based algorithm is used to consider uncertainty on the behaviour of trucks due to, for example, traffic density. Parkes and Ungar apply an auction based method for train scheduling (Parkes and Ungar 2001).

All previous approaches assume that a single vehicle is able to perform the complete route of a transportation task (no drop-and-swap). This may not be the case in our transportation problem. Additionally, CNP and ECNP have shortfalls for our complex planning and task allocation requirements. In chapter 5, we present the Provisional Agreement Protocol, an extension of ECNP, which overcomes shortfalls of ECNP for our planning and task allocation requirements. In chapter 7, we present extensions to Fischer *et al.*'s transportation scheduling framework to accommodate our global transportation scheduling problem by allowing drop-and-swap, and thus multiple vehicles are able to perform the route of a transportation task.

Funk *et al.* looks at intermodal transportation scheduling, which does consider the case of multiple transportation assets performing parts of the route of a transportation task (Funk, Vierke et al. 1999). Transportation tasks are split into three tasks, an initial and final leg that is performed by trucks, and a middle leg that is performed by trains. Montana *et al.* use agents to automate (U.S.) military transportation scheduling (Montana, Herrero et al. 2000). Agents process and decompose received transportation tasks into smaller tasks and delegate these tasks to other suitable agents, which they may do the same. Dong and Li use agents to represent companies and their resources, and define social protocols required between the agents to achieve their transportation goals (Dong and Li 2003). In these systems, the agent social structure and roles are relatively constrained (predefined), and thus so too are the agents' distributed plans and task allocations. Our domain may not be constrained. A suitable distributed plan and task allocation must be found at runtime, based on the (unknown) type of agents available at the time.

Shehory *et al.* apply physics-oriented methods, a model based on classical mechanics, to large scale agent systems in the transportation domain (Shehory, Kraus et al. 1998). They consider cooperative agent system, where agents altruistically attempt to increase the global utility. In our domain, agents are self-interested, and thus act in order to increase their own utility.

2.4.3 Logistics Information Services

Agents have been useful in assisting with the information function of logistics systems. Choi *et al.* use agents to efficiently exchange information among collaborative agents to assist product-support logistics (Choi, Kim et al. 2002). Hofmann *et al.* utilise agents to reduce messages transmitted in a tracking and tracing logistics system (Hofmann, Deschner et al. 1999). Agents provide a flexible decentralised information system that allows messages to be searched for on demand when required, rather than pushed throughout the system unnecessarily.

There are other agent based information services which are not specific to logistics, but could be used in logistics. Sycara *et al.* describe an agent based system to retrieve, filter and fuse information, called RETSINA (Reusable Task Structure-based Intelligent Network Agents) (Sycara, Decker et al. 1996; Sycara and Zeng 1996). They assume that agents have knowledge of the task domain, and which other agents can perform parts of the tasks. This assumption is not valid in our domain. Klusch and Shehory use coalition formation approaches among information agents (Klusch and Shehory 1996), which has been applied to RETSINA (Shehory, Sycara et al. 1997). We discuss coalition formation later in the chapter.

Although we identify information gathering as important in military logistics, we do not focus on this issue in this thesis.

2.4.4 Simulation

Henoch and Ulrich discuss using agents for simulating logistics systems, which seems largely conceptual (Henoch and Ulrich 2000; Henoch and Ulrich 2000). Swaminathan *et al.* describe an agent-based approach of simulating the supply chain (Swaminathan, Smith et al. 1998). Agents to model organisations (e.g. retailers and wholesalers), their control elements (e.g. inventory policy) and interaction protocols (e.g. message types). We focus on a planning system for logistics rather than a simulation system.

2.4.5 Supply Chains

Agent technology is seen as an emerging technology for supply chains (van Hillegersberg, Moonen et al. 2004). A supply chain is a network of organisations, such as suppliers, factories, warehouses and retailers, through which resources are acquired, transformed, produced and delivered to the customer (Chen, Peng et al. 1999). Nissen uses agents for flexible integration of the supply chain (Nissen 2000). Kimbrough *et al.* investigate using agents to play the beer game, which simulates a supply chain (Kimbrough, Wu et al. 2002). Agents must decide how much to order from a supplier to service its customers, with the aim of minimising the long term system-wide inventory cost. Strategies for playing the game, as well as the use of agents to automate such a process, were studied. Chen *et al.* use agents for supply chain management and present a framework of negotiation to support it (Chen, Peng et al. 1999).

Hildum *et al.* (Hildum, Sadeh et al. 1997; Sadeh, Hildum et al. 1998) describe a blackboard-based (Erman, Hayes-Roth et al. 1980; Nii 1989; Corkill 1991) agent for supporting integrated process planning and production scheduling, and applied it to large and dynamic manufacturing facility. This was extended to allow multiple agents for planning and scheduling at various levels of abstraction in the supply chain, in a system called Multi-Agent Supply Chain cOordination Tool (MASCOT) (Sadeh, Hildum et al. 1999). Their mixed-initiative capability allows humans to assist the agents with planning by exploring alternative tradeoffs and imposing or retracting various assumptions.

Fischer *et al.* investigates using agents to apply a holonic approach to the supply chain, facilitating flexible supply chains, and hence labels it a *supply web* (Fischer, Funk et al. 2001). Agents (or holons) represent organisational elements in the supply chain, such as wholesalers and retailers. They interact with each other via agents running appropriate coordination mechanism, such as auctions, in order to acquire and distribute goods.

Walsh and Wellman use market protocol to allocate tasks in a supply chain (Walsh, Wellman et al. 2000; Walsh and Wellman 2003). In (Walsh and Wellman 2003), an auction is created for each good in the supply chain. Agents in the supply chain submit bids to buy and sell the good. Goods are allocated to agents successful in the auction. In (Walsh, Wellman et al. 2000), agents in the supply chain submit all-or-nothing (one-shot) bids for goods that they wish to sell and buy. An auctioneer computes the appropriate allocation of goods in the supply chain. Babaioff and Nisan use double auctions (similar to a stock market) to allocate resources in a supply chain (Babaioff and Nisan 2004).

Although supply chain concepts are important in our military logistics problem, we do not focus on this particular problem in this thesis.

2.4.6 Manufacturing

Agents have been applied to manufacturing, such as Job Shop Scheduling, as a practical approach to the recent concept of Holonic Manufacturing Systems (HMS) (Langer and Bilberg 1997; Bongaerts 1998; Bussmann 1998; Fischer 1999; Fischer, Funk et al. 2001; Uliera, Walker et al. 2001; Conen 2002). HMS consist of a collection of holons, which are autonomous, cooperative, and can be intelligent, and thus are ideally implemented using agents (Fischer 1999; Uliera, Walker et al. 2001). A holon can be made up of other holons, which combine (e.g. form enterprises) to achieve tasks. In a manufacturing system, all its elements may be holons, such as machines and resources. Holons cooperate with each other to perform tasks such as planning, production, scheduling and the physical production, in order to manufacture products. Holons in holonic systems cooperate to achieve common goals, displaying the hierarchical features, and are autonomous and distributed, thus increasing the system reactivity and adaptivity, which displays the heterarchical features.

Peng *at al.* describe an agent system for enterprise integration in manufacturing planning and execution (Peng, Finin et al. 1998). Cicirello and Smith use agents that exhibit behaviour analogous to insects for manufacturing (Cicirello and Smith 2001; Cicirello and Smith 2001; Cicirello and Smith 2004). Wasp-like agents were used for manufacturing scheduling and control (Cicirello and Smith 2001; Cicirello and Smith 2004), and agents that behave similar to a colony of ants are used for shop floor routing (Cicirello and Smith 2001).

Our military logistics problem is at a higher level of abstraction than manufacturing logistics problems. We consider organisational logistics processes, and their interaction and exchange of services with other (self-interested) organisations. Literature in manufacturing generally focuses on logistics processes within a single organisation. The agent approaches are usually cooperative (act to maximise the global utility) and do not consider organisational interaction such as contracting.

The concept of HMS and virtual enterprises (see next section) is of interest to our modern logistics problem, as well as other modern military and commercial problems. There is a push for manufacturing systems to contain a heterarchical (i.e. decentralised, many-to-many interaction) structure rather than a centralised or rigid hierarchical structure (Hatvany 1985; Duffie and Piper 1987; Dilts, Boyd et al. 1991). Decentralisation provides greater flexibility, reduced complexity in developing these systems, improved fault tolerance, systems that are easily reconfigurable and adaptable, allow faster diffusion of information, and have greater modularity (Hatvany 1985; Duffie and Piper 1987; Dilts, Boyd et al. 1991). Our modern logistics environment is inherently decentralised, and therefore is suited to a system with a decentralised structure, with the potential benefits described above. There is also a push for military command and control, and thus their systems, to also move from the traditional rigid hierarchical structure towards a flexible decentralised structure (Lambert 1999; Lambert and Scholz 2005; DoD-US 2006). The commercial sector are finding these types of concepts, i.e. virtual enterprises (see next section), useful. In this thesis, we present a protocol that allows decentralised agents to interact and plan in a decentralised environment, and thus can be applied to these modern planning problems.

2.5 Agent Cooperation, Protocols and E-Commerce

There are various agent subject areas that are related to the use of agents to address our logistics and transportation scheduling domains. Each entity (e.g. organisation), or agent which represents the entity in our distributed domain, is unable to achieve tasks on its own. Agents must cooperate with each other, in their planning and task allocation, in order to achieve their goals. Cooperation among agents can be achieved, for example, using distributed agent planning and coalition formation. In order to facilitate this cooperation, agent protocols are required to specify the social interaction that must be undertaken. Since our domain involves an exchange of services among organisations, areas of automated mediated e-commerce and virtual enterprises also apply. In this section, we discuss some of the agent literature in these areas.

2.5.1 Virtual Enterprises

Virtual enterprises (VE) is composed of a number of cooperating companies that share their resources to support a particular business goal, for as long as it is viable to do so (O'Leary, Kuokka et al. 1997). VE allows dynamic alliances of small, agile organisations that can utilise their resources together, which they cannot if they acted in isolation. In our logistics domain, a situation may present itself where resources need to be transported when one organisation (ADF, coalition or civilian) cannot achieve the goal. Therefore, a collection of organisations needs to form an alliance to collectively achieve the goal. The alliance remains only until the goal is achieved. Potential advantages of VE include: maximising flexibility and adaptability to respond to environmental changes, developing a pool of competencies and resources by combining its member's resources; and adjusting itself according to the market constraints (Martinez, Fouletier et al. 2001). Agent technology is suitable for VE, which requires autonomous entities to interact in flexible ways (Fischer, Muller et al. 1996; O'Leary, Kuokka et al. 1997). Most VE application issues can be interpreted and modelled as agent coordination problems (Ricci, Omicini et al. 2002).

Fischer *et al.* use holonic concepts for virtual enterprises in the supply chain, manufacturing and transportation domains (Fischer, Funk et al. 2001). Elements in the system, such as organisations, machines or truck components, are represented as agents (or holons). To achieve a larger (business) goal, the correct combination of these holon must cooperate and merge to form a larger holon – i.e. form an *enterprise*. The holon formation is coordinated by the *head* agent, which also represents that holon to the rest of the agent society. Holons have the autonomy to join and leave holons (enterprises) as required. Karageorgos *et al.* also applies agent technology and the holonic paradigm to form virtual enterprises in the logistics domain using a Nested Contract Net Protocol, an extension of the Contract Net Protocol (CNP) (Smith 1980). In later sections we present some other agent research areas related to VE, such as coalition formation, agent contracting and negotiation, and agent mediated e-commerce.

There are a selection of systems and frameworks to facilitate VE. The CoABS Grid is middleware that integrates heterogeneous agents (Kettler). It includes the ability to register agents, advertise their capabilities, discover agents based on their capabilities, and send messages between agents. Agentcities is an initiative to create a global, open, heterogeneous network of agent platforms and services to which researchers can connect their agents (Willmott, Dale et al. 2001). Other systems, which form enterprises via e-commerce, will be briefly discussed later.

2.5.2 Agent Communication

In order for distributed agents to cooperate, they need to communicate effectively with each other. We assume that agents have the necessary physical infrastructure to communicate, such as local networks and the Internet. In order to facilitate the cooperation, an agent communication language (ACL) is required (Chaib-Draa and Dignum 2002). Projects aimed at developing suitable ACLs include KQML (Finin, Labrou et al. 1995; UMBC 2002) and more recently, the Foundation for Intelligent Physical Agent's ACL (FIPA-ACL) (Foundation for Intelligent Physical Agents). Most of the research has investigated the generation and interpretation of messages communicated, and conversational policies (agent social *protocols*) (Chaib-Draa and Dignum 2002). Our focus is on protocols (in the second part of the thesis), which specify the messages, or *speech acts*, agents must communicate to others in order achieve its designed purpose, which in our case is planning and task allocation in an open market. A speech act is an action that is performed by communicating something (Searle 1969). Some examples include "I promise to pay you", "Drop your weapon or I'll shoot", or "I now pronounce you husband and wife". The speaker of the speech act expresses a certain attitude, which can be described and defined in terms of beliefs, desires and intentions (Chaib-Draa and Dignum 2002).

2.5.3 Cooperative Distributed Problem Solving

Cooperative Distributed Problem Solving (CDPS), or cooperative multi-agent systems, studies how loosely coupled network of autonomous problem solvers, i.e. agents, can work together to solve problems that are beyond their individual capability (Durfee, Lesser et al. 1989; Durfee 1999; Lesser 1999). CDPS includes agent subject areas such as distributed planning, task allocation, team work and coordination. Durfee and Lesser's Partial Global Planning (PGP) allows distributed agents to cooperate and coordinate their activities by exchanging local plan information (Durfee and Lesser 1987). Agents develop short term plans to achieve their local goals, and then exchange local plan information to identify where other agents' plans and goals interact. Agents' local plans are altered in order to coordinate it with others. Decker and Lesser generalised and extended the PGP in their TAEMS testbed, producing the Generalised Partial Global Planning (GPGP) framework (Decker and Lesser 1995; Decker 1996). Ephrati and Rosenschein enabled multi-agent planning by allowing agents to solve their individual sub-plans and then merging these plans into a global plan (Ephrati and Rosenschein 1994). Ephrati and Rosenschein have also used voting to facilitate multi-agent (joint) planning (Ephrati and Rosenschein 1993). Agents incrementally construct a plan by voting on each joint action at each step of the group plan. Zhang *et al.* use a multi-dimensional, multi-step negotiation mechanism for task allocation among cooperative (not self-interested) agents based on distributed search (Zhang, Lesser et al. 2005).

Team-based agents comprise distributed agents working collaboratively towards the same goal. Such architectures follow the theory of joint intentions by Cohen and Levesque, which is a joint commitment to perform a collective action while in a certain shared metal state (Cohen and Levesque 1991). Jennings *et al.* developed the ARCHON (formally GRATE*) agent architecture, allowing agents to be developed in order to collaboratively solve problems (Jennings 1993; Jennings 1995; Jennings, Mamdani et al. 1996). This was used in application domains such as electricity transportation management and industrial control systems. Tambe developed the Steam framework, which encodes about 300 domain-independent rules that enables agents to exhibit team behaviour (Tambe 1997). Steam was used in military mission simulations and the RoboCup robotic soccer simulation. Tidar *et al.* discuss the formation of agent teams, and modelling teams and team tactics as part of their Smart Whole AiR Mission Model (SWARMM) (Tidhar, Selvestrel et al. 1995; Tidhar, Rao et al. 1996). Grosz and Kraus present a formalism that provides a model of collaborative planning among agents (Grosz and Kraus 1996), which extends the original SharedPlans formulation (Grosz and Sidner 1990). In their model, agents are able to contract out tasks.

Constraint satisfaction involves finding a consistent assignment of values to variables. In distributed constraint satisfaction, variables and constraints are distributed among agents, and message exchange is used to find variable values (Yokoo, Durfee et al. 1998; Faltings and Yokoo 2005). Distributed constraint satisfaction facilitates distributed resources allocation. Each agent has its own tasks, and there are several ways (plans) to perform each task. Shared resources among agents result in constraints between plans. The aim is to find combinations of plans so that all tasks can be executed simultaneously, where each task is a variable and possible plans are variable values.

In the literature above, agents are assumed to achieve a common goal, are able to freely exchange plan information, or do not involve contracting among agents. In most situations in our domain, distributed agents are self-interested, and thus aim to achieve their own goals rather than a common goal, and may be reluctant to release information about their plans. Since our domain involves interaction among organisations, it requires contracting when organisations plan and allocate tasks to each other. Agents with contracts lack obligation to assist others with their tasks compared to when they are acting as a team and jointly performing and committed to tasks (Grosz 1996). To follow, we discuss CDPS research that focuses on self-interested agents, and in particular, distributed planning, task allocation and agents in e-commerce.

2.5.4 The Contract Net Protocol

The Contract Net Protocol (CNP) is a popular and powerful mechanism for decentralised task allocation, first introduced by Smith (Smith 1980). It also allows the concept of contracting between agents, and thus CNP approach is well suited to our planning and task allocation problem. A brief description of the protocol is as follows. An agent whom requires a task achieved, which we call an *auctioneer*, announces its task to potential agents, the *bidders*, that may achieve the task. Bidders submit bids to achieve the task, based on their capabilities, before some deadline set by the auctioneer. After the deadline, the auctioneer evaluates the bids. The most suited bid for the task is granted (or accepted) and the other bids are rejected.

There have been various extensions to CNP in order to meet the requirements of particular domains. Two such extensions include the Extended Contract Net Protocol (ECNP) (Fischer and Kuhn 1993; Fischer, Muller et al. 1996) and another which we will refer to as CNP-ext (Arknine, Pinson et al. 2004). In chapter 4, we describe CNP, ECNP and CNP-ext in greater detail, and show that they have serious limitations for our domain. In chapter 5, we present an extension to ECNP to address the requirements of our domain. Our extension is able to address the problems that CNP, ECNP and CNP-ext are unable to.

CNP has been used in a variety of problems. Collins *et al.* use an approach similar to CNP in their Multi-Agent Negotiation Test-Bed (MAGNET) system for agents to acquire and schedule services in order to achieve a set of tasks (Collins, Bilot et al. 2001; Collins, Ketter et al. 2002). As with our transportation domain, a temporal component complicates the task allocation problem. A mediator is used to facilitate agent interaction in MAGNET. Eymann proposes a completely distributed approach for decentralised economic coordination among agents using CNP without a mediator (Eymann 2001). Tidhar and Rosenschein use a centralised advisor to provide information regarding which agents are suitable contractors for a task (Tidhar and Rosenschein 1992).

Sandholm and Lesser investigated using a levelled commitment contract with CNP (Sandholm and Lesser 1995; Sandholm and Lesser 1996; Sandholm and Lesser 2001; Sandholm and Lesser 2002). They show that agents benefit from allowing contracts to be broken (agents *decommit* from the contract), where the agent that breaks the contract pays a penalty to the other party. The penalty is agreed upon when the contract is formed. We focus on the agent interaction in forming a contract, while planning and allocating tasks, rather than their behaviour after a contract is formed. Once contracts are formed, we assume contracts have a levelled commitment.

Sen and Durfee use a CNP approach for agents to schedule meetings for their human counterparts (Sen and Durfee 1994; Sen and Durfee 1998). In their specification, agents do not have strict contractual commitments as required in our domain, i.e. agents can decommit from a contract without penalty. Fatima and Wooldridge use a protocol similar to CNP to allocate tasks to agents within an organisation (Fatima and Wooldridge 2001).

Agents in the organisation are assumed to be benevolent. If agents within the organisation cannot perform all the tasks, then the organisation purchases agents (or contractors) from other organisations to perform the tasks.

2.5.5 Agent Mediated E-commerce, Mechanism Design & Markets

Agent mediated e-commerce (AMEC) looks at using agents for automating e-commerce (Sandholm 2000; He, Jennings et al. 2003; Sierra 2004). Information and communication technology provides organisations with access to a new and increasing open markets, and can potentially improve their interaction (faster, cheaper, more personalised and agile), but at the expense of an increase in complexity to deal with this new environment. Agents can potentially automate some of the complexities in e-commerce. AMEC involves investigating, for example, mechanisms required to facilitate the buying and selling of goods and services, for example, auctions, negotiation and bargaining.

Collins *et al.* developed the Multi-Agent Negotiation Test-bed (MAGNET) system which supports electronic commerce by mediating interactions between customer and supplier agents (Collins, Bilot et al. 2001; Collins, Ketter et al. 2002). MAGNET consists of a market, market session and agents. The market is a forum for commerce in a particular business area that provides common services. Agents may register if they have an interest in doing business in the market, which assists in matchmaking. A market session is where the agent interaction occurs via a mediator, which, for example, enforces the protocol rules. Two types of agents in MAGNET are customer agents, who require services to achieve their tasks, and supplier agents, who can provide services to achieve customers' tasks. García-Sánchez *at al.* also present a framework for developing e-commerce applications, without requiring a mediator (Garcia-Sanchez, Valencia-Garcia et al. 2005).

Mechanism design is the design of protocols for governing agent interactions, such that these protocols have certain desirable properties (Mas-Colell, Whinston et al. 1995; Varian 1995; Nisan 1999; Parsons and Wooldridge 2002; Cramton, Shoham et al. 2006). Possible properties include *maximising social welfare* and *pareto optimality* (or efficiency) (Sandholm 1999). A protocol maximises social welfare if it guarantees that any outcome maximises the sum of utilities of the negotiating agents. A negotiating outcome is pareto optimal if there is no other outcome that will make at least one agent better off without making at least one other agent worse off.

Auction protocols are investigated under mechanism design. Agents have been used to facilitate auctions, allowing efficient allocation of resources in the presence of self-interested agents (Sandholm 2000). Types of auctions include the English (first-price open-cry), Vickrey, first-price sealed-bid and Dutch (decending) auctions. Each comprises an auctioneer that wants to allocate (e.g. sell or buy) some resource, and set of bidders that want to acquire the resource by bidding for the resource. In the English auction, each bidder openly announces its bid, and is free to raise its bid if it is lower than the current highest bid. When no bidder is willing to raise it anymore, the highest bidder wins the item at the price of its bid. In the first-price sealed-bid auction, each bidder submits one bid without knowing others' bids. The highest bidder wins the item and pays the amount of its bid. In the Dutch auction, the auctioneer continuously lowers its price until one of the bidders takes the item at the current price. In the Vickrey auction, each bidder submits one bid without knowing others' bids. The highest bidder wins, but at the price of the second highest bid. Each auction has its pros and cons, depending on the particular setting (Sandholm 2000).

Agent-mediated auction systems have been developed, which include the AuctionBot (Wurman, Wellman et al. 1998) and the eMediator. Vulkan and Jennings have used agents that apply English auction to allocate services among agents (Vulkan and Jennings 2000). The auction negotiates over price *and* quality of service.

Wellman introduced the idea of market-oriented programming, or general equilibrium market mechanisms (Wellman 1993; Wellman 1996; Cheng and Wellman 1997). Problems are defined as computational markets and market prices are used to allocate resources. Markets have consumers and producers, and a set of commodity goods with a global price. Consumers can buy, sell or consume goods, and producers can transform goods into others. An auction is associated with each good, and agents may submit one bid for each good they are are interested in. Bids are demand functions, specifying the quantity demanded for any possible price of the good, under the assumption that the prices of the remaining goods are fixed at their current prices. General equilibrium occurs when the markets clear, and consumers and producers maximise their preference for goods and profits, respectively, given the prices.

Auctions and market mechanisms are generally used for dynamically priced electronic trades, or problems modelled in such a way. They are used to determine the price of items (*price determination*), and as a result, find a suitable allocation of the auctioned resource. In this thesis, we do not focus on problems of price determination, or strategies by which bidders can obtain the best price in auctions (Priest 2000; Byde, Priest et al. 2002; Shehory 2002; Airiau and Sen 2003; Anthony and Jennings 2003; He, Leung et al. 2003; Cheng, Leung et al. 2005; Greenwald and Boyan 2005; Reeves, Wellman et al. 2005). We assume that agents (bidders) have true and fixed valuations (e.g. prices) for their services (bids) – a take-it-or-leave valuation. This is quite common in e-commerce (Sandholm 2000), particularly with organisations in our domain. Additionally, general equilibrium approaches typically use a centralised mediator (Sandholm 1999). Therefore, agents may not have control over who receives their sensitive information.

As will be discussed later, we apply a (distributed) contract net approach to planning and task allocation, which essentially runs a first-price sealed-bid auction. The auction only serves as a mechanism to extract the most suitable bid, rather than for price determination among bidders' bids. This type of auction process is well understood and used between many organisations (government and commercial), as well as within organisations themselves. Therefore, our protocol remains relatively consistent with current organisational interaction processes, such as contracting and allocation mechanisms. This may increase the likelihood of organisations accepting its use. While maintaining consistency, we still incorporate additions in order to provide organisations with greater expressiveness with their planning and task allocation requirements.

In some auctions, a collection of items need to be allocated. Combinatorial auctions are auctions in which bidders can place bids on combinations of items, called packages, rather than individual items (Nisan 2000; Sandholm 2002; Cramton, Shoham et al. 2006). We discuss this type of auction in more detail in chapter 6 where we use it as an application domain to apply and evaluate our protocol.

2.5.6 Coalition Formation

Coalition formation, based on game theory, is where agents cooperate with each other in order to increase their utility (Rosenschein and Zlotkin 1994; Fischer, Muller et al. 1996; Sandholm and Lesser 1997; Shehory and Kraus 1998; Sandholm, Larson et al. 1999; Shehory and Kraus 1999; Kraus, Shehory et al. 2003). Usually agents have a set of tasks to perform, and exchange their tasks with other agents in order to execute their own tasks at a lower cost. The cost saved by the agents in the coalition is distributed among the agents. Some coalition formation approaches contain three steps. The first step is to generate the coalition groups who will cooperate, and usually these groups are disjoint – a member in one group is not a member in another. The second step is to solve the optimisation problem of determining an appropriate task allocation among the group of agents based on their capabilities. The third step is to divide the value of the coalition among the agents in the coalition.

Our domain primarily concerns itself with the interaction between customers and service providers, rather than between services providers. It becomes a problem of distributing (or allocating) tasks, that the customers cannot perform, to agents at minimal cost, rather than exchanging tasks that it must and can perform in order to reduce the cost of performing its tasks. Dividing the cost saved among agents is not applicable in task distribution. The planning (optimisation) problem, of decomposing a task into suitable subtasks that can be allocated to agents, is solved in some frameworks using a single (centralised) agent (Sandholm and Lesser 1997). Even when the process is distributed, each agent decides on the task allocation of others (Sandholm, Larson et al. 1999).

Shehory et al. (Shehory and Kraus 1996; Shehory, Sycara et al. 1997; Shehory and Kraus 1998) presents an approach where agents form coalitions in order to achieve a set of global tasks, where the tasks are not associated with any service providing agent, and thus could have been provided by a customer. Their approach does not suit our domain because their agents benevolently act to increase a global utility whereas in our domain selfish agents may not be willing to act in order to increase the customer's utility. Agents in (Shehory and Kraus 1996; Shehory, Sycara et al. 1997; Shehory and Kraus 1998) must also communicate their capabilities to other agents.

The primary purpose of the protocol that we devise (see chapter 5) is to address both the planning and task allocation problem, in a decentralised manner. Therefore, our protocol is able to perform the first two steps of the coalition formation process at the same time, without the requirement of having disjoint coalitions. Our protocol addresses the coalition formation scenario where a single agent has a set of tasks to achieve and coordinates the coalition formation process by allocating the tasks to a group (coalition) of agents.

2.6 Summary

The *modern* military logistics domain, which includes the global transportation scheduling domain, was described. In addition to typical logistics complexities, such as calculations and satisfaction of constraints, our domain has a complex social (open market) environment in which organisations must plan, which is decentralised, dynamic and open, and involves a many-to-many setting. Centralised approaches, such as OR and AI, are not well suited to our decentralised domain. Distributed agent technology could provide a viable approach to addressing our complex logistics and transportation scheduling domains. Agents have been applied to various aspects of logistics, but have not been applied to our particular unconstrained logistics or global transportation problems. From our knowledge, an investigation of the use and suitability of the BDI paradigm for agent development in *modern* logistics – the modelling of organisations' logistics business processes, social interaction (protocols) and expertise – has not been undertaken. In the remainder of this thesis, we address these issues.

The social interaction and cooperation required by organisations, and hence agents, in our *modern* logistics domain can be seen as an agent coordination and e-commerce problem. We have presented relevant background literature in this area, such as virtual enterprises, contract net protocol, agent automated e-commerce, and coalition formation. Current approaches have limitations for our particular requirements to plan and allocate tasks in the complex social environment, that involve current organisational interaction such as contracting. In this thesis, we develop a protocol in order to overcome the limitations.

Chapter 3

3 Multi-Agent Logistics Tool

In this chapter, we present the concept of the Multi-Agent Logistics Tool (MALT) (Perugini, Lambert et al. 2002) and the components that have been successfully implemented. Presented is a methodology we used to model an organisation's processes and expertise, and to embed agents' autonomous, proactive (goal-directed), reactive and social characteristics, using the BDI-based agent programming language ATTITUDE. The type of domains in which to use the BDI paradigm is also discussed. These discussions arose from our experience in developing a practical agent-based logistics support system. Components of MALT were involved in DARPA's international CoAX demonstration (Perugini, Wark et al. 2003; Wark, Zschorn et al. 2003).

3.1 Multi-Agent Logistics Tool

MALT aims to automate aspects of logistics planning to assist the logistics planner to obtain required information and logistics advice, and form and analyse logistics plans, quickly and of high quality (large degree of accuracy, feasibility and optimality). MALT uses agent technology to model organisations' and information sources' logistics processes and expertise within the logistics planning environment and perform their logistics functions, such as analysis, calculations, scheduling, resource allocation, provision and collation of information, and the checking and satisfying of constraints. MALT also supports external (coalition and civilian) organisations due to its decentralised and open nature, allowing external organisations to develop their own agents and plug them into MALT.

3.1.1 Overview of MALT

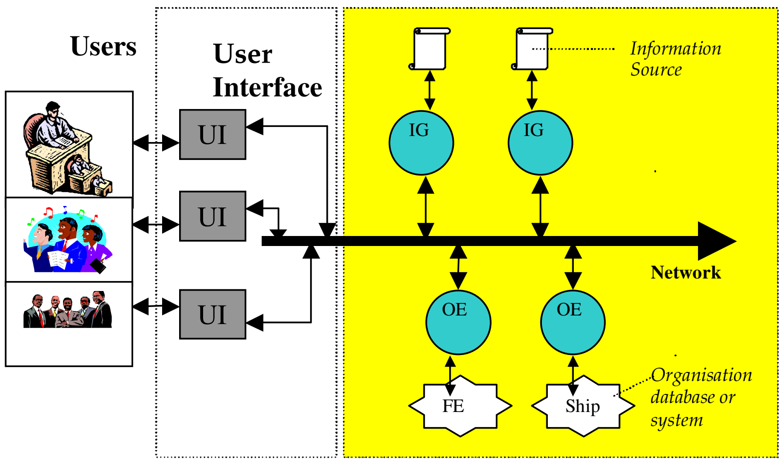


Figure 2. MALT architecture comprising User Interface (UI), Organisational Entity (OE) agents and Information Gathering (IG) agents.

Figure 2 illustrates the general architecture of MALT, containing User Interfaces (UI) (squares) and agents (circles), connected to a network allowing the *distributed* users (human agents) and software agents to communicate. There are two types of agents, Organisational Entity (OE) agents and Information Gathering (IG) agents. OE agents represent and model the logistics "business" processes, expertise and interactions of their associated organisation or force element (FE). They may have resources associated with them, such as supplies and assets (transport assets, people, etc.) which they allocate to achieve their required goals. OE agents access the organisation's database or system to provide the agent with information required to perform the particular logistics functions, such as the status of the organisation's resources (eg. location and availability) and costing information. IG agents will access and analyse information from their associated information source to provide other agents with the information that they require.

Logistics planners interact with MALT via the UI. They may submit queries, such as information required or a force element to be deployed and the required deployment destination and deadline. The UI, which is an agent itself, will cooperate with the various agents, and possibly human agents (human users that are connected via UI and that contain the required information or services), in order to satisfy the query and present the information or plan back to the user. There can be many UIs, which can be geographically distributed or portable, provided they can access the MALT network. This allows greater flexibility in the number of users and the locations from which they can use or contribute to MALT.

To give the user greater confidence with a generated plan or information returned by MALT, information used in order to generate the results could be presented, such as cost, values used, assumptions made, and organisations or assets used in the plan. Agents may be developed to analyse logistics plans and return to the logistics planner information such as possible risks, any shortfalls in resources or supplies, or any unsatisfied or relaxed constraints.

Although MALT aims to automate aspects of logistics planning, some users may want greater control over how results are compiled. Reasons for this include the specific task being over specified (e.g. need to select particular components of a force element for an operation) or to increase the comfort of the user with the result. Agents could potentially be developed to accommodate this flexibility in the level of control, providing autonomy when it suits the user.

MALT is intended to be an open system, meaning that any organisation can develop an agent and "plug" it into the MALT environment. This is important because MALT is to incorporate external organisations that may contain private information and processes, which they may not be keen to release to external sources (or the ADF) in order to develop agents for MALT. Any organisation may develop their own agents and connect it into the MALT environment, at any time – as long as they have suitable social protocols to allow their agents to communicate, compete and cooperate with other agents in MALT. They may program the agent to release only the information or services that they want it to, based on the agent's (the agent developer's) interests. The agent may release services or information only to certain other agents. If an organisation is not comfortable with a software agent releasing information and services, i.e. software doing business on the organisation's behalf, then they may use a UI to access MALT and have a human operator interacting with MALT, providing the information and services. Alternatively, organisations could use a combination of both, having a human operator overseeing the agent's behaviour.

3.1.2 MALT Agents

OE Agents

OE agents may play three roles, and each may play more than one role, depending on whether it is providing or acquiring services, and which services it is providing. The roles are:

- *Supply Agent(s) (**SA**)* – agents that represent OE that *provide* the supply of resources. Examples include a consumer goods supplier that distributes food and water, or military stores that supply equipment.
- *Transport Agent(s) (**TA**)* – agents that represent OE that *provide* the transport of resources, such as military, coalition and civilian (commercial) cargo ships and planes.
- *Manager Agent(s) (**MA**)* – agents that represent OE that *acquire* supply and transport services (from SA and TA) to perform specific logistics functions (or goals) required to support their business goals. MA contain the processes and expertise to manage how resources are utilised in order to perform their particular functions. An example is an agent that represents a military FE that requires deployment to some location (the logistics function) to perform a peace operation (the business goal). The agent, based on the particular FE's logistics processes, can determine the supplies required to support the FE for the particular operation. The FE agent (MA) cooperates with SA and TA to acquire supplies, and transport them to the required destination, respectively, in order to achieve its "deployment" logistics function.

Note that in the rest of this thesis, the terms SA, TA and MA will be used for both singular and plural contexts.

IG Agents

IG agents are connected to various information sources, such as airfields, ports or aviation fuel databases, and weather or climate Internet sites, to gather information requested by other agents. IG agents may analyse the information, either individually or collectively, to provide other agents with information that they have requested. For example, an airfield IG agent can analyse information it has on the runway's length, width and type, to determine which aircraft can land on the runway, sending this information to the agent that requested it. As another example, a weather IG agent may know that it is likely to rain at some location, and a terrain IG agent may know that the roads at the same location are dirt roads. The two pieces of information could be fused (by another agent or the two IG agents) to deduce that the destination will contain muddy roads – which is important for logistics as heavy trucks may not be able to travel on them. Information provided by the IG agents may be tagged with meta-data about the information, such as the time the information was extracted and the source of the information, allowing its reliability to be assessed.

The types of IG agents used in MALT are based on *region then roles*. Each region will have a collection of IG agents, where each agent will have the role of providing a specific type of information about that region. For example, there may be agents associated with each of the states in Australia, hence IG agents for South Australia, Victoria, New South Wales, etc. For each of these regions, the types/roles of IG agents may include (but are not limited to):

- *Weather IG agent* – IG agents to provide weather and climate information about the region.
- *Geography IG agents* – Geography IG agents to each provide information about one or more of the following about the region: terrain; roads; waterways; cities and population; distances.
- *Infrastructure IG agents* – to provide information about the infrastructure in the region. IG agents to each provide information about one or more of the following: ports; airfield; fuel & fuel lines; storage; medical facilities; water; sewerage.

There may be situations where information can cross regions or abstract information regarding a larger region is required. For example, the distance between cities in two states, or the population of Australia, will not be covered by any of the agents described above. In such cases, either a new IG agent can be created for the larger region, or agents for the various regions can cooperate to collectively determine cross regional information.

3.1.3 Agent Goals and Responses

The goals in MALT that are sent between agents (including the UI) in order to *ask* other agents to assist in achieving them, have the form:

<function, what, who, where, when, how>

- ***Function*** – the particular logistics "business" function to be performed. Examples include: information (or inform), deploy, transport, maintain, fix, sustain, stock, supply and build. Deploy includes transport, stock and sustain.
- ***What*** – resources or assets (objects) associated with the required function, or for an information goal, the type of information required. For example, if the function is *transport*, then the *what* component may contain the objects to transport, such as people, military assets and supplies. If the requested function is *build*, then the *what* component could be bridges or roads that are required to be built. If the function is *fix*, then the *what* component could be particular aircraft or vehicles. If the function is *inform*, the *what* component could be weather or population. Examples of objects include: equipment or assets; people; infrastructure (runway, building, bridge, water); FE; and supplies. Examples of types of information are discussed in the previous section.
- ***Who*** – which agent, or group of agents, are to perform the particular function. This does not need to be specified, and in that casse any agent (organisation) that can achieve the specific function can contribute in achieving it. The selected agent(s) must be consistent with the *function* and *what* parameters. Examples include: ADF or coalition force elements (Brigade, SAS); Engineering unit (military or civilian); supplier X and transport agency Y.
- ***Where*** – spatial information about the function to be performed, such as where to perform the function or the region that the information function requires. Examples include: Country; city; town; port; airfield; and base.
- ***When*** – temporal information about the function to be performed, such as when it should occur. For example, with the *transport* function, the *when* component

could be the earliest start time and latest finish time. For an *information* function such as requesting population information, the *when* component could specify the date in the past for what the population was, or a date in the future for what the population is expected to be. Not all goals require the when component, for example, if enquiring about the distance between cities (unless there is a high rate of continental drift). Examples include: time and date; ASAP (as soon as possible); before X days/hours (from start of plan); after event Y.

- ***How*** – used for either:
    - A list of conditions to allow agents or users to control results returned by MALT, giving them greater control and flexibility over MALT. For example, the user can select whether they want to use air or sea transport assets to transport a FE. Examples include: by sea; by air; using ADF assets; using civilian assets; using coalition assets; not using X; using X; minimal cost; highest priority; and lowest priority.
    - Entering information required by agents to perform the function. For example, agents may need to know the type of mission (peacekeeping, intense conflict) to determine the quantity of water required for sustainment, or if asking for stocks of supplies, may need to provide the numbers of days of stocks required.
    - Provide information for responses, such as prices to perform a service (to achieve a goal that was sent) or information in response to a request for information.

An example of a goal to transport 10 casualties from a ship to a medical facility, with earliest start time of 10:00 and deadline of 11:00, using only helicopters that need to winch the casualties, and send this task to all TA that transport resources by air, is:

(Goal

<transport;

(casualties 10);

(agent air_TA);

((from location_of_Ship) (to location_of_medical_facility));

((earliest_start_time 1000) (deadline 1100));

((mode_trans air helicopter), (pickup winch), (casualties serious_condition))>)

The agent that sent this goal should hopefully receive responses from other agents offering services to perfom the task. Note that responses to goals may also have the same structure. A response to this goal may be an ADF helicopter (called helo23) to transport 5 casualties from the ship to the medical facility, picking up at time 1020 and delivering them at time 1040, which is represented by

(Response

<transport;

(casualties 5);

(ADF helo23);

((from location_of_Ship) (to location_of_medical_facility));

((pickup_time 1020) (delivery_time 1040));

((mode_trans air helicopter), (pickup winch), (casualties serious_condition)

(price $10))>)

3.1.4 Ontologies and Service Lookup

In order to understand the terms in goals and responses sent by agents, agents require an ontology, which is a description of terms and their relationships. For example, in the goal above, it requests transport for "air_TA". Agents need to know that this refers to air transport. Additionally, an agent that has access to a Seahawk helicopter must understand that the helicopter can transport items by air, and thus make the link between "air_TA" and its helicopter that can perform the goal. Figure 3 illustrates a simple example of an ontology which may allow agents to determine which assets are air transport assets, which includes the Seahawk helicopter.

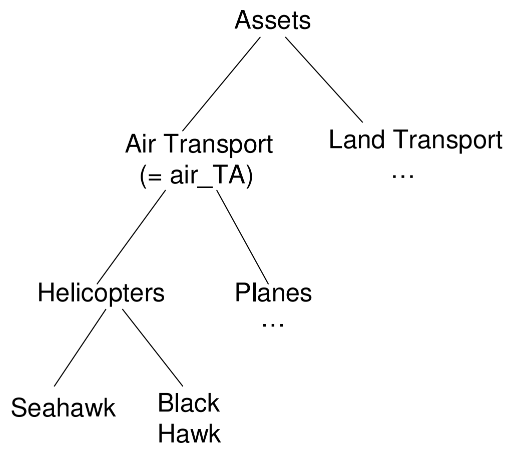


Figure 3. Example ontology – allowing agents to understand that the term "air_TA" in a goal refers to a Seahawk or Black Hawk (and planes that may be included in the ontology).

Ontology development for our logistics domain was *briefly* investigated using the EXPLODE methodology (Hristozova and Sterling 2002). Issues investigated include the representation of a fuel concept (ability of vehicles to travel certain distances) and the requirement for multiple ontologies. Fuel could be represented in litres that is consumed by transport assets or distance (in Km) achievable on a full load of fuel. With fuel consumption, different assets use different fuels and have different efficiencies, potentially making ontology development difficult. For simplification, the latter was chosen (distance) as it was sufficient for our implementation, although the simplification could result in knowledge that is less flexible. Multiple smaller ontologies may be required in our implementation. Helicopters can be described in an ontology regarding their functionality/capabilities, as well as an ontology describing its physical properties and its parts. It may be easier to define two smaller ontologies rather than one larger ontology that incorporates both. We envisage that a separate ontology for each component of the agents' goals is required, i.e. *function* (functional ontology), *what* (object description/relationship ontology), *who* (social ontology), *where* (space ontology), *when* (time ontology), and *how* (domain specific ontology/knowledge). This corresponds closely with the ontology layering discussed in Lambert and Nowak (Lambert 2003; Nowak 2003; Nowak and Lambert 2005).

The EXPLODE methodology assumes an ontological engineer to be involved in the development of the ontology using an agile approach. Our experience in our domain supports the view that an ontological engineer will be useful. Within EXPLODE, implementation and testing plays a mojor role in developing the ontology. Test cases in MALT must cover human to agent communication, e.g. users providing logistics goals, and agent to agent communication, e.g. request for information regarding distance between locations.

Due to the scope of the thesis, we did not investigate the ontology in any great detail. The agents' ontology and knowledge was incrementally created, only as required, during agent development for our particular scenarios. The ontology and knowledge was embedded into the ATTITUDE agents' knowledgebase for the agents to reason with.

It may require extensive communication for an agent to send (or broadcast) a goal request to all agents in a system. Agents may use a service lookup, or matchmaking facility, which for a particular service required, provides a list of agents that can provide that service. Therefore, for a particular goal, the service lookup can be used by an agent to only communicate to those agents that are likely to be able to perform the goal. In order to populate the service lookup, services providing agents that enter the system may register their services with the services lookup server. They can provide as much or as little detail about their services as they desire, depending on what information they are willing to release. Providing greater information regarding services than can be provided is likely to decrease the chance of getting mismatched goal requests, and thus worthless communication. Service providing agents should provide information that is contained in the agent goals to the service lookup server, which is: *function* – the services it can provide; *what* – object associated with the services, or the type of information the agent can provide; *who* – the name of the agent(s) providing the service (e.g. the agent that is registering with the service lookup server); *where* – where it can provide the service; *when* – when it can provide the service (e.g. hours of operation); *how* – e.g. information that is required in a goal for the agent to effectively carry out the service. An agent that must satisfy a goal request may use the *function*, *what*, *where*, *when* and *how* elements in the goal to obtain suitable agents via the service lookup, which will return the *who* component in the goal – the list of agents that are potentially suitable in achieving the goal. The goal can then be sent to these specific agents.

The CoABS Grid and the e-commerce framework by García-Sánchez *et al.* allows agent service lookup (Kettler; Garcia-Sanchez, Valencia-Garcia et al. 2005). The CoABS Grid was used in our implementation of MALT. Collins *et al.*'s Multi-Agent Negotiation Test-bed (MAGNET) system provides a matchmaking facility between customer and service providing agents, via the use of an ontology (Collins, Bilot et al. 2001; Collins, Ketter et al. 2002). The interaction between agents in MAGNET is performed via a mediator. In our framework, we allow direct interaction among agents, as proposed by Eymann (Eymann 2001), giving agents greater control over who receives their sensitive information.

3.1.5 MALT Operation Example

A brief (conceptual) example, using a script, of how MALT may operate follows. Logistics planners must form a logistics plan to deploy a Brigade (Bde) army unit to a fictitious country FC [8]. The logistics planners (users) will need to form a plan to achieve this, using MALT. The script below shows the behaviours (particular logistics functions) and interactions of the various agents in the scenario in order to achieve the logistics goal.

UI_Agent: The users enters a request into the UI (UI_Agent) to deploy the Bde, which in turn generates a logistics goal <*deploy, Bde, Bde_Agent, to_FC, ASAP, via_sea*> and sends the goal to agent Bde_Agent.

Bde_Agent: Bde_Agent plays the role of a MA and runs its logistics process to deploy and sustain its troops. It calculates the quantity of supplies (food, water, ammunition, fuel, maintenance, etc) the deployed Bde requires per day for sustainment. Bde_Agent queries the FC Infrastructure IG agent for current information about supplies (only food, water and fuel), storage (for the supplies) and medical facilities that are available at the

[8] To "deploy a Bde", logistics planners must determine a plan to transport the Bde and required supplies (e.g. water, food and fuel) to the destination (FC). Logistics planners need to consider: how much supplies are required; where to obtain supplies; how to store the supplies at the destination; etc.

destination, using goal <*inform, (supplies & storage & medical facilities), FC_Infras_Agent, FC, current, (supplies_interested_in (food water fuel))* >.

FC_Infras_Agent: FC_Infras_Agent replies to Bde_Agent with the relevant information. One piece of information sent is that country FC contains 100 tonnes of food and 250 tonnes of fuel, using messsage <*inform-response, storage, FC_Infras_Agent, FC, current, ((food 100 tonnes) (fuel 250 tonnes) (water 0 tonnes))*>.

Bde_Agent: Using the information from FC_Infras_Agent, Bde_Agent determines whether it must provide these supplies from home (Australia) or use the supplies at FC. The quantity of supplies required from home can now be calculated (for this example, assume only 100 tonnes of water), and thus sends this goal to the relevant SA (Supply_Agent) to check if it can supply the water, at what cost, and where and when they will be supplying this item. The goal sent is <*supply, (water 100 tonnes), SA_Agent, Australia, ASAP, (quality drinking_water)*>.

Supply_Agent: After checking its inventory, Supply_Agent responds saying that it can supply the 100 tonnes of water from Sydney at 11am tomorrow, at a price of $200, using message <*supply, (water 100 tonnes), Sydney, (1100 tomorrow), ((quality drinking_water), (price $200))*>.

Bde_Agent: Bde_Agent now has the quantity of resources that must be transported from Australia to FC, where the resources are the elements of the Bde (personnel, equipment and vehicles) and supplies to sustain the Bde (100 tonnes of water in this example). Assuming the Bde unit is also in Sydney, Bde_Agent sends the goal <*transport, ((Bde) (water 100 tonnes)), (Sydney to FC), (1200 tomorrow), via_sea*> to the TA (Transport_Agent) asking it to move the Bde and water from Sydney to FC at 12pm the next day, using only sea transport assets.

Transport_Agent: Transport_Agent first checks if the resources to be transported can be transported using its transport assets (assume Transport_Agent has one cargo ship), because for example, large trucks may not be able to be transported by air. It uses an ontology to determine which assets are associated with a Bde (e.g. number of people, and number and type of vehicles), and their weight. Transport_Agent determines that all the Bde elements and water *are* able to be transported by its ship. It then sends a request for information to the FC Weather IG agent (FC_Weather_Agent), Infrastructure IG agent (FC_Infras_Agent) and Geography IG agent (FC_Geog_Agent), to determine if the trip can be made and what conditions are required for the trip.

FC_Weather_Agent: FC_Weather_Agent returns information that the there will be calm seas, little wind and no rain, making it a smooth and fast trip for the ship.

FC_Infras_Agent: FC_Infras_Agent returns information that the port at the deployment destination has facilities to easily dock and unload the ship.

FC_Geog_Agent: FC_Geog_Agent returns information that the distance is 3000Km from Sydney to FC. This is less than the maximum trip length that the ship can make, and therefore the ship is able to make the trip to transport the resources.

Transport_Agent: From information received, Transport_Agent determines that conditions are satisfactory to transport the resources. Transport_Agent replies back to Bde_Agent that it is able to perform the (complete) transportation goal, with message *<transport, ((Bde) (water 100 tonnes)), (Sydney to FC), (1200 tomorrow), (via_sea, (price $350))>.*

Bde_Agent: Bde_Agent forms the logistics plan, which includes schedules of supplies and transport required to deploy and sustain the Bde. It sends the plan to an agent that analyses plans for the Bde (the Bde_Analysis_Agent).

Bde_Analysis_Agent: Bde_Analysis_Agent finds that there is a risk with the plan – storage for fuel may be insufficient if the tempo of the operation increases, as more fuel will need to be deployed and stored.

Bde_Agent: Bde_Agent accepts the risk and continues to present the plan (with information regarding the risk – the analysis) to the UI_Agent.

UI_Agent: UI_Agent presents the plan and analysis to the users. Users can analyse the plan and the information presented, and alter or fine tune the plan to meet their requirements, either off line or using submitting a new goal – with different parameters (e.g. different start or finish times) or use the *how* goal parameter to try different conditions (e.g. specify to use a certain type of ship for transport). It may be possible that various agents for the various services (e.g. supply and transport) involved in the logistics plan have secured the services with the associated organisations (in our example, Supply_Agent and Transport_Agent). If the user is satisfied with the plan, the user can give the okay for the plan via the UI_Agent, in which case the UI_Agent will send a request to all the agents to secure and accept the services offered. This makes it easy for the user to request and secure various services from potentially many organisations.

3.2 MALT Implementation

Although many components of MALT are conceptual, some components of MALT have been successfully implemented. Agents were developed using the BDI based agent programming language ATTITUDE, presented in chapter 2. As will be discussed in the following subsections, a component of MALT has been implemented and demonstrated in an international (DARPA) project called Coalition Agent eXperiment (CoAX). An agent was developed to form a logistics plan to perform a medical evacuation of casualties from an Australian ship to a medical facility. Additionally, an agent was partially written to perform the logistics planning process of deploying a Bde (Bde_Agent), as observed at a military logistics training course, which also involves collating various pieces of information to deduce logistics facts. The methodology used for developing these agents using a BDI approach is presented.

3.2.1 Modelling Logistics (Organisational Business) Processes

Figure 4 describes the high level logistics *process* that the Bde (organisation) uses to achieve its *organisational goal* of *deploying its Bde unit* (obtained from the military training course; due to the sensitivity of the information, further details cannot be provided). Each of the steps in Figure 4 contains sub-processes (the details) that the Bde undergoes to perform that step, which are not shown. Note that the details in the sub-processes that the Bde undergoes to achieve some of the high level steps to deploy its Bde may differ from another ADF force element. Therefore, we model agents for each particular organisation and model *their* particular processes that they use to achieve *their* particular functions and business goals.

***Bde deploy logistics process**:*

(1) Gather information about the deployment destination

(2) Calculate the sustainment requirement for the deployed unit(s)

(3) Try to organise supplies at the destination (so as to not have to transport them from home), but if not successful, organise supplies from home ready to be transported to the destination

(4) Form a plan to transport resources (Bde unit and supplies) to the destination

(5) Analyse the plan

Figure 4. Logistics process to achieve the Bde's organisational goal of deploying its Bde Unit.

```
routine (deploy ?what ?where ?when ?how) is
((^
        Bde_Agent desire (gather_information ?where)
        Bde_Agent desire (calculate sustainment FE)
        ( |   {comment: exclusive union operator – goal "obtain supplies"}
                Bde_Agent desire (check_availability sustainment ?where)
                Bde_Agent desire (order sustainment home)
        )
        Bde_Agent desire (transport resources ?where ?when ?how)
        Bde_Agent desire (report_analysis)
));
```

Figure 5. ATTITUDE plan for Bde_Agent to model the Bde's "deploy" logistics process.

Figure 5 illustrates the *BDI-based* ATTITUDE plan used to model the Bde's *deploy* logistics process – in the Bde's agent called Bde_Agent. Plans in ATTITUDE are labelled routines. The first line states that it is a plan to achieve goal *deploy*, and the plan has input parameters (variables) ?what, ?where, ?when and ?how. The plan contains sub-goals, which are steps in the deploy logistics process, that must be performed in order to achieve the deploy function. The sub-goals are given by the instructions "I desire <sub-goal>", which will result in the agent looking for a plan in its own plan library (the Bde's logistics sub-processes) to achieve this sub-goal. The first sub-goal will run the Bde's plan to acquire an agent to gather information about the deployment destination, that is stored in the variable ?where. This will result in the agent cooperating with, and extracting information from, the relevant IG agents. The next sub-goal will result in the agent calculating the amount of sustainment required by the Bde, based on what Bde components are to be deployed. We then encounter the exclusive union operator ("|"), discussed in section 2.3.3, being applied to content that aim to achieve the goal of obtaining supplies. The agent will attempt the first sub-goal (check_availability sustainment ?where), which will check if the supplies for sustainment are available at the deployment destination. If it succeeds, then supplies are available at the deployment destination, and thus do not need to be transported from home. If it fails, then supplies are not available at the deployment destination, hence, the next sub-goal (order sustainment home) will execute, causing the agent to cooperate with SA to find suitable suppliers to supply the sustainment items from home. Note that in reality, some supplies may be obtained at the destination and others will need to be transported from home. To simplify this example, we assume the supplies are either obtained from the destination or from home, but not both. The next sub-goal will result in the agent cooperating with TA to form a plan to transport the resources (Bde force elements and supplies) to its destination. The last sub-goal will result in the agent performing the required analysis on the plan and other information it has collected or deduced, to provide the user with essential information about the plan. Note that the Bde_Agent plays the role of a MA in achieving its deploy function.

It can be seen that the modelling of an organisation's (logistics/business) processes into an agent is relatively easy and straightforward. The first step is to document the process, and sub-processes, that the organisation performs to achieve its particular function(s) – as in Figure 4. Once this is done, the mapping to the agent (ATTITUDE) code is straight forward. The process and sub-processes are analagous to agent plans (or routines). Each step in the process is either a specific action to be performed by the agent or a sub-goal that must be achieved, in which case the sub-goal's associated sub-process(es) (i.e. associated agent plan(s)) is used to achieve it. Organisational processes that have a complex flow of execution (e.g. loops to perform a set of steps in a process over and over until some condition is met, or perform two steps simultaneously until one succeeds) can also be modelled by the (ATTITUDE) agents using the operators discussed in section 2.3.3.

Lambert discusses a methodology for capturing processes for agent implementation (Lambert 2003). Speech to text is used in order to capture dialogue as an expert(s) is working through the problem domain and describing the process. The expert's utterances are recordered and converted to text. The text is then analysed exposing: the steps in the process, and thus the goals in the agent plan; and the subsequent steps required when a particular step either succeeds or fails to be achieved, describing the flow of execution of the goals. In the following section, we discuss extensions to Lambert's methodology, which evolved from our experience in developing agents in MALT. The captured process is further analysed in order to enable agents to respond appropriately from both failures and changes in the environment, increasing the robustness of the system.

3.2.2 Concepts of Agency within MALT

With the code shown in Figure 5, there is an exclusive union operator that contains code to achieve the goal of obtaining supplies. There are two sub-goals that can be used to achieve this, "I desire (check_availability sustainment ?where)" (or sub-goal 1) and "I desire (order sustainment home)" (or sub-goal 2). Bde_Agent will first attempt sub-goal 1 in order to achieve its goal of obtaining supplies. Sub-goal 1 may fail, and hence Bde_Agent is unsuccessful in achieving its goal of obtaining supplies using that sub-goal (or the plan associated with the sub-goal). If this situation occurs, Bde_Agent is able to recover from the failure by trying a new goal (and hence plan) in order to achieve the goal of obtaining supplies, which is sub-goal 2.

The ability for agents to respond to failure by trying alternative methods of achieving a goal, and hence showing goal-directed behaviour (pro-activeness) by being persistent in achieving a goal, together with their reactive ability to respond to a changing environment by altering their goals and behaviour, is primarily where agents attain their *intelligent* behaviour. Agents' pro-active characteristic (or agent specification) assists in providing agents with *greater robustness* than other software approaches (e.g. object oriented programming) that do not have this characteristic explicitly specified. If agents fail in achieving a goal (or function), then rather than crash, those agents should know the goal they were trying to achieve and can keep trying alternative methods of achieving it. Of course, this behaviour should be programmed into the agent. As mentioned, ATTITUDE agents are structured so that every instruction either succeeds or fails. If instructions fail, the agent does not crash, but instead the flow of execution proceeds. How the flow of execution proceeds is based on the particular ATTITUDE operator comprising the failed instruction, allowing the developer to respond to the failure as required. Therefore, code can be easily written within (ATTITUDE) agents to respond to failure.

How agents should be developed to respond to failure in a particular situation depends on the organisational processes that they are modelling and their responses to failure. Therefore, the agent response to failure should be captured when modelling the organisational processes, as in Figure 5. For each step (or goal) in the process, the developer should consider more than one sub-process (or plan) that may be used to achieve the step (if possible) and in what order, or under what context/conditions, they can be used to achieve the step. This provides multiple alternatives to achieve the same goal, and hence if one fails, the agent may attempt another. The developer should then consider how to respond in the case that all the plans may fail, and therefore the particular step in the process cannot be achieved. Additional steps may be required to compensate for the step that could not be achieved. For example, with the obtaining supplies example mentioned above, if subgoal 1 fails to obtain supplies then subgoal 2 is attempted. The developer should do the same for all steps in the sub-processes, and their sub-sub-processes, and so on. The resulting organisational processes, which now describe their responses to failure, can be easily embedded into the agents, as described above, and illustrated in Figure 5.

An agent's reactive behaviour should also be captured when modelling organisational processes. A developer should capture the organisational responses that take place when changes occur to the environment or a particular situation occurs – such as late information being received during planning that rebuts an assumption that was made, or information that was used, early on in the planning, resulting in current elements of the plan being incorrect. There are two situations that a developer should consider: (i) responses during or after a process (to achieve some organisational goal or step); and (ii) independent responses to situations.

(i) With *responses during or after a process*, a developer should consider what environmental factors (or information) influence each step in each process, and hence influences the organisational goals/steps that the process achieves, and what responses are required if environmental changes occur, during the process and after it completes. Responses, with some examples (using the process in Figure 4), may include:

- *Repeat* – some or all of a step, or sequence of steps, can be repeated to take into account the changes in the environment. For example, if at step (4) of Figure 4, when forming a transportation schedule, the number of troops in the Bde unit to deploy changes from 900 to 1000, the resulting sustainment calculations for the quantity of supplies required to sustain the troops will be incorrect. Therefore, the process may need to go back to step (2) and perform the sustainment calculations again, and then perform step (3) again to obtain the new quantity of supplies. Some components of a step, or a complete step, may not need to be repeated. In the previous example, a change in the number of troops may only affect the sustainment calculations for food and water, and not other supplies such as fuel. Additionally, if the food and water may already be provided at the deployment destination, then step (3) does not need to be repeated. Therefore, only the relevant food and water calculations in step (2) need to be repeated. Considering elements in a process that do not need to be repeated in a given situation can save time and effort.
- *Use a different process* – the context for which a process was selected to achieve a step or organistional goal may change, requiring a new process to achieve it. For example, sustainment calculations in step (2) (Figure 4) are performed assuming a peacekeeping operation. If information is received that the operation is a training exercise, then a new process is executed to perform the sustainment calculations, consistent with the new context.
- *New steps to replace current ones* – new steps in the same process may be required to replace a step(s) that is currently executing, as they are no longer

required to achieve the process. For example, sustainment calculations in step (2) (Figure 4) are performed to determine the quantity of fuel required by vehicles to be deployed. If the vehicles are later found to be electric (in this example), then the goal of determining fuel is no longer valid. A new goal of determining the quantity of batteries, rather than fuel, to run the vehicles is required.

- *New steps to recover and continue* – new steps may be performed to recover from effects of environmental changes, allowing continuation of the current step/process, rather than stop the current process/step in order to repeat or perform new steps/processes, as in the three points above. For example, sustainment calculations in step (2) (Figure 4) are performed to determine the quantity of unleaded fuel for vehicles. Suppose it is later discovered that they require diesel fuel, and not unleaded. Rather than perform the calculations again, a new step can be attempted to change the current quantity of unleaded fuel already calculated to the appropriate quantity of diesel fuel required (if such a calculation exists).

(ii) *Independent responses to situations* is when some event occurs that requires a goal or action to be performed in order to respond to the event, which has no influence on the achievement of other processes that may be executing at the same time. For example, the Bde organisation, while performing the process to deploy a Bde unit in Figure 4, receives a request for information regarding the number of vehicles it possesses. The Bde may respond by finding and providing this information, while still performing the process to deploy a Bde unit. Therefore the new goal created by the event, or the event itself, has no influence on the achievement of the process to deploy a Bde unit. Developers should consider various situations and events that the agent may come across, and the response (goals and processes) required. These events may include the trigger for the organisation's high level goals, such as requests to form logistics plans or provide information, or for simple reactive behaviours, such as stopping a vehicle if it is about to collide with an obstacle.

ATTITUDE contains three instructions, in addition to ATTITUDE operators to control the flow of plans, which assists in making these organisational processes and their responses relatively easy to program into agents. They are: (1) *anticipations*, e.g. "*I anticipate <information request> then also desire <reply information>*"; (2) *expectations*, e.g. "*I* *expect <obstacle approaching> before <delay 15>*", and; (3) *also desire* removals, e.g. *"I not also desire <information request>"* (instructions slightly modified for this thesis). With anticipations, the agent anticipates some belief appearing in its knowledge base (the event), e.g. (information request), and when it does the agent attempts to achieve some goal, e.g. (reply information), concurrently with other tasks it is performing. Expectations will cause an agent to stop and wait during a plan until an expected belief appears in its knowledge base, e.g. (obstacle approaching). Once the event occurs, the agent will continue with the plan. The expected event must occur before some deadline, e.g. before 15 seconds, otherwise the instruction fails. The also desire removal will cause the agent to stop the execution of a plan to achieve goal, e.g. (reply information), that it is currently executing. These instructions can be used to monitor for events (changes in the environment) and execute the required responses if these events occur, or to stop current plans that are being executed when their goals are no longer required. Providing agents with this reactive behaviour further increases their robustness as they are able to respond, and not fail, to a wide range of events and changes that may occur.

Various means of embedding an agent's pro-active and reactive behaviours, or characteristics, have been presented. Agents have two more characteristics, autonomy and social ability. Agents are inherently autonomous. Agents are able to automate organisational processes, and thus perform these tasks on their own. Agent can be developed to make their own decisions based on their own (organisation's) goals, and they do not have to release information to, or perform tasks for, other agents.

An agent's required social capability, or protocol, depends on the particular situation. Some agents require a simple protocol, such as sending tasks to other agents, and others require more complex protocols, such as negotiation and task allocation. In the next chapter, a complex social protocol developed to perform planning (and task allocation) in a decentralised, dynamic and open environment is discussed. In order to develop a social protocol for a particular situation, developers need to capture the social processes that organisations carry out among each other to achieve their social goals. This is done in the same way as modelling organisational processes, except it captures the sequence of messages (communication) and events between organisations. Some factors that should be considered include the type of messages communicated (e.g. request, inform, etc.), who to communicate to, information required in the messages, and the message structure. Once these processes have been defined, they can be embedded within the agents, as discussed above, except that the goals (desires) in the plan (Figure 5) are likely to be goals to send or receive messages. All agents that are expected to interact using the social protocol must have knowledge of the protocol.

3.2.3 When to Use BDI's Procedural Approach to Agent Planning

As mentioned in the chapter 2, the BDI-architecture uses a procedural approach to agent planning, where agent plans are defined *a priori* during development rather than via first-principles during execution. From our experience, the procedural approach works well for the organisational logistics business processes we are modelling in our logistics domain as their individual processes and social processes (protocols) are highly constrained. This may not be the case for other domains that agents may be applied to. On basis of experience, Figure 6 presents a graph that indicates which agent planning approaches are suitable for domains with a certain characteristics. The characteristics we use are the level of constraint in the domain and the time available for the agent to plan. From Figure 6, if the domain is highly constrained, then a procedural (i.e. BDI) approach is suitable since plans for these (relatively) fixed processes can be defined *a priori*.

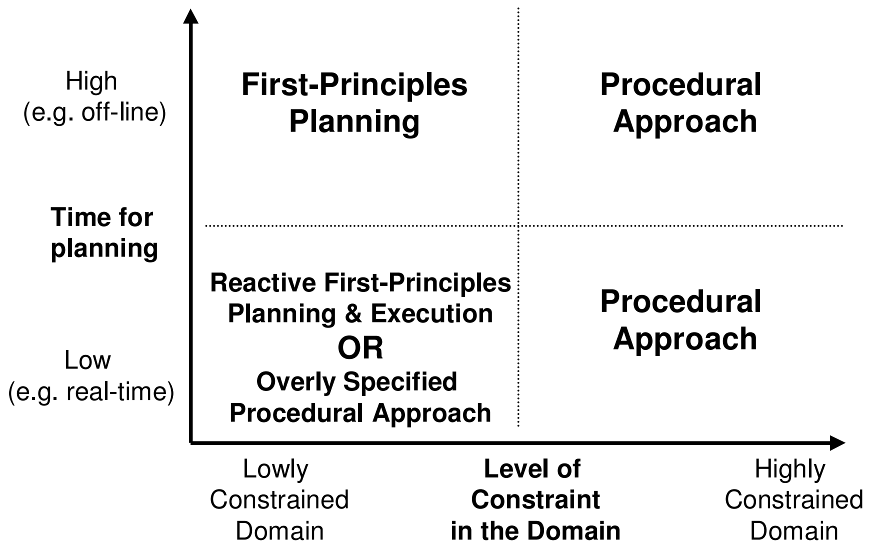

High
(e.g. off-line)
Time for planning
Low
(e.g. real-time)
First-Principles Planning
Procedural Approach
Reactive First-Principles Planning & Execution
OR
Overly Specified Procedural Approach
Procedural Approach
Lowly Constrained Domain
Level of Constraint in the Domain
Highly Constrained Domain

Figure 6. Type of agent planning approaches suited to various domain characteristics (how constrained the domain is and the time the agent has to plan).

If the domain is not highly constrained, then processes to achieve certain goals may not be known before hand, or there may be many plans that are possible due to a large number of variables in the domain. For example, in our transportation domain that we discuss in the next few chapters, an agent requires a distributed plan to achieve its transportation goal. There may be many possible routes with many different modes of transport that a resource could travel to get to its destination. Defining every possible route would not be practicable. Additionally, some routes would depend on the availability of transportation services that are available at the time. A developer may not know what type of transportation services will be present in order to predefine the agent plans. Therefore, a procedural approach to determine the social distributed plan to achieve transportation goals may not be appropriate.

From Figure 6, if the agent has a large amount of time available for planning, then it may plan from first-principles. If the agent does not have a large amount of time to plan in the particular domain (e.g. a real-time system), then there are two possibilities. It may perform reactive first-principles planning and execution, or means-ends analysis (Newell and Simon 1972). As the agent is forming each step (or multiple steps) in the plan, it is executing it. Deliberative reasoning may be required in order to allow the agent to select suitable (primitive) actions to achieve its goal. The second possibility is for the agent to use a procedural approach, but have an overly specified plans library where many plan possibilities have been defined. Therefore, for most situations, the agent may be able to find a suitable plan to achieve its current goals. This approach may require extensive effort to set up the plan library, extensive memory to store the plans, and the agent may only use a small proportion of the plans available during execution. It may be necessary to do this, for example, because a reactive first-principles and execution approach does not produce suitable plans due to its reactive nature.

With our transportation domain (see chapter 7), although the domain is not constrained, there is a reasonable amount of time for the agents to form a distributed plan to achieve their transportation goals. Therefore agents use a first-principles planning approach – top left quadrant of Figure 6. Since our transportation domain is highly complex and decentralised (requires communication), a reactive first-principles approach is taken (depth-first search) since a deliberative search may not be practicable. In the following chapters we devise an agent protocol, which is a constrained social agent process, in order to enable agents to find a distributed plan to achieve their (transportation) goals. The plan is an unconstrained social agent distributed plan.

3.2.4 Capturing Expertise – Collate and Analyse Information

Organisational expertise, which includes information (or knowledge) and (logical and declarative) business rules, can be embedded into agents. Expertise can be used to *collate and analyse information* that is gathered or stored by the agent (including *logistics* plans formed – not to be confused with ATTITUDE agent plans), and thus *deduce logistics facts and check logistics constraints*, as well as monitor changes that may occur in the agent's environment. This reasoning ability is important for IG agents so that they can analyse information in their domain of expertise (e.g. weather, terrain, etc.) and provide this expertise (e.g. logistics facts) to other agents. Agents can use expertise to influence their behaviour. For example, if a constraint in a plan that the agent intends to use is not satisfied, then the agent can respond accordingly, and run a different plan to achieve the same goal.

Rule: check if heavy trucks can travel on roads at the destination

If the destination has dirt roads AND it is raning, then the roads are muddy.

If roads are muddy, then heavy trucks cannot travel on them.

Figure 7. Rule to check if the Bde's heavy trucks can travel at the deployment destination.

An example will be used to illustrate the capturing of expertise in (ATTITUDE) agents. The Bde_Agent discussed in the previous section may need to check the constraint (or infer the logistics fact) about whether their military trucks can travel on the roads at the deployment destination. For example, heavy trucks cannot travel on muddy roads as they get bogged quite easily. If this is the case, it may be appropriate to bring smaller trucks. The Bde organisation (or people within it) knows from experience that if the deployment destination has dirt roads and it is raining at the destination, then the roads are likely to be muddy and heavy trucks are unlikely to be able to travel on them. This expertise (or logical rule) is shown in Figure 7 (purposely represented as two separate rules).

This expertise can be embedded into Bde_Agent using two methods in ATTITUDE, shown in Figure 8 and Figure 9. The first method, Figure 8, uses inference rules (declarative approach). The first instruction (believe propositional attitude) enters the rule “trucks cannot travel on a road ?road, IF the road ?road is muddy” into the agent’s knowledge base, as described in Figure 7. The second believe propositional attitude will enter the inference rule “?Y is muddy, IF ?Y is dirt AND it is raining at ?Y” as described in Figure 7 – note that ?Y in this rule does not have to be a road. The next *ask if believe* propositional attitude will query the agent to check if it believes (or it can infer) that trucks will not be able to travel on a particular road. If successful, the variable ?someRoad will be bound to the name of the road that the trucks will not be able to travel on. The *ask if believe* propositional attitude also contains an *inference token* truckQuery. This is used by the *ask again* instruction in the last line in Figure 8 to check other possible inference matches, i.e. if there are other roads that trucks cannot travel on. The *ask again* instruction can used as many times as required to find all possible answers, i.e. find all roads that trucks cannot travel on. If the *ask again* instruction is used, and no further answers that have not been presented are available, the *ask again* instruction will fail.

```
Bde_Agent believe (infer (cannot_travel trucks ?road) (muddy ?road)) in ?event 1.0

Bde_Agent believe (infer (muddy ?Y) (&(raining ?Y) (dirt ?Y))) in ?event 1.0

Bde_Agent ask if believe (cannot_travel trucks ?someRoad) in ?event with truckQuery

ask again truckQuery
```

Figure 8. Modelling expertise for analysis using inference rules.

```
( |           {comment: exclusive union operator}
  (^      {comment: join operator start}
            Bde_Agent ask if believe (raining ?where) in ?event
            Bde_Agent ask if believe (dirt ?where) in ?event
            Bde_Agent show (expression Trucks may not be able to travel on road ?where)
  )      {comment: join operator end}
  Bde_Agent show (expression Roads OK for trucks to travel on)
)
```

Figure 9. Modelling expertise for analysis using plans with control structures.

Figure 9 uses ATTITUDE control structures to make deductions (procedural approach). The first line is an exclusive union operator. It contains two plan alternatives, the first concerns the propositional attitudes inside the join operator (see Figure 9 for join operator start and end) and the second is the single show instruction. The first plan alternative in the exclusive union operator will execute initially. The first *ask if believe* propositional attitude will check the agent's knowledge base if it believes it is raining at ?where. If this succeeds, the plan continues, and the agent again checks the knowledge base if it believes the roads at ?where are dirt roads. If that propositional attitude succeeds, then the agent will print to the screen the message that trucks cannot travel on the roads. If any of the *ask if believe* propositional attitudes fail, then it is not raining or the roads are not dirt roads (or both), in which case the first plan alternative will fail, and the second plan alternative will execute, printing the message that the roads are "ok" for the trucks to travel on.

In the past, there have been debates on the suitability of declarative and procedural approaches. These debates have been largely resolved and both are found to be useful (Winograd 1975). ATTITUDE allows both approaches to be used for agent development.

3.2.5 Coalition Agent eXperiment (CoAX)

DARPA's Coalition Agent eXperiment (CoAX) (Allsopp and al. 2002; Perugini, Wark et al. 2003; Wark, Zschorn et al. 2003) project was aimed at demonstrating the utility of agents for coalition planning. Some 20 organisations from the USA, UK and Australia were involved. These include: the Air Force Research Laboratory (USA); the Artificial Intelligence Applications Institute, University of Edinburgh (UK); BBN Technologies (USA); Carnegie Mellon University (USA); Dartmouth College (USA); Defence Science & Technology Organisation (Australia); Global Infotek Technologies Inc; the Institute for Human and Machine Cognition, University of West Florida (USA); Lockheed-Martin Advanced Technology Laboratory (USA); the Naval Research Laboratory (USA); the Potomac Institute (USA); QinetiQ, Malvern (UK); the University of Maryland (USA); the University of Michigan (USA); and the University of Texas (USA). MALT was implemented within CoAX (Oct 2002), with the aims of demonstrating components of MALT; the use of agents for military logistics planning in the Australian-Coalition contexts; interoperability with foreign agents (i.e. agents not developed by us); and the dynamic capability of MALT.

Our part of the CoAX project involved a vignette where an Australian ship was struck by a torpedo, resulting in damage to the ship and casualties. A medical evacuation (medivac) is required to transport the casualties from the ship to a coalition medical facility using available helicopters. Agents represented the ship's medevac function (medevac agent) and onboard resources, and a single proxy agent represented the coalition helicopter and medical facility resources, providing information regarding the availability of medical facilities to treat casualties, and the availability of helicopters to transport the casualties. The medevac agent modelled the logistics process of planning a medical evacuation from the ship. The agents cooperated in order to form a medevac logistics plan.

The medevac agent was developed using ATTITUDE. The logistics process for a medevac was represented as a plan, and executed when casualties were detected on the Australian ship, triggering the "medevac" function (or goal). Cooperation (e.g. communication) between the medevac agent and the other agents were facilitated by the DARPA's CoABS grid (Kettler).

The medevac agent, when triggered, requests the availability of medical facilities to treat its casualties, and helicopter resources to plan the transportation of casualties from the Australian ship to the nearest suitable medical facility. The proxy agent responds with available medical facilities and helicopters, providing distances to facilities, start location, earliest start time, and types of helicopters available. The medevac agent uses prior knowledge of the carrying capacity and speed of the types of helicopters.

A simple algorithm is used by the medevac agent to form a transportation plan. The helicopter that can transport the injured to the medical facility at the earliest time is selected to perform the transportation task. Highest priority casualties are transported first. If the selected helicopter cannot transport all the injured, the process is continued with the remaining injured.

The plan formed is sent to the foreign Multi-Level Coordination Agent for processing, to deconflict and optimise (merge) the medevac plan with existing flight plans developed by foreign (coalition) agents. The plan is then distributed to the appropriate coalition helicopters and medical facility for execution. The medevac agent reacts to any changes, such as helicopter availability, the number and type of casualties, or the availability of the helicopter landing pads and replans if necessary.

The CoAX demonstration was held in October 2002 at the US Navy Warfare College, Newport RI. It successfully presented components of MALT in operation, and thus demonstrated the use of agent technology for military logistics planning. DARPA's UltraLog (or ALP) and CDM Technologies JFCT has also demonstrated that agents can be used for aspects of the U.S. logistics planning domain (see section 2.4.1), but we are focusing on the Australian-Coalition contexts, and using the BDI agent paradigm. The logistics process of a (ship) medevac was effectively modelled in the medevac agent using ATTITUDE. The medevac agent successfully cooperated with foreign agents (all agents used the same simple social protocol), sending the final medevac plan to the foreign Multi-Level Coordination Agent for processing and distribution to coalition helicopters and medical facility. The medevac agent was able to dynamically replan when the situation changed.

3.2.6 Brigade Agent

In sections 3.2.1and 3.2.4, details of an agent Bde_Agent were given, which modelled the logistics process of deploying a Bde unit, as observed and performed at the military logistics training course. The agent was partially implemented – the first two goals (steps) in Figure 4 (*gather information* and *sustainment calculation*), and some components of the last goal (*analyse plan*).

With *gathering information*, one of the examples used was that presented in section 3.2.4 – for the agent to determine whether heavy trucks can use the roads at the deployment desination. Two IG agents were set up to provide information on weather (if it is raining) and terrain (if the roads are dirt) at the deployment destination. Bde_Agent first gathered the required information from the IG agents by sending requests for information to the particular agent, in which they replied with the information requested. Once the information was received, and thus is stored in Bde_Agent's knowledgebase (as beliefs about the world), one of the rules in Figure 8 and Figure 9 were used to deduce the fact of whether the Bde heavy trucks were able to use the roads at the destination. This fact was used to determine whether heavy trucks were to be deployed, and thus influenced the calculations of the quantity of resources (e.g. trucks) to be transported to the destination.

Rules (expertise) were also used to analyse the parts of the logistics plan currently formed (for the *analyse plan* goal). A similar example as above, if the roads at the destination are rough, then certain vehicles could have their movement restricted. A rule was written to check if the surface at the destination was rough (information gathered from the terrain agent), and if so, if a particular vehicle was selected to be deployed, then this fact would be flagged to the user, making the user aware of the risk with the plan. Agent plans were used by Bde_Agent to analyse the current elements of the logistics plan formed (primarily supplies required by the deployed Bde unit) – modelling processes (Bde) planners will use to analyse their logistics plan. Bde_Agent provided information such as the quantity of the various types of supplies required in the operation and the financial cost (price) expected to purchase supplies for the logistics plan. Agents need only release logistics plan information that is required by the users, hiding logistics plan (or planning) details. Information provided could be customised for particular users, depending on their information requirements, or could be set up such that abstract logistics plan information is first provided, and further details can be provided if requested.

Bde_Agent also performed *sustainment calculations* as performed in the logistics training course, working out the quantity of various supplies required to support the deployed Bde unit. In the training course, these repetitive and numerous calculations were done manually, with a calculator. They were tedious, time consuming, and prone to error, where one error earlier on in the calculations may carry on in later calculations. Bde_Agent, being software, was able to perform these mathematical calculations very quickly without error (if code and hardware not erroneous). Bde_Agent performed the calculation in a matter of seconds (under 3 seconds) compared with taking over 15 minutes for manual calculation at the military logistics training course. At the training course, if there was a change in the logistics goal, such as a change in the size of the Bde force to be deployed (due to the campaign planners changing the campaign plan), then these sustainment calculations were perfomed again, taking at least another 15 minutes. Bde_Agent is able to replan and perform these calculations again for the training course scenario almost instantly. The Joint Operational Logistics Tool Suite (JOLTS), which can also perform logistics sustainment calculations using Excel *software*, is also able to perform the calculations very quickly. Therefore software, and hence agents, when it comes to calculations required in logistics planning, is likely to be much faster and more responsive to the dynamic nature of logistics goals, than performing the calculations manually (as observed at the training course), due to the speed and accuracy that software can perform calculations.

Bde_Agent was developed so that users could enter parameters allowing them to control Bde_Agent's planning process. Items in the planning process that the user could control include which individual Bde elements to deploy, and information or values used in the sustainment calculations (such as the consumption rate and whether the operation is a peacekeeping operation). If these parameters are not given, the agent will use default values or attempt to gather information in order to deduce appropriate values (in the Bde_Agent prototype, "information gathered" in this instance was already stored locally in the agent's knowledge base). Therefore, Bde_Agent was developed such that users can direct as much control over the agent, and its planning process and actions, as they desire.

Users may provide Bde_Agent with an abstract goal, without any details, if for example, they trust the agent will use the correct values and perform the correct actions, or want approximate values/plans for a logistics goal (hence, exact values are not a great concern). Otherwise, users may enter more detailed instructions and information to Bde_Agent, giving the user greater control over Bde_Agent's actions.

The components of Bde_Agent that have not yet been fully completed are the scheduling of supplies and transportation. In order to schedule supplies and transportation, Bde_Agent must cooperate and obtain services from other agents – cooperate with SA to obtain supplies and TA to obtain transportation services. The transportation scheduling component was attempted first. It was discovered that such a task was not trivial, due to the complexity of the transportation scheduling (or planning) problem required by the military, and that planning was to be performed in a decentralised, dynamic and open environment, requiring a complex social protocol to facilitate the planning. The next few chapters provide further details regarding the protocol and the transportation implementation.

3.3 Discussion

MALT was presented, using agents to automate aspects of logistics planning and information gathering. This chapter reflects our experience in implementing components of MALT using the BDI agent paradigm. The implementation of two agents in MALT was described (note that due to the sensitivity of the information, further details regarding implementation and results could not be provided). In the description, a methodology was presented to model the organisation's (business) processes and expertise, and to embed agents' autonomous, proactive (goal-directed), reactive and social characteristics, using the BDI-based agent programming language ATTITUDE. Agent (BDI) concepts and our methodology enable the developer to think about failures and possible events when developing the agent, producing a robust software system. Types of domains where BDI, or procedural, approaches are appropriate are in highly constrained domains or domains with little time available for planning (e.g. real-time systems). Most organisations in MALT are highly constrained. In lowly constrained real-time systems, considerable effort may be required to develop the agents due to the large number of plans required to consider many possible situations. The ease at which agents can be developed to model organisations' (individual and social) processes and expertise to produce a robust software support system indicates that they are suitable for modern military logistics. Agent technology provides a suitable conceptual framework by which agents can be developed for elements within the logistics domain.

Jennings *et al.* investigates using agents for business process management in a system called Advanced Decision Environment for Process Tasks (ADEPT) (Jennings, Faratin et al. 1996). Similar to our domain, agents represent various entities within their domain and automate aspects of their business processes. Brazier *et al.*'s DESIRE framework enables high-level specification of a system's conceptual design (Brazier, Dunin-Keplicz et al. 1997). It models the knowledge, interaction and coordination of complex tasks in agent systems. The GAIA methodology (Wooldridge, Jennings et al. 2000), ROADMAP (Juan, Pearce et al. 2002), the architecture-centric method (Park and Sugumaran 2005) and Prometheus (Padgham and Winikoff 2002) provide methodologies for developing agent systems. Taveter and Wagner propose an agent-oriented approach to representing and visualising agent business rules and processes, which uses a declarative agent approach (Taveter and Wagner 2001). Graham uses an Object-Oriented approach to modelling business processes with agents (Graham 1997). Bose designs agents with object-oriented technology and implements agents using Prolog in order to automate organisational tasks (Bose 1996).

These agent methodologies and systems typically investigate the social structure of agents, their social or abstract roles and functions, behaviours and interactions, their general architecture, or do not consider the BDI paradigm and its intricacies. None of the above mentioned literature investigates the methodology of developing the individual agent plans, and thus the agents, using a BDI framework – the transition from a description of the business processes and expertise into the final (BDI) agent – and how to obtain desirable agent properties such as reactivity and proactiveness.

Chapter 4

4 Contract Net Approaches to Task Allocation

In this Chapter an analysis and comparison of existing contract net protocols is explored, to determine their appropriateness to perform the planning and task allocation required in our logistics domain. The protocols we compare are Smith's Contract Net Protocol (CNP) (Smith 1980), Fischer *et al.*'s Extended Contract Net Protocol (ECNP) (Fischer and Kuhn 1993; Fischer, Muller et al. 1996) and Aknine *et al.*'s CNP extension, which we will refer to as CNP-ext (Aknine, Pinson et al. 2004). To explore the different contract net approaches, processes of planning and task allocation among agents are defined. Additionally, we present a new protocol representation, called the Protocol Flow Diagram. We show that the CNP, CNP-ext and ECNP fall short in addressing the planning and task allocation required in *our* logistics domain, in particular our transportation domain.

4.1 Planning, Task Allocation, and Protocol Processes

In this thesis, we consider a multi-agent system to comprise of a set of agents $\mathscr{A} = \Lambda \cup \Gamma$, consisting of a set of auctioneers $\Lambda \subseteq \mathscr{A}$ and a set of bidders $\Gamma \subseteq \mathscr{A}$, which are not necessarily mutually exclusive sets. Each auctioneer $\alpha \in \Lambda$ has a goal G that it wishes to achieve by the achievement of a set of tasks T. Note that the auctioneer need only achieve some subset of its set of tasks $T^{alloc} \subseteq T$ to achieve G. The partial achievement of T is allowed in the combinatorial auction application domain discussed in chapter 6. Specifically, an allocation of a subset, T^{alloc} of T, may be preferred (e.g. is more profitable) over an allocation of the complete set of tasks T. Each bidder $\gamma \in \Gamma$ holds a possibly empty set of bids B that may be used to achieve the set of tasks desired by the auctioneers. Auctioneers allocate tasks to bidders according to the merit of their bids.

The three types of planning and task allocation problems that are considered in this thesis are as follows:

1. *Task allocation* – An auctioneer α allocates its set of tasks T^{alloc} to a *single* bidder γ in order to attain its goal G. The auctioneer accepts a single bid b from bidder γ, which can completely achieve T^{alloc} (i.e. achieve all tasks in T^{alloc}) and is as preferable as any other possible achieving bid available by any bidder in Γ.

2. *Task allocation with bid planning* – An auctioneer α allocates its set of tasks T^{alloc} to a *set of bidders* $\Xi \subseteq \Gamma$ to attain its goal G. The auctioneer must find and accept a set of bids $\boldsymbol{B}^{\boldsymbol{acc}}$ (the bid plan) by bidders Ξ. Each bid $b \in \boldsymbol{B}^{\boldsymbol{acc}}$ by bidder γ achieves a subset of the set of tasks $\Theta \subseteq T^{alloc}$ (set of tasks Θ allocated to γ) and $\boldsymbol{B}^{\boldsymbol{acc}}$ is as preferable as any other possible achieving set of bids by bidders in Γ.

3. *Planning and task allocation* – An auctioneer α must find a specific set of tasks T^{alloc} to allocate to a *set of bidders* $\Xi \subseteq \Gamma$ to attain its goal G, where there are many sets of tasks (plans) that can achieve G. The auctioneer must find a suitable set of tasks T^{alloc}, as well as find and accept a set of bids $\boldsymbol{B}^{\boldsymbol{acc}}$ (the bid plan) by bidders Ξ that can achieve T^{alloc}. The set of tasks T^{alloc} and set of bids $\boldsymbol{B}^{\boldsymbol{acc}}$ are as preferable as any other possible set of tasks and achieving set of bids, and each bid $b \in \boldsymbol{B}^{\boldsymbol{acc}}$ by bidder γ achieves a subset of the set of tasks $\Theta \subseteq T^{alloc}$ (set of tasks Θ allocated to γ).

Problem (1) is special case of problem (2) which is a special case of problem (3). Note that auctioneers are manager agents (MA) and bidders are transport agents (TA) in the terminology used in Chapter 3. The terms MA and TA will be used when we are referring to the protocols applied to the transportation scheduling domain, and we will use the terms Auctioneer and Bidder elsewhere, including applying the protocol to combinatorial auctions.

In order to address the problems above, auctioneers and bidders undergo negotiations. In a negotiation, agents follow a *protocol*, which specifies the states of an interaction, i.e. the set of speech acts (or messages) and events, between an auctioneer α and a bidder γ in order to accept a bid and ultimately commit to the bid to fully or partially achieve an auctioneer's set of tasks $\Omega \subseteq T$. We define *backtracking* as the process of undoing the acceptance (or selection) of a bid in order to find another bid option to achieve the set of tasks. A *protocol process* is the execution of the protocol (the negotiation) between an auctioneer and a bidder. We define a negotiation, or protocol process, as a 9-tuple:

$Neg = < \alpha, \gamma, \Omega, \Sigma, S, A, \delta, s, F >$

- $\alpha \in \Lambda$ where α is an auctioneer from the set of auctioneers Λ.
- $\gamma \in \Gamma$ where γ is a bidder from the set of bidders Γ.
- $\Omega \subseteq T$ is the set of tasks from α that the agents α and γ are negotiating over (see below).
- $\Sigma \subseteq B$ is the set of bids from γ that the agents α and γ are negotiating over.
- S is the set of possible states in the protocol.
- A is the set of possible speech acts or events that can occur in the protocol resulting in a transition between states. Speech acts are the messages communicated between the agents. Events are situations, which are not speech acts, that result in the protocol changing state, e.g. no communication.
- $\delta \subseteq (S \times A \times S)$ is a set of triples which specifies the transitions between all the possible states of the protocol, i.e. <*current state,* ***speech act or event****, next state*> $\in \delta$.
- $s \in S$ is the start state of the negotiation (protocol process) between α and γ.
- $F \subseteq S$ is the set of final states of the negotiation between α and γ.

We will see later that an initial protocol process by an auctioneer to achieve T may result in T being only partially achieved by a bidder's bid. New protocol processes for the set of tasks $\Omega \subseteq T$ that were not achieved are then executed between the auctioneer and bidders (see section 4.7.5). Partial achievement of Ω may result in further protocol processes to achieve $\Omega' \subseteq \Omega$, and so on. The set of tasks allocated (achieved by the selection of bidders' bids) by the auctioneer α through the execution of one or more protocol processes associated with the set of tasks T is $T^{alloc} \subseteq T$.

Due to the agent system's many-to-many setting, an auctioneer α may be negotiating with a set of bidders simultaneously regarding the achievement of its set of tasks Ω. Similarly, a bidder γ may be negotiating with a set of auctioneers Λ regarding the achievement of the set of each auctioneers' set of tasks, i.e. $\bigcup\{\Omega_\alpha \mid \alpha \in \Lambda\}$.

4.2 Contract Net Protocol and the Protocol Flow Diagram

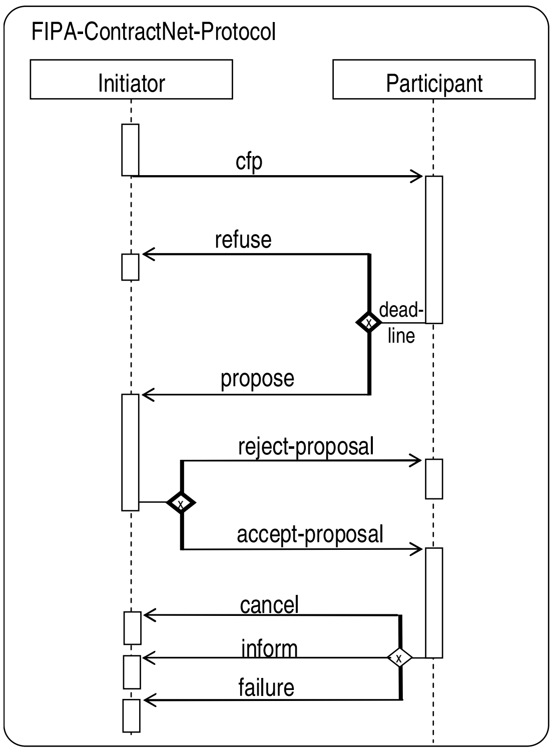


Figure 10. Conventional representation of the Contract Net Protocol (CNP). cfp represents Calls for Proposals.

A conventional representation of the contract net protocol (CNP) is shown in Figure 10, which was obtained from the Foundation for Intelligent Physical Agents (FIPA) (FIPA 2005). CNP operates similarly to a first-price sealed-bid auction (or reverse auction). An initiator agent announces a call for proposals (**cfp**), i.e. a task that it would like achieved, to one or more participant agents. The participant agents may either submit a **proposal** (or bid) to achieve the *complete* task or **refuse** to submit proposals. A participant agent's proposal may specify details such as the time at which the task can be performed and the price that must be paid in order for the participant to execute the proposal. After some deadline specified by the initiator, the proposer will select the best proposal, based on some criteria depending on the particular situation, and accepts this proposal. The initiator sends an **accept-proposal** speech act (*or message*) to the appropriate participant agent. The initiator then rejects all other proposals and sends a **reject-proposal** speech act to all other participants. The participant agent that had its proposal accepted will perform the task and inform the proposer of the completion of the task.

The conventional representation of CNP makes it difficult to represent the other contract net-based protocols, such as CNP-ext, ECNP and our Provisional Agreement Protocol (PAP). These protocols have complex state transitions, such as repeated states. Therefore, we define a new representation, called the *Protocol Flow Diagram*, which is essentially a modified event flow diagram. Figure 11 illustrates CNP (using different terminology) using the Protocol Flow Diagram representation. It may be easier for a reader to comprehend an illustration such as that in Figure 10 and Figure 11, rather than pseudo-code, to describe speech acts and events between agents. An initial description of PAP using pseudo-code required six separate algorithms to describe the various step(s) of the protocol among the agents. Each algorithm was linked with other algorithms (via speech acts or events between agents/algorithms) (Perugini, Lambert et al. 2003), which may not be easy to follow.

In the Protocol Flow Diagram representation in Figure 11, each vertical line represents the agents involved in the protocol (the auctioneer and bidder agents). There are also three matching boxes on each vertical line. These three pairs of boxes correspond to the three steps in the protocol, as denoted by the brackets on the far right hand side of Figure 11. Each pair of boxes represents an eXclusive-OR (XOR) step in the protocol, and therefore, only one speech act or event may occur at that step. In step one of CNP, only one speech act (**Task Announcement**) may occur, but in step two, either the **Bid** speech act or the **Refuse** speech act must occur, but not both. Similarly, in step three, either the **Grant Bid** or the **Reject Bid** speech act must occur, but not both. The XOR applies between two agents that are interacting because, for example, an auctioneer (initiator) may be announcing a task to many bidders (participant), and thus be receiving **Bids** from some agents and **Refuse** from others. In this case, it receives both speech acts, but only receives one or the other from each agent that it is interacting with. The auctioneer may be at different steps in the protocol between different agents.

Each speech act or event between agents in the Protocol Flow Diagram has an associated *dotted line* that emanates from it and protrudes outside the vertical lines. The dotted line is used to describe the next step in the protocol if that speech act or event occurs. In Figure 11, **Task Announcement** speech act in step one has a dotted arrow emanating from it and points to the second XOR box (from the top) along the vertical line, which represents step two of the protocol. Therefore, after the task announcement, the protocol proceeds to step two. In step two, if the auctioneer receives **Bids** from the bidders, then the arrow emanating from the **Bids** speech act indicates that the protocol proceeds to step three (third XOR box from the top). If a bidder sends a **Refuse** speech act, or the auctioneer sends a **Reject Bid** speech act, then the dotted line ending with a diamond emanating from that speech act indicates that the protocol (process) between the two respective agents ends *without a contract* (note that the protocol process for the same task announcement may still be running between the auctioneer and other bidders). When the auctioneer sends a **Grant Bid** speech act, the protocol process between the two respective agents ends *with a contract*. This is indicated by the dotted line ending with an oval emanating from the **Grant Bid** speech act.

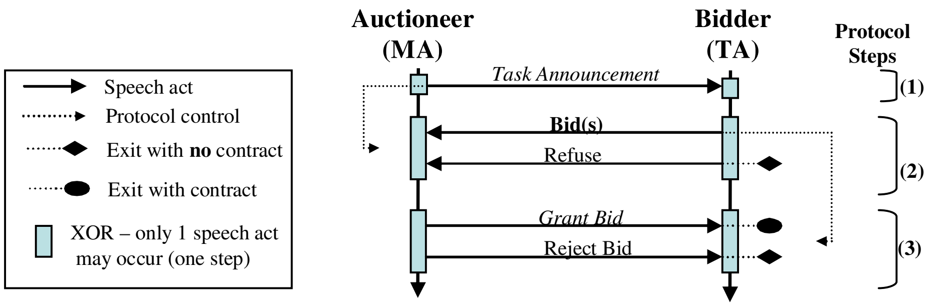


Figure 11. CNP, using the Protocol Flow Diagram representation.

Each step in the Protocol Flow Diagram, in addition to the two type of exits (with and without contracts) are the *states* of the protocol *shared* by the agents involved in the negotiation. The CNP, and thus the two agents using CNP, have 5 states, which are the three protocol steps and the two types of exits. The dotted control lines indicates the transition between states if a particular speech act or event occurs. Further details regarding this for the CNP is discussed in section 4.4.1. The side (auctioneer or bidder side) that the dotted control lines and exits appear are irrelevant.

Some additions to the Protocol Flow Diagram representation are used to illustrate ECNP and PAP. These additions will be introduced later when they are required. CNP representation in Figure 11 removes the last three speech acts (responses related to the completion of the task) from FIPA's original CNP description because as we are concerned with the allocation of tasks, rather than the successful execution of these tasks after they are allocated. Our representation also uses different terminology to the FIPA's description of CNP. Call for proposals (**cfp**) is termed **Task Announcement**, **proposals** is **Bid(s)**, **accept-proposals** is **Grant Bid** and **reject-proposals** is **Reject Bid**. The names of the two types of agents are also different. The initiator is the Auctioneer and the participant is the Bidder.

In the protocol description above, we use the term speech acts *and events* rather than only speech acts. In the CNP description, all steps involve speech acts, which are messages communicated between the agents, and the dotted line indicates the next state when that speech act occurs. Later, when discussing PAP, the protocol process may move to a new state without any messages being communicated, for example, if the auctioneer decides to *backtrack* or if there is *no communication* between the agents in a step and they move on to the next state (agents may realise the event has occurred when the next speech act is sent). Therefore, we refer to such conditions as events in the protocol, which cause a transition to a new state in the protocol process.

4.3 Contract Net Protocol Applied to Transportation Scheduling

In the transportation domain, the task announcement in CNP (see Figure 11) may be a transportation task that the Auctioneer, or MA, desires to be achieved. Bids sent by the bidders, or TA, may be *services* (actions) that they can perform to achieve the transportation task. Bids may contain times that the task can be achieved and a price that the MA must pay for the bid. The MA may select the most appropriate bid, based on criteria required of the task, which may include the price of the bid and the time to complete the transportation task.

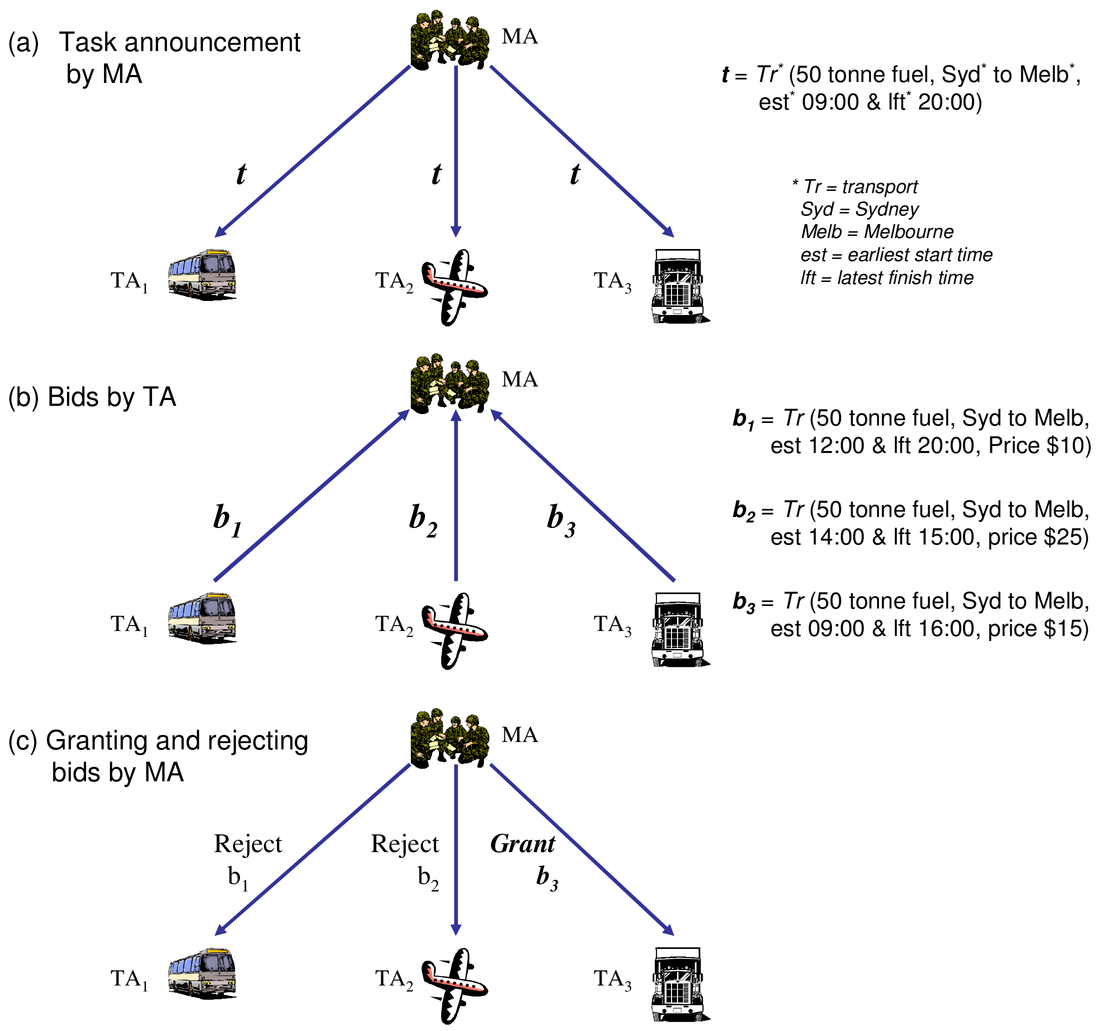

(a) Task announcement by MA
MA
t = Tr* (50 tonne fuel, Syd* to Melb*, est* 09:00 & lft* 20:00)
t
t
t
* Tr = transport
Syd = Sydney
Melb = Melbourne
est = earliest start time
lft = latest finish time
TA1
TA2
TA3
(b) Bids by TA
MA
b1 = Tr (50 tonne fuel, Syd to Melb, est 12:00 & lft 20:00, Price $10)
b2 = Tr (50 tonne fuel, Syd to Melb, est 14:00 & lft 15:00, price $25)
b3 = Tr (50 tonne fuel, Syd to Melb, est 09:00 & lft 16:00, price $15)
b1
b2
b3
TA1
TA2
TA3
(c) Granting and rejecting bids by MA
MA
Reject b1
Reject b2
Grant b3
TA1
TA2
TA3

Figure 12. Example of CNP applied to the transportation scheduling domain. ***t*** *= transportation task,* $\boldsymbol{b_x}$ *= bid to achieve* ***t****. From problem 1 in section 4.1,* $\{t\} = T^{alloc} = T$*, to attain goal G of moving 50 tonnes of fuel from Sydney to Melbourne.* $\alpha = MA$*,* $\Gamma = \{TA_1, TA_2, TA_3\}$*,* $b = b_3$*. Bid evaluation criteria is to minimise price and latest finish time.*

Figure 12 illustrates an example. In Figure 12 (a), a manager agent (MA), which may represent an army unit, announces a task ***t*** to three transport agents (TA) to transport 50 tonnes of fuel from Sydney to Melbourne, with earliest starting time of 09:00 and latest finish time of 20:00. In Figure 12 (b), the three TA respond with bids b_1, b_2 and b_3, respectively, to completely achieve the task. b_1 transports the fuel with earliest starting time of 12:00 and latest finishing time of 20:00, with price $10; b_2 transports the fuel with earliest starting time of 14:00 and latest finishing time of 15:00, with price $25; b_3

transports the fuel with earliest starting time of 09:00 and latest finishing time of 16:00, with price \$15. The MA evaluates the bids in order to determine which is most appropriate for selection. From Figure 12 (c), the criteria that the MA uses to evaluate the bids are to minimise price and the latest finish time. Since this is an *illustrative* example, we ignore details of calculations the MA may use to evaluate the bids. The MA decides that b_3 is the most appropriate bid and grants b_3, and rejects b_1 and b_2 – as b_3 has the best compromise between price and time, where b_1 is cheap but arrives late and b_2 arrives earliest but is expensive.

4.4 CNP Analysis

4.4.1 CNP Protocol Description

In the previous section, CNP was presented diagrammatically using the Protocol Flow Diagram. We will define CNP using the negotiation (or protocol process) description presented in section 4.1, and the Protocol Flow Diagram illustration in Figure 11. We have a negotiation defined as

$$Neg = < \alpha, \gamma, \Omega, \Sigma, S, A, \delta, s, F >$$

where: α is an auctioneer in the negotiation; γ is a bidder in the negotiation; $\Omega = T$ is the set of tasks that the agents α and γ are negotiating over for its full or partial achievement (with *one bid* to attain goal G); Σ is the set of bids that the agents α and γ are negotiating over. The other tuples are dependent on the protocol:

- S = {step1, step2, step3} $\cup$ {exit^{NC}, exit^{C}} is the set of possible states in the protocol. exit^{NC} is the state which exits with *no* contract and exit^{C} is the state which exits with a contract.
- $A = A_\alpha \cup A_\gamma \cup A_e$ is the set of possible speech acts or events where: A_α is the set of speech acts associated with the auctioneer; $A_\gamma$is the set of possible speech acts associated with the bidder; and A_e is the set of events (common to both agents). A_α = ({**task announcement**}$\times \Omega$) $\cup$ ({**grant bid**}$\times \Sigma$) $\cup$ ({**reject bid**}$\times \Sigma$); A_γ= ({**bid**}$\times \Sigma$) $\cup$ ({**refuse**}$\times \Omega$); $A_e = \emptyset$.

- $\delta \subseteq (S \times A \times S)$ specifies the transitions between the possible states, such that δ is a function, with δ(step1, < **task announcement**, ω>) = step2; δ(step2, < **bid**, σ>) = step3; δ(step2, < **refuse**, ω>) = exit^{NC}; δ(step3, < **grant bid**, σ>) = exit^{C}; δ(step3, < **reject bid**, σ>) = exit^{NC}.
- s = step1, is the start state.
- $F = \{\text{exit}^{NC}, \text{exit}^{C}\}$ is the set of final states.

4.4.2 Communication

The best-case communication requirements, which is the number of messages (speech acts) sent and received, for CNP to achieve an auctioneer's set of tasks will be formulated. This allows us to compare the best-case communication requirements of CNP to the worst-case communication requirements of our Provisional Agreement Protocol (PAP) in chapter 5.

Refer to the CNP specification in Figure 11. Assume a maximum of κ bidders and a maximum of br bids received per task announcement. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, therefore communicates κ messages. At step 2, the bidders submit (communicates) br **bids** – in the best case, no **refuse** messages are communicated. At step 3, the auctioneer sends a **grant** for it most preferred bid, and sends **reject** messages for every other bid, and thus br messages are communicated. Therefore, the total communication required by CNP for a task allocation is the sum of the number of messages communicated:

Eq 1: Total communication for CNP $= \kappa + br + br = \kappa + 2 \cdot br$

4.4.3 Memory

The best-case memory requirements, which is the number of tasks and bids that must be stored at any one time, for CNP to achieve an auctioneer's set of tasks will be formulated. This allows us to compare the best-case memory requirements of CNP to the worst-case memory requirements of our Provisional Agreement Protocol (PAP) in Chapter 5.

Refer to the CNP specification in Figure 11. Assume a maximum of κ bidders and a maximum of *br* bids received per task announcement. At step 1, the auctioneer **announces** its set of tasks to each of the κ bidders. Therefore, the auctioneer and each bidder must store *1* set of tasks. The protocol proceeds to step 2, where bidders submit *br* **bids**, and therefore, the auctioneer must store *br* bids and the bidders must store their *1* submitted bid. The protocol proceeds to step 3 where the auctioneer will **grant** its preferred bid and **reject** the other bids. Therefore, the bidders may delete the set of tasks and its bid, and the auctioneer may delete all the *br - 1* rejected bids. The total memory requirement for the auctioneer is *1* set of tasks and *br* bids, and for the bidders are *1* set of tasks and 1 bid, which is deleted if/when their bid is rejected.

4.5 Contract Net Protocol – Extension

CNP was extended by Aknine (Aknine, Pinson et al. 2004), which we refer to as *CNP-ext* (CNP-ext is not to be confused with Fischer's extension, ECNP). CNP-ext allows bidders to negotiate with many auctioneers simultaneously. The protocol is presented in Figure 13 using the Protocol Flow Diagram representation. There are five steps (and hence 7 states) in the protocol. In step 1, the auctioneer sends a **Task Announcement** to all bidders, and the protocol proceeds to step 2. Bidders may send a **PreBid** for the *complete task*, which are temporary bids for the task, or send nothing, **No Communication**. For both events the protocol proceeds to step 3. At step 3, the auctioneer selects the best pre-bid for the task, and gives the agent associated with the pre-bid a provisional grant, which temporarily grants the bid, and proceeds to step 4. All the agents associated with the unsuitable pre-bids are given either a **Provisional Reject**, in which case the protocol proceeds back to step 2 to allow the bidder to send an *updated* bid to improve their pre-bid for the task, or send a **Confirm Reject**, in which case the protocol exits without a contract for the auctioneer and bidder involved in the negotiation. At step 4, the bidder sends a **Definitive Bid**, which is a final bid for the task and *can differ to its pre-bid*, and the protocol proceeds to step 5. If the auctioneer accepts the definitive bid, then it sends the associated bidder a **Confirm Grant**, and the negotiation is complete (protocol exits with a contract) and the task has been allocated. Otherwise, the auctioneer may **Provisionally Reject** the bid because, for example, the definitive bid is worse than the pre-bid or other agents' updated pre-bids are better than the definitive bid (or both). In this case, the protocol proceeds to step 2 to allow the rejected bidder to send an updated pre-bid, and the auctioneer can select a new pre-bid at step 3. The auctioneer is also able to send a **Confirm Reject** at step 5 to the bidder associated with the provisionally granted pre-bid, and the protocol exits without a contract for the two agents.

Note that the CNP-ext specification in uses the terms pre-accept, pre-reject, definitive grant, and definitive reject where we have used the terms provisional grant, provisional reject, confirm grant and confirm reject, respectively. This allows us to keep terminology consistent with the terminology we have used with our PAP discussed in Chapter 5.

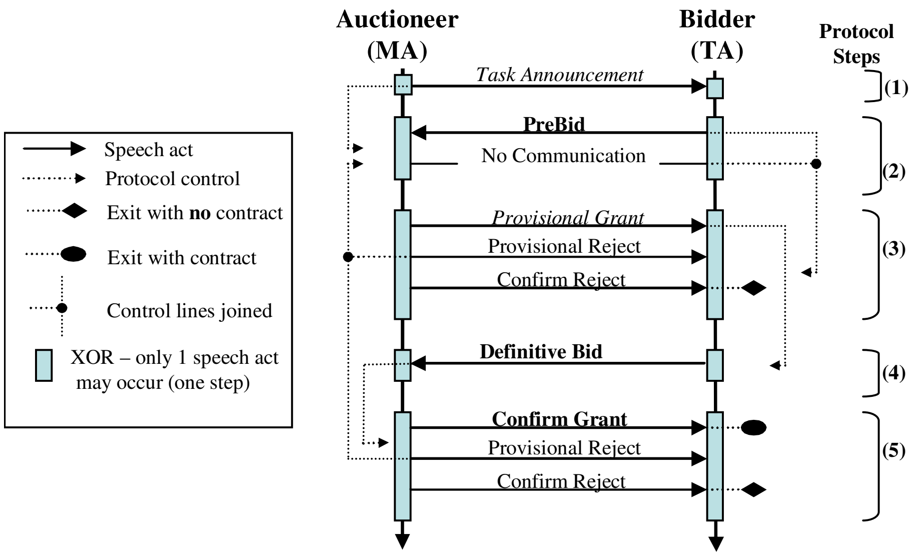


Figure 13. Aknine et al.'s CNP-ext.

With CNP, as if more than one task is received, the bidder may only bid for one task and wait for the negotiations for this task to complete (grant or reject the bid) before it may bid for the next task. Auctioneers for other tasks may have already allocated their tasks during this time, resulting in the bidder losing potential contracts. With CNP-ext, when tasks are received from various auctioneers, the bidder sorts the tasks, forms bids for the tasks and sends the bids to the various auctioneers. Therefore, CNP-ext allows bidders to negotiate with many auctioneers at the same time.

Bidders sort tasks received from auctioneers based on their preferences, where tasks (and their bids) lower in the order are dependent on tasks (and their bids) higher in the order. Tasks that commenced lower in the order and whose bids are rejected are moved higher in the order, replacing higher order tasks whose bids were rejected, so that the bidder can submit an improved (updated) bid for the lower order task.

4.5.1 CNP-ext Applied to Transportation Scheduling

CNP-ext in the transportation domain operates in a similar manner to CNP presented in section 4.3. The unsuitable bids b_1 and b_2 in Figure 12 (c) would first receive a provisional reject (in which case the TA would be able to update their bids, if possible) and then receive a confirm reject once a bid is confirm granted. The selected bid b_3 would first receive a provisional grant, the TA would then send a definitive bid b_3, and the auctioneer would send a definitive grant for b_3.

4.6 CNP-ext Analysis

4.6.1 CNP-ext Protocol Description

We will define CNP-ext using the negotiation (or protocol process) description presented in section 4.1, and the Protocol Flow Diagram illustration in Figure 13. We have a negotiation defined as

$$Neg = < \alpha, \gamma, \Omega, \Sigma, S, A, \delta, s, F >$$

where: α is an auctioneer in the negotiation; γ is a bidder in the negotiation; $\Omega = T$ is the set of tasks that the agents α and γ are negotiating over for its full or partial achievement (with *one bid* to attain goal G); Σ is the set of bids that the agents α and γ are negotiating over. The other tuples are dependent on the protocol:

- S = {step1, step2, step3, step4, step5} $\cup$ {exit^{NC}, exit^{C}} is the set of possible states in the protocol. exit^{NC} is the state which exits with *no* contract and exit^{C} is the state which exits with a contract.
- $A = A_\alpha \cup A_\gamma \cup A_e$ is the set of possible speech acts or events where: A_α is the set of speech acts associated with the auctioneer; A_γ is the set of possible speech acts

associated with the bidder; and A_e is the set of events (common to both agents). A_α = ({**task announcement**}× Ω) ∪ ({**provisional grant**}× Σ) ∪ ({**provisional reject**}× Σ) ∪ ({**confirm reject**}× Σ) ∪ ({**confirm grant**}× Σ); A_γ= ({**PreBid**}× Σ) ∪ ({**definitive bid**}× Σ); A_e = ({**no communication**}).

- δ ⊆ (S×A×S) specifies the transitions between the possible states, such that δ is a function, with δ(step1, < **task announcement**, ω>) = step2; δ(step2, < **PreBid**, σ>) = step3; δ(step2, <**no communication**>) = step3; δ(step3, < **provisional grant**, σ>) = step4; δ(step3, < **provisional reject**, σ>) = step2; δ(step3, < **confirm reject**, σ>) = exit^{NC}; δ(step4, < **definitive bid**, σ>) = step5; δ(step5, < **confirm grant**, σ>) = exit^{C}; δ(step5, < **provisional reject**, σ>) = step2; δ(step5, < **confirm reject**, σ>) = exit^{NC}.
- s = step1, is the start state.
- F = {exit^{NC}, exit^{C}} is the set of final states.

4.6.2 Communication

The best-case communication requirements, as in section 4.4.2, will be formulated. Refer to the CNP-ext specification in Figure 13. Assume a maximum of κ bidders, a maximum of br bids received per task announcement, and η is the number of repeated negotiations (a provisionally granted bid is rejected to select another bid) before a bid is confirm granted. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, and therefore communicates κ messages. At step 2, the bidders submit (communicates) br **pre-bids**. The protocol proceeds to step 3, which is the point where the negotiation repeating commences (see below). Since we are not at the end of planning, we ignore the **confirm reject** speech act. The auctioneer **provisionally grants** its preferred bid and **provisionally rejects** the other bids, and therefore communicates br messages. The bidders associated with the bids that were provisionally rejected proceed to step 2 and send an updated **pre-bid**, and therefore, communicate $br - 1$ pre-bids, and proceed to step 3. The agent associated with the bid that is provisionally granted, will send *1* **definitive bid**. For the best case communication (compared with PAP in chapter 5), the definitive bid must be different to the provisionally granted bid [9]. Therefore, the total number of bids sent after the provisional grants and rejects at step 3 is $br - 1 + 1 = br$. Once a definitive bid is submitted, the protocol proceeds to step 5. We can ignore the **confirm reject** speech act at step 5 as the CNP-ext specification states that it is only used to fully reject a definitive bid to discourage bidders from submitting a definitive bid that is much worse than the provisionally granted bid. If the auctioneer finds a better pre-bid than the definitive bid, then the auctioneer will send the *1* bidder associated with the definitive bid a **provisional reject** at step 5 (proceeds to step 2 to send *1* updated bid), send *1* bidder associated with the current preferred pre-bid a **provisional grant** at step 3 (proceeds to step 4 to send *1* definitive bid) and sends the other $br - 2$ bidders associated with the rest of the updated pre-bids a **provisional reject** message at step 3 (proceed to step 2 to send $br - 2$ updated bids). The total number of messages communicated if a definitive bid is not confirm granted at step 5 is $1 + 1 + br - 2 = br$, which is equal to the number of provisional grant/rejects submitted in the initial negotiation (the first of the repeated negotiations). After this, the number of definitive bids and pre-bids submitted is br again, equivalent to the number of pre-bids and definitive bids submitted after the provisional grant/rejects in the initial negotiation (the first of the repeated negotiations). Therefore, if this process (provisional grants and rejects, then submission of definitive bids and updated pre-bids) repeats η times, CNP-ext would communicate $(br + br)\cdot\eta$ messages. At the η^{th} repeated negotiation, the definitive bid submitted at step 4 is better than all the updated pre-bids sent by bidders, and therefore, at step 5, the auctioneer sends the bidder associated with the definitive bid a **confirm grant**, and sends all the $br - 1$ bidders that has unsuitable pre-bids (at step 3) a **confirm reject** message. Hence, the

[9] With the PAP (see chapter 5), this is equivalent to withdrawing the bid, submitting an updated bid (which is the "different" definitive bid), provisionally granting the updated bid and having the provisional grant accepted (which commit the bidder to the bid in the PAP, and hence is equivalent to the bidder being committed to the definitive bid in CNP-ext). If the definitive bid is the same as the provisionally granted bid, then no speech acts are required in the PAP.

auctioneer sends *1* + *br* – *1* = *br* messages. The total communication required by CNP-ext to allocate a set of tasks is the sum of the number of messages communicated:

Eq 2: Total communication for CNP-ext = $\kappa + br + (2 \cdot br) \cdot \eta + br$

4.6.3 Memory

The best-case memory requirements, as in section 4.4.3, will be formulated. Refer to the CNP-ext specification in Figure 13. Assume a maximum of *κ* bidders and a maximum of *br* bids received per task announcement. At step 1, the auctioneer sends out a **task announcement** to each of the *κ* bidders. Therefore, the auctioneer and the *κ* bidders store *1* set of tasks. The protocol proceeds to step 2, where bidders submit *br* **pre-bids**, and hence the auctioneer stores *br* received bids and the bidders store their *1* submitted bid. Since we are not at the end of planning, we ignore the **confirm reject** speech act at step 3. The auctioneer **provisionally grants** its preferred bid and **provisionally rejects** the other bids. The auctioneer may delete its *br* – *1* provisionally rejected pre-bids, and thus has the *1* set of tasks and *1* provisionally granted pre-bid stored. The bidders that have their *br* – *1* pre-bids provisionally rejected may delete their pre-bids, but do not delete their *1* set of tasks because they proceed to step 2 and send *br* – *1* updated **pre-bids**. The bidder associated with the bid that is provisionally granted, will send a **definitive bid**, and the protocol proceeds to step 5. Therefore, at this stage, the auctioneer has *1* set of tasks and *br* + *1* bids stored, and the bidders have, in total together, *κ* set of tasks and *br* bids (the provisionally granted pre-bid is deleted and replaced with the definitive bid). The auctioneer may compare the provisionally granted pre-bid and the definitive bid before deleting the pre-bid, because in the CNP-ext specification, if the definitive bid is much worse than the pre-bid, the auctioneer may **confirm reject** the definitive bid at step 5 (we assume this does not occur). There are two possibilities: (i) the auctioneer finds a better pre-bid than the definitive bid, or (ii) the definitive bid is still preferred over all pre-bids.

Case (i): The auctioneer will send the bidder associated with the definitive bid a **provisional reject** at step 5, which the auctioneer and bidder will delete the *1* definitive bid. CNP-ext proceeds to step 2 for the bidder where the bidder submits *1* updated pre-bid to replace the provisionally rejected bid, and thus the auctioneer and bidder each store the *1* updated pre-bid. The bidder associated with the current preferred pre-bid receives a **provisional grant** at step 3, and proceeds to step 4 to send a definitive bid. Therefore, the auctioneer will store the *1* definitive bid, as well as the *1* pre-bid that was provisionally granted (as above), and the bidder will replace its pre-bid with it definitive bid, therefore storing 1 bid. The other *br - 2* bidders associated with the rest of the updated pre-bids receive a **provisional reject** message at step 3, and proceed to step 2 to send *br - 2* updated bids. Again, the auctioneer has *1* set of tasks and *br + 1* bid stored, and the bidders have *1* set of tasks and *1* bid stored, and the protocol is back to case (i) or (ii). Therefore, it doesn't matter how many times the auctioneer provisionally rejects the definitive bid at step 5, the auctioneer and bidders will require the same amount of memory.

Case (ii): The auctioneer sends the bidder associated with the definitive bid (at step 5) a **confirm grant**, and sends all the *br - 1* bidders that has unsuitable pre-bids (at step 3) a **confirm reject** message, and the bidders delete their bids and it associated sub-task. The protocol completes. No extra bids or sub-tasks are stored.

Therefore, the maximum memory is as above – the auctioneer has a maximum of *1* set of tasks and *br + 1* bid stored, and the bidders have together, in total, *1* set of tasks and *1* bid stored.

4.7 Extended Contract Net Protocol

4.7.1 Limitations of CNP and CNP-ext to Fischer's Transportation Problem

Fischer found that CNP was insufficient for his transportation problem (Fischer and Kuhn 1993). The quantity of resources that a MA requires to be transported may be greater than any one TA's capacity. This results in no TA submitting bids, even though collectively they have the capacity to transport the quantity of resources (if the resources can be divided among the TA). An example is illustrated in Figure 14. The MA requires 50 tonnes of fuel transported, but the TA can only transport 40 tonnes, 20 tonnes and 10 tonnes respectively. The MA is able to decompose the transportation task into a set of transportation tasks with smaller quantities such that the TA are able to bid for the tasks. In order to find a suitable decomposition, the MA would require details of the type of TA that are available and their available services – i.e. knowledge of the capabilities of the agents in the society. This information may not be available (agents may not be willing to release this information), may require extensive communication, or due to the dynamic nature of the domain (agents come and go and their services are continually changing), will be difficult to keep up to date. CNP-ext also suffers from the same problem as CNP because bids from the TA (or bidders) must completely achieve the announced task.

In order to overcome the shortfalls of CNP and CNP-ext for Fischer's transportation domain, Fischer proposed an extension to the contract net protocol, called the *Extended Contract Net Protocol (ECNP)*. ECNP does not require the MA to decompose the transportation task *a priori* or possess knowledge of other agent's capabilities during the task allocation process.

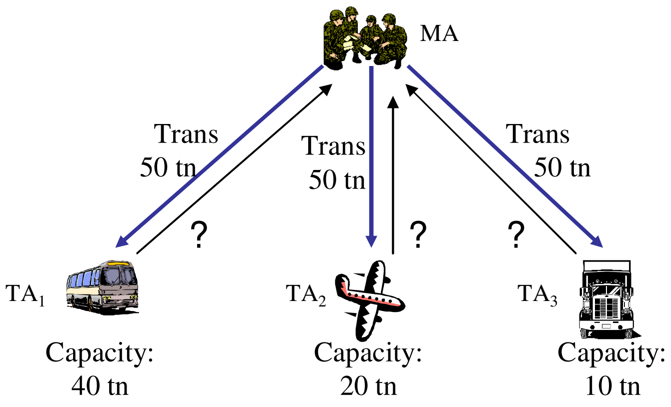


Figure 14. Shortfall with CNP – the quantity of resources that MA requires to be transported is greater than any one TA's capacity.

4.7.2 Extended Contract Net Protocol

Fischer's ECNP specification for his transportation domain is shown in Figure 15. There are two primary extensions to the original CNP. First, ECNP allows *partial bids*, where bids do not necessarily have to achieve the complete task that is announced. Therefore, if a MA (termed *shipping company* in Figure 15) announces a transportation task where the quantity to be transported is greater than the capacity of any single TA (termed *truck* in

Figure 15), then the TA are able to send a bid with the maximum capacity that they are able to perform – a *partial quantity bid*. After the MA selects a partial quantity bid, the MA will re-announce the remaining quantity of the transportation task that is left to achieve, and this process continues until the complete transportation task is achieved.

The second extension to CNP is that the **grant** and **reject** speech acts are split into four speech acts – **temporal grant**, **temporal reject**, **definitive grant** and **definitive reject**. Once a bid is received in ECNP, the MA may reject the bid using **temporal reject**, or grant the bid *temporarily* using **temporal grant**, in which case the TA modify their local plan incorporating the bid. Once a collection of partial bids are (or one single bid is) found that can completely achieve the transportation task, then the MA decides whether the allocation is suitable. If the allocation is suitable, then the MA sends all the TA with a temporal granted bid a **definitive grant**. If the allocation is not suitable, then the MA sends the TA with temporal grants a **definitive reject**, in which case the TA are no longer committed to their bids and they revert to their original plan without the bid. In Fischer's specification, an external agent determines whether an allocation is suitable, as shown by the "send a bid" and "wait for Grant/Reject" steps in Figure 15 (a). We do not make assumptions where this decision is made, whether it is the MA itself or an external agent, and thus reduce these two steps to "is allocation suitable". Additionally, the transportation task is received by the MA from an external agent – see "receive order" step in Figure 15 (a). We do not make any assumptions regarding how the MA's transportation task came about.

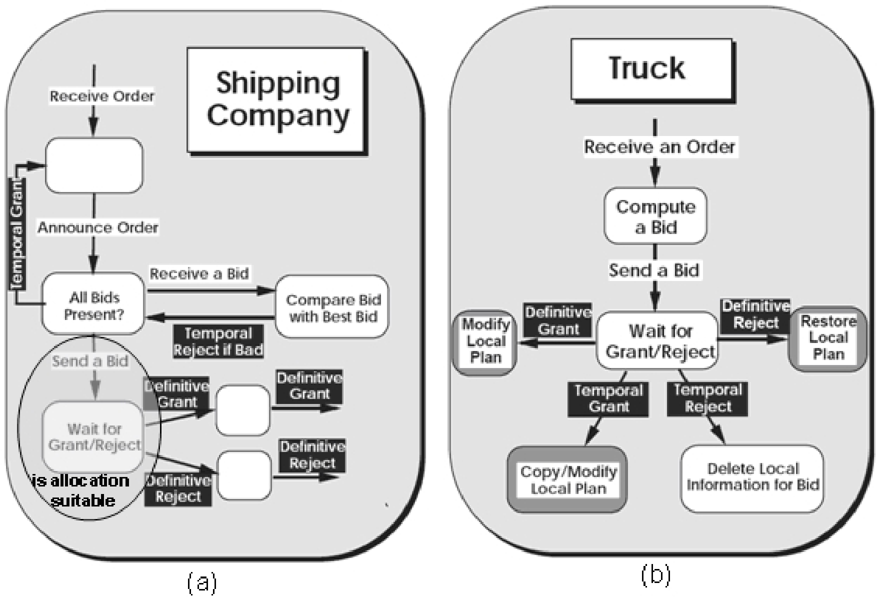


Figure 15. ECNP specification for the (a) MA (shipping company) and (b) TA (truck), obtained from (Fischer and Kuhn 1993).

Figure 16 shows ECNP in use for Fischer's transportation problem: (a) the MA announces a task to transport 50 tonnes of some resource, and the TA respond with bids that are the maximum capacity that they can transport, which is 40 tonnes, 20 tonnes and 10 tonnes respectively; (b) the MA prefers the 40 tonnes bid and sends that agent a temporal grant for the bid; (c) the remaining task left to achieve is to transport 10 tonnes of resources, and hence the MA announces this *new task*, and receives two bids from the TA to transport 10 tonnes of resources each; (d) the MA selects the most preferred bid and sends the associated TA a temporal grant for the bid; (e) the complete (transportation) task has now been achieved and the MA finds that the final allocation is suitable, and therefore sends the two agents that have a bid with a temporal grant a definitive grant.

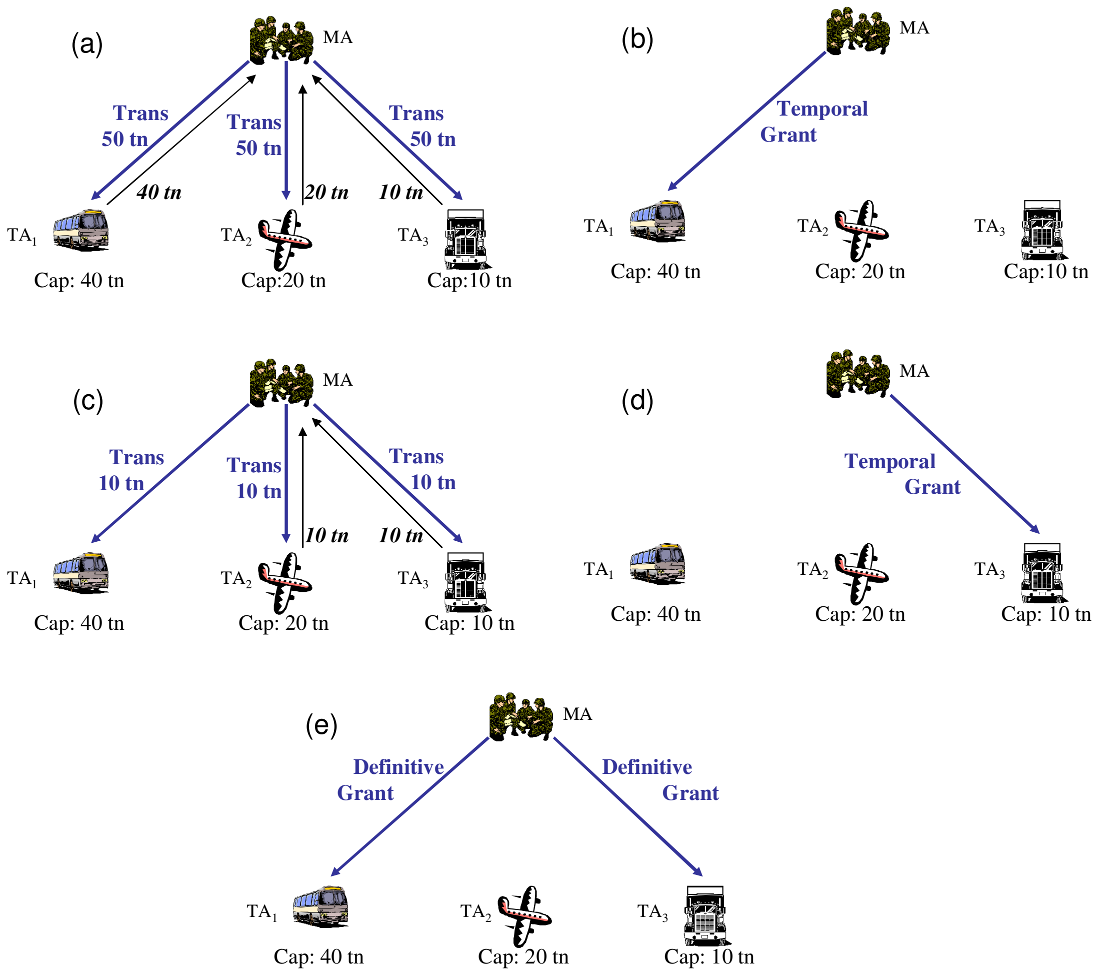


Figure 16. Example of ECNP in use. trans = transport; tn = tonnes; cap = capacity.

ECNP does not require the MA to decompose the task *a priori* in order to match the announced tasks with the services (or capabilities) of agents in the society. Task decomposition (plan) for the overall task is obtained during the planning process by extracting and assembling the relevant services that are available by the agents (TA) at the time for the particular task at hand. Therefore, the MA do not need to possess knowledge of other agents' capabilities *a priori* in order to find a plan and allocation of its task(s).

4.7.3 Simulated Trading

Fischer uses Simulated Trading (ST) in addition to ECNP to improve the solution obtained by ECNP. ST uses a market mechanism (stock market, double auction) to enable TA to exchange (*reallocate*) allocated transportation tasks with other TA if the exchange is beneficial (saves money). Our focus in the thesis is on the initial allocation of tasks from MA to TA, i.e. the type of interaction that ECNP facilitates, and not the reallocation of tasks between TA. In section 6.6, we discuss the unsuitability of double auction type mechanisms, which ST uses, for the initial allocation of tasks for our domain.

4.7.4 ECNP Facilitates Decentralised Greedy Search with Limited Backtracking

In general, ECNP facilitates a decentralised greedy search (or means-ends-analysis planning (Newell and Simon 1972)) with limited backtracking – the *complete* final plan can be abandoned if deemed unsuitable. Refer to Figure 17. ECNP commences with the MA's (or the auctioneer's) initial task announcement, which is t_0 in Figure 17. This is the overall goal that the MA would like to achieve, and corresponds to the root node of the search tree. The bids sent by the TA (or bidders) are the options the MA has to achieve, or move it closer to, its goal, and hence correspond to the branches of the search tree. In Figure 17, the bids for t_0 are branches b_1 and b_2. The planning is decentralised because the TA makes the decision on how (and how much) they will achieve the task at hand, if at all, and they present this service (bid) to the MA for consideration. The MA does not have any control over the TAs' decisions, and the TA do not need to provide the MA any further information regarding their available services or any other private information.

By granting a bid, the MA traverses one of the branches of the search tree, selecting one of the options it has available to fully or partially achieve its goal. In the example in Figure 17, the MA decides to (temporally) grant b_2. If the bid only partially achieves the goal, then the remaining component of the task (goal) that was not achieved by the bid is the new task to be announced by the MA. In the example in Figure 17, the new task is t_1, which is the difference between the task t_0 and the component of the task that bid b_2 achieves. Again, bids are received (b_3 and b_4) for the announced task (t_1). This process continues – the MA greedily selects (grants) the best bid for the current task and proceeds to achieve the remaining portion of the task that is not achieved by the bid (announces the new task), until the complete task is achieved. In Figure 17, the selection of bid b_5 results in the complete goal being achieved as the difference between the task t_2 and its selected bid b_5 is *null* – '∅'.

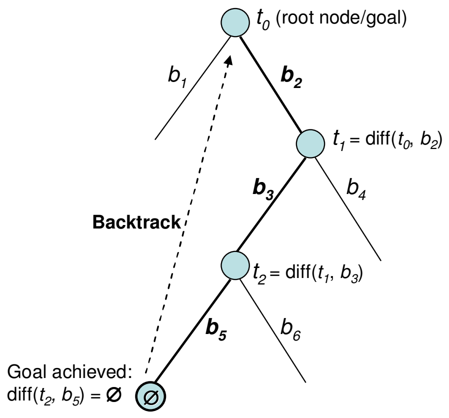


Figure 17. Search tree representing ECNP decentralised greedy search with limited backtracking – tasks t in ECNP are the nodes and bids b in ECNP are the branches of the search tree. diff(t, b) is a function that returns a task which is the difference between the task t and the portion of the task that the bid b achieves.

The branches (bids) along the path in the search tree from the root node to the null node, which have been selected (granted) by the MA, is the plan to achieve the MA's initial goal (task). In Figure 17, the final plan is b_2, b_3 and b_5 to achieve t_0. ECNP allows limited backtracking because it enables the MA to abandon the complete plan if it is unsuitable (by sending the TA with temporally granted bids a definitive reject message), but does not allow the MA to backtrack individual bids/branches in the search tree. If the plan is suitable, the MA may definitively grant the bids in the plan to secure the plan and allocate the tasks (the tasks that the granted bids are to achieve) to the relevant TA.

4.7.5 ECNP Using the Protocol Flow Diagram Representation

Figure 18 illustrates ECNP using our Protocol Flow Diagram representation. Note that we use the terms *provisional* and *confirm* rather than the terms *temporal* and *definitive*, respectively. There are three new symbols used to represent ECNP, which are "+", "-" and "(x)". The symbol "(x)" is used when there is greater than one option for the next step of the protocol if a particular speech act is used. Therefore, in Figure 18, if the **provisional grant** speech act is used in step (3), then the agents may take option (a) and proceed to step (1), or option (b) and proceed to step (4). The option taken may depend on certain conditions, and may be dictated by a particular agent. In ECNP, after a **provisional grant**, the *MA* takes option (e), i.e. goes back to the **task announcement**, if the bid *does not* completely achieve the task, or takes option (f), i.e. may **confirm grant** or **confirm reject** the final plan (provisionally granted bids), if the bid *does* completely achieve the task.

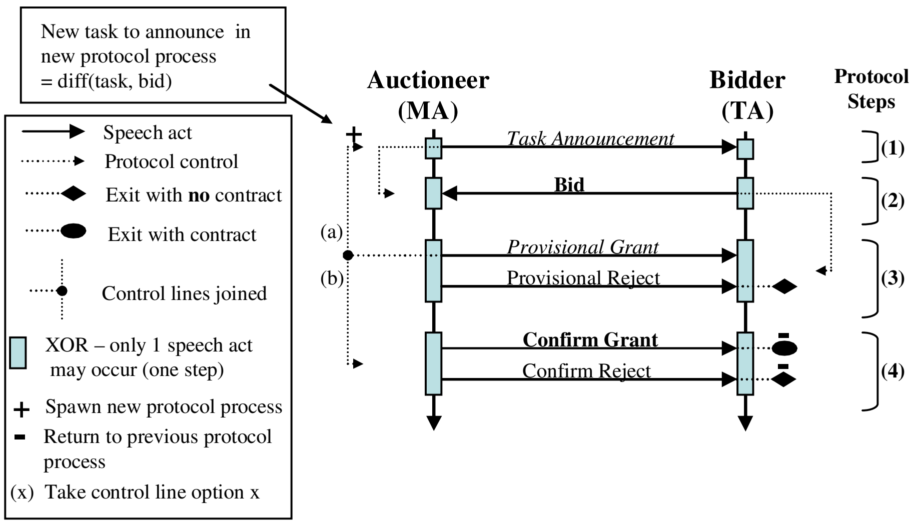

New task to announce in new protocol process = diff(task, bid)
Speech act
Protocol control
Exit with **no** contract
Exit with contract
Control lines joined
XOR – only 1 speech act may occur (one step)
+ Spawn new protocol process
- Return to previous protocol process
(x) Take control line option x
Auctioneer (MA)
Bidder (TA)
Protocol Steps
Task Announcement
Bid
Provisional Grant
Provisional Reject
Confirm Grant
Confirm Reject
(a)
(b)
(1)
(2)
(3)
(4)

Figure 18. ECNP using the Protocol Flow Diagram representation. diff(task, bid) is a function that returns a task which is the difference between the task and the portion of the task that the bid achieves.

The illustration in Figure 19 will be used to explain the use of symbols "+" and "-", and the concept of a protocol process. The protocol representation in Figure 19 is equivalent to that in Figure 18, but without the "+" and "-" symbols, and hence, graphically illustrates the symbols purpose – to describe the control of the protocol execution between protocol processes. A single protocol process is the execution of the protocol for a single task, and thus contains its associated state variables (such as bids sent/received for the task) and the position in the protocol. Therefore in Figure 19, the initial protocol process corresponds to achieving task t_0, and its n bids (say $b^0_1, \ldots, b^0_n$), and the protocol process completes when the bids are (provisionally or confirm) granted or rejected for t_0. In the initial protocol process, if the provisionally granted bid does not achieve the complete task, then a new protocol process is spawned (see Figure 19, option (a) in t_0's protocol process) for task t_1, which is the portion of t_0 that the provisionally granted bid for t_0 does not achieve. The protocol process for t_1 has m new bids associated with it (say $b^2_1, \ldots, b^2_m$). The initial protocol process is halted until the spawned (*next*) protocol process is complete (confirm grant or confirm reject a bid for t_1). When this occurs, the initial protocol process continues at step (4), shown by the control lines from step (4) of the spawned protocol process to step (4) of the initial protocol process in Figure 19, which is to confirm grant or confirm reject the provisionally granted bid for t_0. Therefore, the symbols "+" and "-" in Figure 18 indicates that control should proceed to the next or previous protocol process, respectively, and the control line associated with the symbols indicate which step in the next or previous protocol process execution should commence.

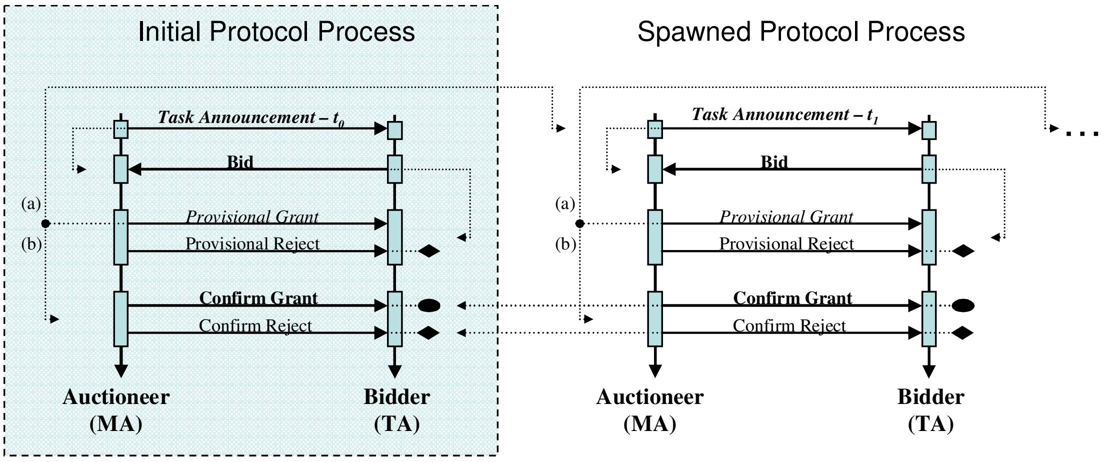


Figure 19. Representation of ECNP without the"+" and "-" symbols.

4.8 ECNP Analysis

4.8.1 ECNP Protocol Description

We will define ECNP using the negotiation (or protocol process) based on the description presented in section 4.1, and the Protocol Flow Diagram illustration in Figure 18. Since the description of ECNP is complicated with new protocol processes spawned due to the partial fulfilment of its set of tasks $\Omega \subseteq T$, multiple negotiations between an auctioneer and bidder may be required to achieve T. We use the subscript x to indicate the x^{th} protocol process, and hence x^{th} negotiation, between auctioneer α and bidder γ in order to achieve T. We have a negotiation defined as

$$Neg_x = < \alpha, \gamma, \Omega_x, \Sigma_x, S_x, A, \delta_x, s_x, F_x >$$

where α is an auctioneer in the negotiation, and γ is a bidder in the negotiation. Other tuples are dependent on the protocol:

- $\Omega_x \subseteq T$, is the set of tasks that the agents α and γ are negotiating over in order to fully or partially achieve it, and $\Omega_{x+1} \subseteq \Omega_x \subseteq \ldots \subseteq \Omega_0 = T$. $x = 0$ denotes the negotiation corresponding to the initial (ECNP) protocol process to achieve T. $x + 1$ denotes the negotiation (Neg_{x+1}) corresponding to the protocol process to achieve Ω_{x+1}, spawned from the x^{th} protocol process negotiation (Neg_x) due to the *partial* achievement of Ω_x.

- Σ_x is the set of bids that the agents α and γ are negotiating over to fully or partially achieve the set of tasks Ω_x.

- $S_x = \{\text{step1}_x, \text{step2}_x, \text{step3}_x, \text{step4}_x, \text{step1}_{x+1}, \text{step4}_{x-1}\} \cup \{\text{exit}^{NC}_x, \text{exit}^{C}_x\}$ is the set of possible states that are accessible by the protocol in the x^{th} protocol process. exit^{NC}_x is the state which exits with *no* contract and exit^{C}_x is the state which exits with a contract.

- $A = A_\alpha \cup A_\gamma \cup A_e$ is the set of possible speech acts or events where: A_α is the set of speech acts associated with the auctioneer; A_γ is the set of possible speech acts associated with the bidder; and A_e is the set of events (common to both agents).

A_α = ({**task announcement**}$\times$ Ω) ∪ ({**provisional grant**}$\times$ Σ) ∪ ({**provisional reject**}$\times$ Σ) ∪ ({**confirm reject**}$\times$ Σ) ∪ ({**confirm grant**}$\times$ Σ); A_γ= ({**bid**}$\times$ Σ); $A_e = \emptyset$.

- $\delta_x \subseteq (S_x \times A \times S)$ specifies the transitions between the possible states in the xth protocol process, such that δ is a function, with δ(step1$_x$, < **task announcement**, ω>) = step2$_x$; δ(step2$_x$, < **bid**, σ>) = step3$_x$; δ(step3$_x$, < **provisional grant**, σ>) = step1$_{x+1}$; δ(step3$_x$, < **provisional grant**, σ>) = step4$_x$; δ(step3$_x$, < **provisional reject**, σ>) = exit$^{NC}_x$; δ(step4$_x$, < **confirm grant**, σ>) = exit$^{C}_x$ & step4$_{x-1}$ (*see below*); δ(step4$_0$, < **confirm grant**, σ>) = exit$^{C}_0$; δ(step4$_x$, < **confirm reject**, σ>) = exit$^{NC}_x$ & step4$_{x-1}$ (*see below*); δ(step4$_0$, < **confirm reject**, σ>) = exit$^{NC}_0$.
- s_x = step1$_x$, is the start state for the xth protocol process.
- F_x = {exit$^{NC}_x$, exit$^{C}_x$} is the set of final states for the xth protocol process.

(exit$^{C}_x$ & step4$_{x-1}$) and (exit$^{NC}_x$ & step4$_{x-1}$) asserts that the participating agents exit the current (xth) protocol process with and without a contract, respectively, and then proceeds to state step4 of the previous (x-1th) protocol process.

4.8.2 Communication

The best-case communication requirements, as in section 4.4.2, will be formulated. Refer to the ECNP specification in Figure 18. Assume a maximum of κ bidders, a maximum of *br* bids received per task announcement, and *m* is the depth of the search (number of bids to achieve the initial set of tasks). At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, therefore communicates κ messages. At step 2, the bidders submit (communicates) *br* **bids**. The protocol proceeds to step 3. The auctioneer **provisionally grants** its most preferred bid, and **provisionally rejects** all the other bids, and thus communicates *br* messages. If the selected bid does not achieve the auctioneer's complete set of tasks, then ECNP proceeds to step 1 (control option (a)) to announce the remaining set of tasks that the bid did not achieve, and the same process is repeated until the *m*th bid is provisionally granted at step 3 which does fully achieve the set of tasks announced by the auctioneer. Therefore, ECNP has performed step 1 to step 3 *m* times, and thus the total communication at this point is $(\kappa + br + br)\cdot m$. The protocol proceeds to step 4 (take control option (b)) where the auctioneer will either send all *m* provisionally granted bids a **confirm grant** (if the plan is suitable) or a **confirm reject** (if the plan is unsuitable), and thus communicates *m* messages. The total communication required by ECNP to plan and allocate tasks is the sum of the number of messages communicated:

Eq 3: Total communication for ECNP = $(\kappa + 2\cdot br)\cdot m + m$

4.8.3 Memory

The best-case memory requirements, as in section 4.4.3, will be formulated. Refer to the ECNP specification in Figure 18. Assume *br* is the maximum number of bids submitted for each announced set of tasks, κ are the number of bidders that submit bids and *m* is the depth of the search (number of bids required to achieve the auctioneer's initial set of tasks). At step 1, the auctioneer **announces** its set of tasks to each of the κ bidders. Therefore, the auctioneer and the κ bidders store the *1* set of tasks. At step 2, the bidders submit *br* **bids**, and hence the auctioneer stores *br* received bids and the bidders store their *1* submitted bid. The protocol proceeds to step 3. The auctioneer **provisionally grants** its most preferred bid, and **provisionally rejects** all the other bids. Therefore, the auctioneer and bidders may delete the rejected bids and its associated task. The auctioneer has stored a maximum of *1* set of tasks and *br* bids, and the κ bidders have stored *together* a maximum of κ set of tasks and κ bids, and only the bidder that had its bid provisionally granted still has the set of tasks and its bid stored. If the selected bid does not achieve the auctioneer's complete set of tasks, then ECNP proceeds to step 1 (control option (a)) to announce the remaining set of tasks that the bid did not achieve, and the same process is repeated. For the next announced set of tasks, again the auctioneer will send its set of tasks, receive *br* bids, and therefore the auctioneer will have *2* sets of tasks and *br+1* bids in memory. The bidders will have stored the new sets of tasks and its bids, in addition to the previously granted bid and its associated set of tasks. Therefore, the bidders have stored together $\kappa + 1$ set of tasks and $\kappa + 1$ bids. When bids are provisionally granted/rejected, the auctioneer deleted the rejected bids and the bidders associated with the rejected bids will delete their set of tasks and its bid. Again, if the bid does not fully achieve the set of tasks, the process repeats, until the m^{th} bid is provisionally granted at step 3 which does fully achieve the set of tasks. The auctioneer would have a maximum of *m* sets of tasks and *br+m-1* bids stored in memory (just before the provisional reject deleted *br – 1* bids for the final set of tasks). The total maximum memory required for all the κ bidders together is $\kappa + m - 1$ sets of tasks and $\kappa + m - 1$ bids. After the provisional reject, the auctioneer deletes the *br – 1* rejected bids, and all but one bidder delete their set of tasks and its bid. The protocol proceeds to step 4 (take control option (b)) where the auctioneer will either send all *m* provisionally granted bids a **confirm grant** (if the plan is suitable) or a **confirm reject** (if the plan is unsuitable). Therefore, the total maximum memory required for the auctioneer is m sets of tasks and $br + m - 1$ bids, and the total maximum memory requirements for all the κ bidders together is $\kappa + m - 1$ sets of tasks and $\kappa + m - 1$ bids.

4.9 ECNP Shortfalls for Our Transportation and General Planning Problems

There are a few shortfalls with ECNP for our particular transportation problem (Perugini, Lambert et al. 2003), and hence to our general planning problem. We require a protocol that can facilitate agent planning and task allocation in a decentralised, dynamic and open environment, within a many-to-many setting (open-market). By many-to-many setting, or open-market, we mean that there are many agents, the MA/auctioneers, acquiring services (or in the reverse case, could be providing goods or services) from many services providing (or requesting) agents, the TA/bidders, and many TA providing services to (or requesting goods and services from) many MA, *simultaneously* (see Figure 20). We require a protocol that allows the MA to find the suitable plan and allocation of its task(s) (i.e. goods or services provided or requested) to TA in this complex environment.

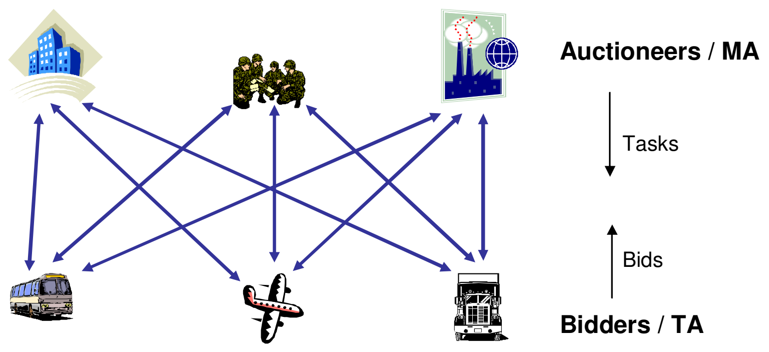


Figure 20. The complex and dynamic many-to-many setting/interaction.

The many-to-many setting, in addition to the dynamic (organisations) and open nature of the environment, increases the complexity and dynamics of the interaction required for planning and task allocation. From the perspective of the TA, MA tasks (or business goals) are continually changing, i.e. new tasks emerge and old tasks are withdrawn, for three reasons. First, due to the dynamics of organisations, their business goals may change. Second, due to the open environment, agents may enter or leave the system at any time, and thus bring new tasks and withdraw old tasks, respectively. Finally, an agent may acquire a service (bid) which partially achieves a task, and thus creates a new task to be achieved. If newly created task cannot be achieved, then the corresponding bids and tasks must be withdrawn.

From the perspective of the MA, TA bids (services or capabilities) are continually changing, i.e. new bid emerge and old bids are withdrawn, even during the MA's planning process (of finding suitable bids to achieve their tasks), for three reasons. First, due to dynamics of organisations, they may acquire new, and dispose of old, capabilities, which therefore, change the services (bids) that they can provide. Second, due to the dynamic nature of the environment, agents may enter or leave the system at any time, and thus bring new bids and withdraw old bids, respectively. Finally, due to the many-to-many setting, a bid that was available to a MA could be acquired by another MA, and thus the bid is no longer available for the MA. Alternatively, a bid acquired by a MA could no longer be required and released, and therefore, the bid is now available to other MA. In addition to the complexity of the dynamics, agents must manage many interactions from many agents simultaneously.

The applications we required a protocol to facilitate planning for are of greater complexity than Fischer's transportation problem. The military transportation requirements for MALT are global; hence we term the problem *global transportation scheduling*. We require a large quantity of resources is to be transported on a global scale – between countries, states and cities. Thus, a single transport asset is unable to transport the full quantity of resources the complete distance, or *route*. This results in TA submitting partial quantity *and* partial route bids for a transportation task, compared with just partial quantity (full route) bids with Fischer's transportation problem. This increases the complexity because we have two dimensions (quantity and route) of partiality to consider. Time constraints associated with the transportation task and the route are dependent on each other, adding a third temporal dimension, which also results in time-space issues that need to be resolved during planning. For example, consider the task of transporting fuel from Melbourne to Sydney. If a TA bids to move the fuel from Canberra to Sydney starting at time *t*, and the bid is granted, then there may not be a single TA, or a collection of TA, that can transport the fuel from Melbourne to Canberra by time *t*, or at all. The greater complexity (and search space) results in a greater likelihood of an infeasible or bad solution being encountered in planning. Therefore, we require a more flexible planning protocol for these complex problems than the greedy planning with limited backtracking provided by ECNP.

In light of the complexity and dynamics of our agent domain described above, three shortfalls with ECNP for our planning requirements are:

- *Backtracking* – Due to the increased complexity of the applications, and thus likelihood that an infeasible (or bad) solution may be encountered, backtracking is required. The protocol needs to be able to reject a granted bid that results in an infeasible (or bad) solution, and reselect a bid for the associated task to replace the rejected bid. ECNP's specification does not accommodate backtracking because it does not allow the MA to reject *individual* bids that were previously granted.

- *Multiple MA/auctioneers and the eager bidder problem* – ECNP (and CNP) suffers from the eager bidder problem (Schillo, Kray et al. 2002), which could limit the TA's ability to bid for more than one MA simultaneously. When TA bid for a task, they are committed to the bid even though the bid has not been granted, and hence cannot make the same or a conflicting bid to other MA. Having to wait for negotiations to complete with one MA in order to make a bid to another can result in the TA losing potential bid allocations (contracts) with other MA.

- *Dynamic bids and tasks* – ECNP does not consider dynamic bids and tasks. Bids and tasks that are sent may no longer be available, and are therefore withdrawn, when agents may attempt to use them. New bids by TA could emerge for a MA during its planning, and thus the TA should be able to send updated bids. The ECNP specification does not allow for withdrawn bids and tasks, and updated bids.

In addition to these shortfalls, a protocol for decentralised planning could involve extensive communication, particularly in a large system (many agents) for a large planning problem. Therefore, minimising the amount of communication required in the protocol that facilitates decentralised planning is another primary concern. To follow, we present the Provisional Agreements Protocol (PAP), which is an extension of ECNP, developed to overcome the shortfalls of ECNP to our planning (and hence, global transportation scheduling) problem[10]. Note that details regarding our global transportation problem and it implementation are discussed in chapter 7.

[10] Note that ECNP was not desiged for multiple auctioneers or the combinatorial auction application we use to evaluate PAP (Chapter 6).

Chapter 5

5 Provisional Agreement Protocol

This chapter presents the Provisional Agreement Protocol (PAP) (Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2005). PAP is an agent protocol that facilitates planning and task allocation in a decentralised, dynamic and open environment within a many-to-many setting and involves contracting, as required by the modern military logistics domain. PAP allows agents to acquire and assemble services from other agents, within the complexity and dynamics of an open market, in order to achieve their logistics goals. PAP is an extension of Fischer's Extended Contract Net Protocol (ECNP), overcoming its shortfalls for our transportation and planning domains. Formal analysis of PAP is presented. We show that PAP has greater flexibility with planning (performs a decentralised depth-first search) and reduced communication (under certain circumstances) over CNP, ECNP and CNP-ext presented in the previous chapter. PAP does not fall into livelock or deadlock.

5.1 Protocol Specification

5.1.1 Protocol Policies

Agents in PAP implement commitment, persistence and bidding policies.

Commitment Policy:

The *commitment policy* states that bidders are not committed to submitted bids unless they are *provisionally granted*, and the grant is accepted. This allows the bidders to bid many conflicting bids to different auctions concurrently, and prevents the eager bidder problem of waiting for rejection of a bid sent to one agent before submitting a bid elsewhere. Therefore, auctioneers do not need to reject bids, but must act quickly in order to secure the bids – first in first served. Auctioneers are only committed to a bid when they send a *confirm grant* for it, and are therefore able to *reject a provisionally granted bid*, in which case the bidder is no longer committed to the bid.

Persistence Policy:

PAP allows backtracking, and thus agents store received bids and task announcements for future use. Since bids and tasks may no longer be available at a later time when agents decide to use them, the *persistence policy* states that agents' tasks and bids are considered persistent unless they reveal otherwise, and therefore communication is not required for unavailable bids and tasks. Hence, agents will only discover that tasks and bids are unavailable, or *(provisionally) withdrawn*, after they try to bid for it, or grant it, respectively. This allows MA to not have to reject bids that they do not intend to use, allowing bids to remain available later if required during backtracking.

Bidding Policy:

There are a few components to the *bidding policy.* A bidder may bid for a task *anytime* that it believes the task is available – even after the deadline as the task, and hence its bids, may be revisited during backtracking. Only *one* bid may be submitted for a task per bidder, which is the bid that the bidder believes the auctioneer most prefers, and that can *fully or partially* achieve the task. Bidders can determine which of their bids the auctioneer prefers because the auctioneer supplies bidders with a *bid evaluation function* with the task announcement, which is a function the auctioneer will use to evaluate the bids for the task. Bidders send an *updated bid*, if possible, if their worst submitted bid is either rejected or attempted to be provisionally granted but is (provisionally) withdrawn, to replace the unwanted or unavailable bid, respectively. An *updated better bid* is sent when new opportunities arise and a bid becomes available that is better than the worst submitted bid for a task. The auctioneer should therefore have, at any time, all the bidders' best bids through to its worst submitted bid for a task (if it's in the bidder's interest to do so). Bid replacements are only sent when the worst bid is rejected or withdrawn because if the auctioneer is not interested in the bidder's worst submitted bid, then it would not be interested in an updated bid that is the same or worse.

5.1.2 The Protocol

Figure 21 illustrates PAP using our Protocol Flow Diagram representation, and Table 1 presents the meaning of the speech acts and events in the protocol.

Table 1. Meaning of the speech acts and events used in PAP between the auctioneer (or MA) and the bidder (or TA).

Speech Act or Event	**Meaning**
Task Announcement	The auctioneer requests bids (from bidders) to partially or fully achieve a task.
Bid	The bidder sends its *desire* (a bid) to fully or partially achieve the task, but is not committed to the bid.
No Communication	There is no communication between the auctioneer and bidder at the particular step in the protocol.
Provisional Grant	The auctioneer would like to *provisionally* accept a bid for the partial or full execution of the task (so will not be committed to the bid, and thus may *reject* the bid later if it is not suitable), and therefore asks the bidder to accept the grant and commit to the bid (i.e. the auctioneer asks the bidder to formally offer the bid with commitment).
Provisional Grant Accepted	The bidder accepts the provisional grant (i.e. the bidder formally offers the bid with commitment) and is therefore committed to the bid. The auctioneer is not yet committed to the bid until the auctioneer accepts the contract (sends a confirm grant) for the bid, and can therefore reject the bid. In the acceptance, the bidder specifies some deadline by which the auctioneer must commit/grant or reject the bid, after which time the bidder may decommit.
Provisionally Withdrawn	The bidder's bid is not available at the moment (or does not currently desire to perform the bid), but may become available sometime in the future.
Withdrawn	From the bidder – the bidder's bid is no longer available (or does not desire to perform the bid) now or in the future. From the auctioneer – the task that the bidder submitted a bid for is no longer available (no longer needs to be achieved).
Provisional Reject	The auctioneer informs the bidder that its bid is not *currently* suitable for the task, and therefore may decide to grant the bid later if it discovers otherwise. If the bid is provisionally granted, then the bidder is no longer required to be committed to the bid.
Backtracking	The *current task* is the portion of the *previous task* that its partial bid did not achieve. The partial bid granted for the *previous task* that created the *current task* is not suitable as the

	current task cannot be achieved, or cannot be achieved well. The auctioneer will revert back to the previous task by rejecting the partial bid that was granted for the previous task, in an attempt to grant another (more suitable) bid to achieve the previous task.
Confirm Grant	The auctioneer confirm grants the bid, and hence formally accepts the bid to partially or fully achieve the task. The auctioneer and bidder are committed to the bid (have a contract) and cannot decommit (without penalty).

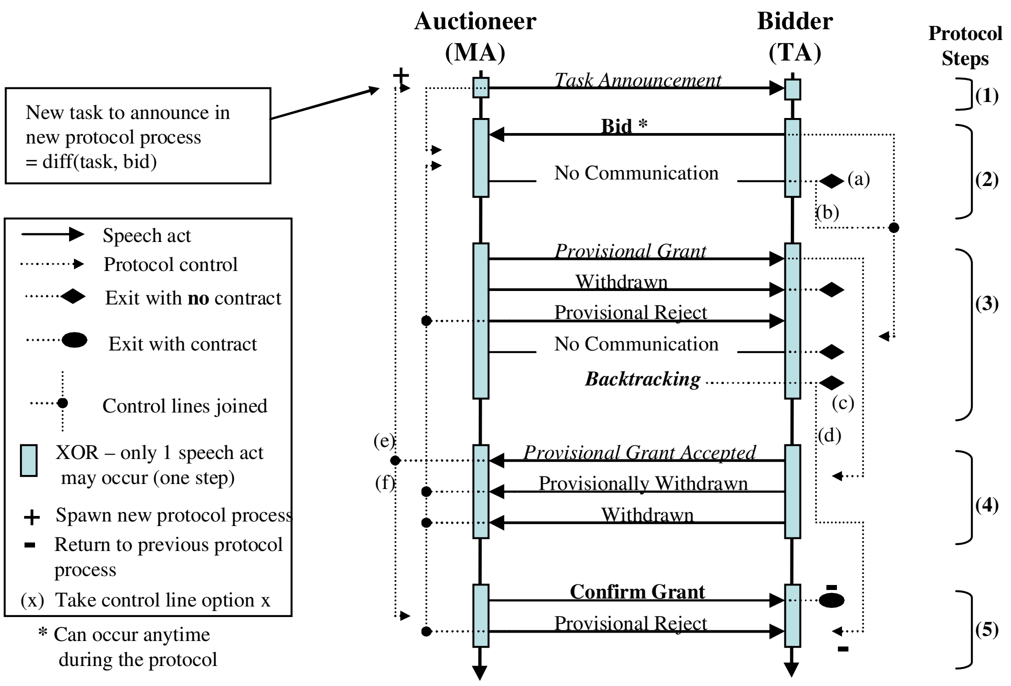


Figure 21. PAP using the Flow Diagram Representation. diff(task, bid) is a function that returns a task which is the difference between the task and the portion of the task that the bid achieves.

The protocol contains five steps, and operates as follows (see previous chapter for discussion on the concept of a protocol process):

Step 1: The auctioneer broadcasts a **Task Announcement** to all *suitable* bidders. The announcement contains a set of tasks T, a *bidding* deadline d which is the earliest time that the auctioneer may select a bid for the task, and bid evaluation function $f : \Sigma \rightarrow \mathbb{R}$ for the set of bids Σ. $f(b)$ informs bidders how the auctioneer will evaluate bid b, so bidders can submit their best bid for T. An example is $T = \{t_1,$

t_2, t_3, t_4, t_5}, where t_i are sub-tasks to be performed to achieve the overall task *T*, *d* = 5 seconds, *f(b)*= min[price(*b*) / number-of-sub-tasks(*b*)], i.e. want to minimise the price (*p*) paid per unit task. Control then proceeds to step 2. The initial task is *T*, which the initial protocol process attempts to achieve. If a partial bid is selected for task *T* (in the following protocol steps), then a new protocol process is spawned to achieve the remaining task T^+ that is not achieved by the partial bid, commencing at step 1.

Step 2: Bidders may submit *one* (best/next-best) **Bid** *b* that they believe can partially or fully achieve the task, or submit nothing – **No Communication**. The initial bid submitted for the task should be the bidder's best bid (determined by *f(b)*). If this step is arrived at from a following step (a submitted bid is rejected or withdrawn), then the *updated bid* submitted should be the bidder's next best bid. If no communication occurs, then the protocol either exits for the bidder for that task (path (a) in Figure 21) or proceeds to step 3 (path (b) in Figure 21) if the bidder still has bids submitted which have not been provisionally granted [11]. The auctioneer proceeds to step 3 if no bids are submitted (path (b) in Figure 21). Whether a bid is submitted or not, if an opportunity arises later for the bidder to submit an initial bid (if **no communication** previously) or a bid that is better than its worst submitted bid (*updated better bid)*, then bidder may submit the bid at anytime during the protocol. After a bid is submitted, control proceeds to step 3. An example of bids is: $b_1 = \{t_1, t_4\}$, $p_1 = \$10$, $f(b_1) = 10/2 = 5$; and $b_2 = \{t_2, t_4, t_5\}$, $p_2 = \$60$, $f(b_2) = 60/3 = 20$. Since $f(b_1) < f(b_2)$, b_1 is preferred by MA over b_2. Therefore, b_1 should be sent by the bidder.

Step 3: After the bidding deadline *d,* the best bid for the task *T*, based on *f(b),* is given a **Provisional Grant** and control proceeds to step 4. The provisional grant is supplied with a *provisional grant acceptance deadline* (*pgad*) by which the bidder must accept the provisional grant, otherwise the auctioneer may no longer

[11] This may occur if the agent arrives at this step from a following step (bid rejected or withdrawn), and therefore may have already submitted other bids not used by the auctioneer (e.g. updated better bids).

consider the bid (assume its withdrawn). If a submitted bid is unsuitable, the auctioneer sends a **Provisional Reject**, and the bidder proceeds back to step 2, allowing it to send an updated bid to replace the rejected bid. The reject is provisional so the auctioneer can grant it again later if it wants to – there is no benefit in "fully" rejecting the bid. Auctioneers implement their own strategy for dealing with provisionally rejected bids – when, or if, they should grant the bid again (see below). If a bid is sent for a task that is no longer available (e.g. already achieved), then the auctioneer sends a **Withdrawn** message, and the bidder can delete the task so it does not consider the task for any further bids. Bidders which do not have their bids granted, receive **No Communication** and exit the protocol after some length of time after which the bidder is confident that the bid will not be required. If no bids are received by the deadline, the auctioneer assumes no solution exists for the task, and thus requires **Backtracking** or can accept the current partial solution (see step 5). If this occurs with the initial (root) task, then no solution exists, so path (c) in Figure 21 is taken and the process exits. Otherwise, for task T, path (d) in Figure 21 is taken to step 5 of the previous protocol process associated with task T^- (the protocol process for T was created from a partial bid selection for task T^-). Even if bids are received, the auctioneer may backtrack if it is not satisfied with the current plan.

Step 4: The bidder may **Accept the Provisional Grant**, in which case it becomes committed to the bid, and cannot decommit without penalty. The bidder specifies a *confirm deadline cd* by which the auctioneer must confirm grant the bid, otherwise the bidder may de-commit. If the task T is completely achieved, path (f) in Figure 21 is taken to step 5. Otherwise, path (e) in Figure 21 is taken to step 1 (of a new spawned protocol process for T^+), and the new task T^+ to announce is the remaining portion of the task T not achieved by the bid. In our example with $T = \{t_1, t_2, t_3, t_4, t_5\}$, if $b_1 = \{t_1, t_4\}$ was selected the new task would be $T^+ = T \setminus b_1 = diff(T, b_1) = \{t_2, t_3, t_5\}$. The bidder may not accept the provisional grant, and send a **Provisionally Withdrawn** or **Withdrawn** message instead, or submit nothing by the *pgad* deadline. A provisionally withdrawn message informs the auctioneer that the bid is not available at the moment, but may become available later – e.g. conflicts with another bid that is provisionally granted, and hence may be rejected later, making the bid available again. A withdrawn (or no) message informs the auctioneer that the bid is not available now or in the future – e.g. conflicts with another bid that is confirm granted. Auctioneers may discard withdrawn bids. Control proceeds to step 2 to allow the bidder to submit an updated bid to replace the (provisionally) withdrawn bid. Auctioneers and bidders may implement their own strategy for dealing with provisionally withdrawn bids – when, or if, to grant or resubmit the bid again (see below).

Step 5: If control arrived from step 3 (from backtracking in the following protocol process for task T^+), the auctioneer has two choices for task T. It can accept the current solution, even if it has not fully achieved the task (which is allowed in the combinatorial auction application), and **Confirm Grant** all the provisionally granted bids (in this protocol process for T, and all previous protocol processes, for tasks T^-, T^{--}, etc.), which allocates and secures the bids (forms a contract). Otherwise, if the auctioneer believes the solution is not suitable, then it can *backtrack* by **Provisionally Rejecting** the bid for the task T, in which case the bidder is no longer committed to the bid. Control then proceeds to step 2 and 3, where the bidder that had its bid rejected may send an updated bid for T, and the auctioneer can then select a new bid for T[12]. If control arrived from step 4, then the task was completely achieved, so all the provisionally granted bids may be **Confirm Granted**. If the auctioneer's task T is completely achieved with provisionally granted bids but the solution is not suitable (a bad plan has resulted), the auctioneer may backtrack if it desires, as explained above. Once confirm granted, both the auctioneer and bidders are committed to the bids, and thus cannot decommit without penalty. If a bidder with a provisionally granted bid

[12] In our example, if backtracking task T^+ (e.g. because the auctioneer received no bids for T^+), then reject b_1 for T, where T^+ is associated with the *following* protocol process that backtracked and T is associated with the *current* protocol process. The bidder may then send an updated bid b_2 for T to replace its provisionally rejected bid b_1, and auctioneer may provisionally grant the new bid b_2 for T.

does not receive a confirm grant before deadline *cd*, it may de-commit from the bid.

In the current PAP implementations, agent strategies for dealing with provisionally rejected and provisionally withdrawn bids are to discard them, and hence not used them again. This is one of the conditions that ensure convergence, and results in PAP performing a decentralised depth first search (see section 5.3.4). Also, the provisional grant acceptance deadline *pgad* and confirm deadline *cd* are large enough not to affect planning, but small enough to ensure protocol robustness (see the following sections). Therefore, we can ignore *pgad* and *cd* in the implementations that are presented. The auctioneer waits for the deadline *d* at step 3 even during bid updates, i.e. after a bid is rejected or withdrawn, as the update may be better than the currently held bids.

We assume that agents may be cooperative or self-interested. Additionally, we assume that bidder agents bid their true valuations, and have no preference for having one submitted bid granted over another submitted bid. Their aim is to have their bids (services/requests) granted.

5.1.3 PAP Example in the Transportation Domain

The operation of PAP and its various features will be illustrated by applying it to the transportation problem of finding a route for the transport of resources (simplified for illustrative purposes). Note that since we are applying PAP to the transportation domain, we refer to the auctioneers as Manager Agents (MA) and bidders as Transport Agents (TA). The map of Australia in Figure 22 provides a reference for the locations used in the transportation example.

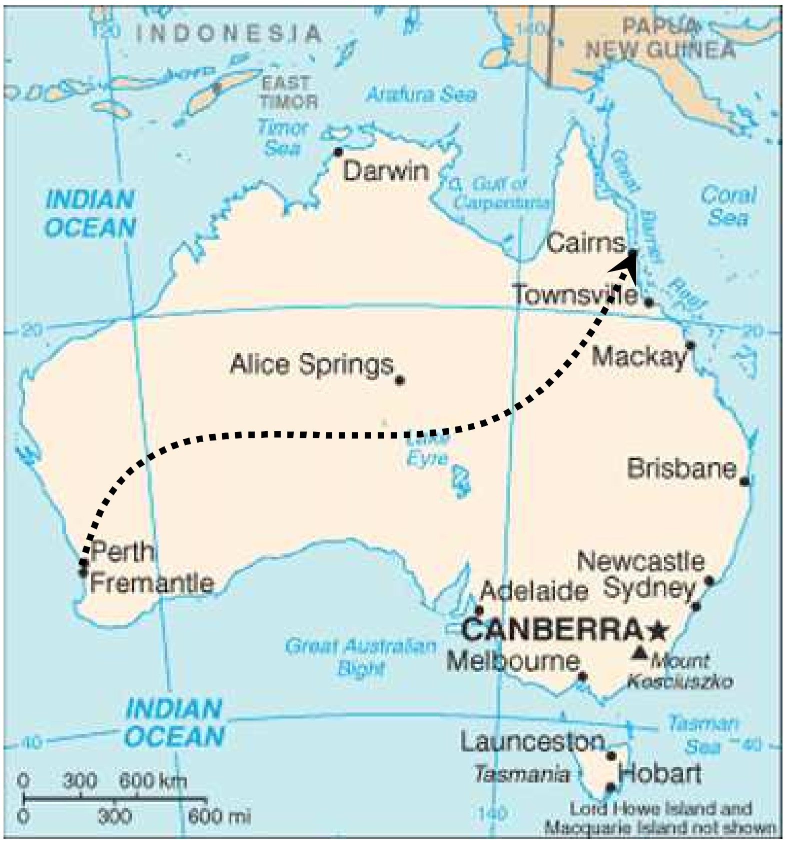


Figure 22. Map of Australia (obtained online from www.flagfocus.info).

The overall transportation task in this example is to *transport resources from Perth to Cairns* – represented as *T(Perth, Cairns)*. This is shown in Figure 23 (a), where a MA announces (**Task Announcement**) the task t_0 to the TA. We assume that the MA wants to minimise the evaluation function *f*, which is sent to the TA with the task announcement (not shown). In this illustrative example, we do not show how *f* is obtained, rather we just state the value of *f(b)* associated with each bid *b*, which we will call the *f-value*, to indicate which bid the MA prefers – the MA prefers the bid with the smallest *f-value*. To the right of the figure is a search tree that illustrates the search process that the MA undergoes in finding a suitable plan, which in Figure 23 (a) consists of just the root node. There is another MA present in Figure 23 (MA_2), as TA may interact with more then one MA simultaneously.

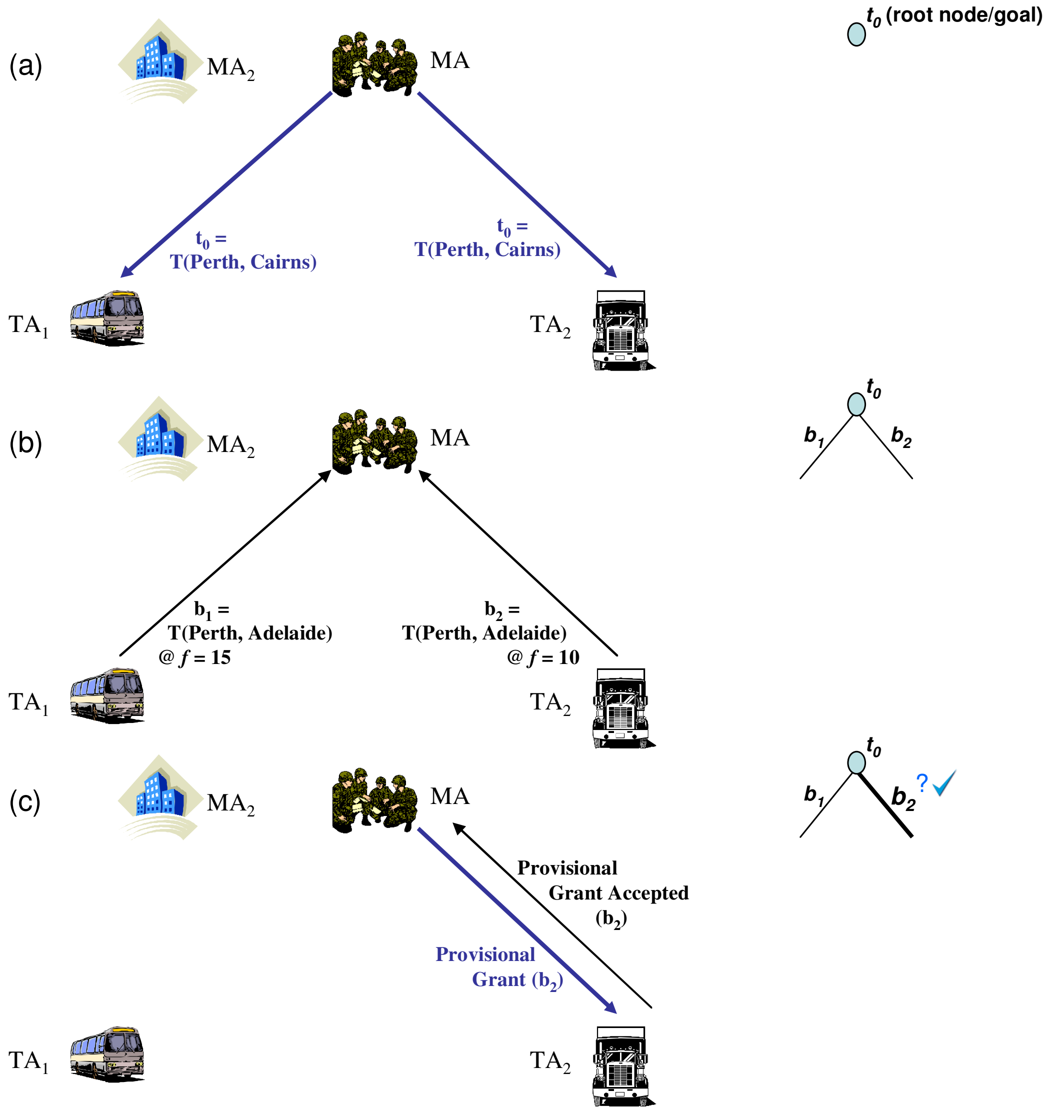
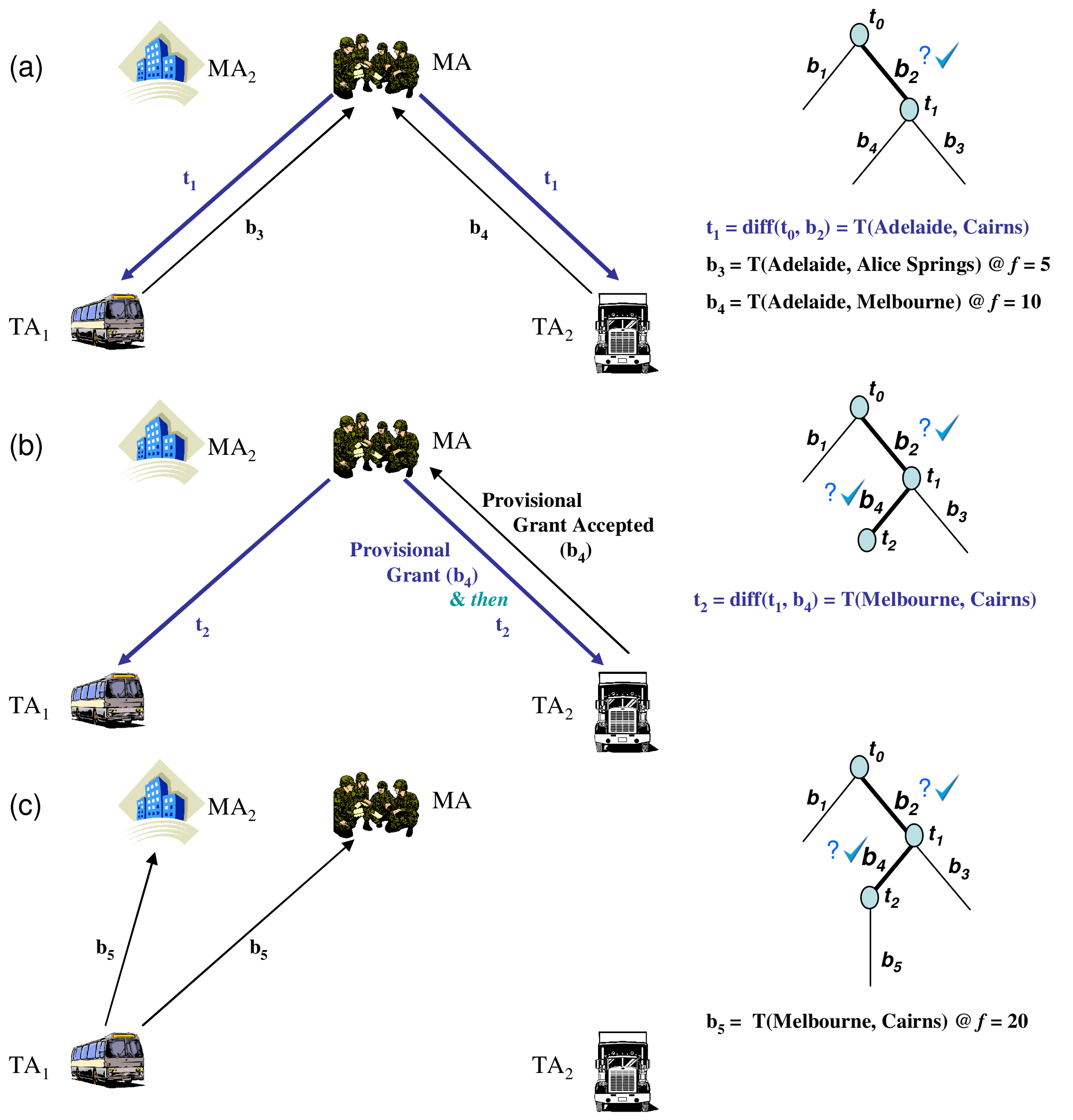


Figure 23. PAP example with (a) ***Task Announcement*** *of task t_0. (b)* ***Bid*** *b_1 and b_2 from TA_1 and TA_2, respectively. (c)* ***Provisional Grant*** *of b_2 and* ***Provision Grant*** *is* ***Accepted****.*

MA's search tree is to the right of the figure. The ?✓ *indicates a provisional grant.*

In Figure 23 (b), both TA_1 and TA_2 submit **bids** b_1 and b_2 respectively, to (partially) achieve t_0. Since b_2 has a smaller *f-value*, MA prefers b_2 over b_1, and therefore, **provisionally grants** b_2 (Figure 23 (c)). The provisional grant is accepted (**Provisional Grant Accepted**) by TA_2. The search tree now consists of the root node t_0, and branches b_1 and b_2, with option b_2 selected to achieve t_0. The ?✓ near b_2 in the search indicates that it has been selected, but the MA is still unsure if the bid will satisfy its needs, and thus MA may reject b_2 later if the bid is found unsuitable.



(a)
MA$_2$
MA
t_1
b_3
b_4
t_1
TA$_1$
TA$_2$
t_0
b_1
b_2 ?✓
t_1
b_4
b_3
t_1 = diff(t_0, b_2) = T(Adelaide, Cairns)
b_3 = T(Adelaide, Alice Springs) @ f = 5
b_4 = T(Adelaide, Melbourne) @ f = 10
(b)
MA$_2$
MA
Provisional Grant Accepted (b_4)
Provisional Grant (b_4) & then t_2
t_2
TA$_1$
TA$_2$
t_0
b_1
b_2 ?✓
t_1
?✓ b_4
b_3
t_2
t_2 = diff(t_1, b_4) = T(Melbourne, Cairns)
(c)
MA$_2$
MA
b_5
b_5
TA$_1$
TA$_2$
t_0
b_1
b_2 ?✓
t_1
?✓ b_4
b_3
t_2
b_5
b_5 = T(Melbourne, Cairns) @ f = 20

Figure 24. Example continued – (a) ***Task Announcement*** *of t_1 (portion of t_0 that bid b_2 did not achieve), then TA* ***Bid*** *b_3 and b_4. (b) b_2* ***Provisionally Granted***, ***Provisional Grant*** *is* ***Accepted***, *then* ***Task Announcement*** *of task t_2. (c) TA$_1$* ***Bid*** *b_5 to MA, but also to MA$_2$ (in a separate interaction), TA$_2$ bids nothing (****No Communication****), as it has no suitable bids.*

The example continues in Figure 24 (and continues through to Figure 29). Figure 24 and Figure 25 illustrate PAP's ability to allow TA to interact with multiple MA simultaneously, in addition to dynamic bids (bids being withdrawn) and backtracking. In Figure 24 (a), since the provisionally granted bid b_2 (to transport resources from Perth to Adelaide) did not completely achieve the task t_0 (to transport resources from Perth to Cairns), the MA **announces** the portion of the task that b_2 did not achieve, which is t_2 (to transport resources from Adelaide to Cairns). MA receives **bid**s b_3 and b_4 to (partially) achieve t_2. In Figure 24 (b), MA **provisionally grants** b_4, which is **accepted** by TA_2, and then **announces** t_2, the portion of the task t_1 that the bid b_4 did not achieve. In Figure 24 (c), TA_1 **bids** b_5 to (fully) achieve t_2. While this occurs, TA_1 is also interacting with MA_2, and also sends the bid b_5 to MA_2 (e.g. to achieve some task for MA_2, this full interaction is not shown), which is allowed as TA_1 is not committed to its bid b_5 (it has not been provisionally granted).

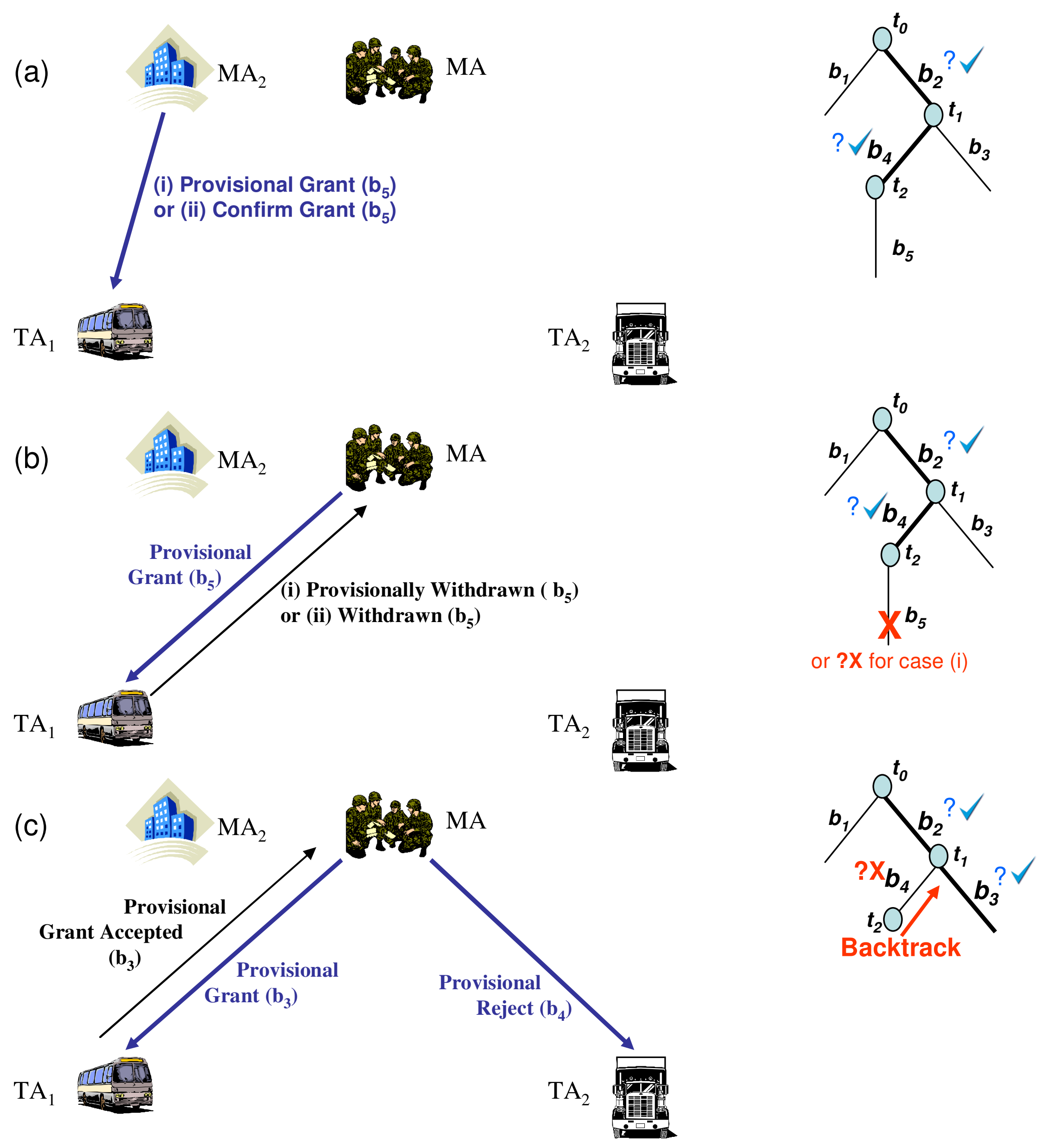


*Figure 25. Example continued – (a) MA$_2$ (i) Provisionally Grants or (ii) Confirm Grants b$_5$. (b) MA tries to **Provisionally Grant** b$_5$, but it is no longer available – it is (i) **Provisionally Withdrawn** or (ii) **Withdrawn**, respectively. (c) No bids to achieve t$_2$, so MA **Backtracks** to t$_1$ by **Provisionally Rejecting** b$_4$, then **Provisionally Granting** b$_3$ for t$_1$, which the **Provisional Grant** is **Accepted**.*

In Figure 25 (a), MA_2 decides to either (i) provisionally grant or (ii) confirm grant (after a initially provisionally granting) bid b_5. Therefore, TA_1 is committed to performing b_5 for MA_2. In Figure 25 (b), MA decides to **provisionally grant** b_5, but since it is no longer available for MA, TA_1 sends MA a (i) **provisionally withdrawn** message (indicated in the search tree by **?**×) or (ii) **withdrawn** message (indicated in the search tree by ×). As previously mentioned our *current* implementation of PAP discards provisionally withdrawn bids to ensure convergence, even though they may become available again at a later time. Hence, in both cases (i) and (ii) the branch b_5 is deleted from the search tree (i.e. the path through b_5 is no longer an option), as shown in Figure 25 (c). In this example, TA_1 does not possess an *updated bid* for task t_2 to replace the (provisionally) withdrawn bid b_5, and thus there is **no communication**.

In Figure 25 (c), the MA has no bids to achieve t_2, and therefore assumes that it can not be achieved (encounters an infeasible solution). The MA decides to **backtrack** by **provisionally rejecting** the bid b_4 (indicated in the search tree by **?**×) for the previous task t_1, and **provisionally granting** another bid option b_3 for t_1, which is **accepted**. Once again, provisionally rejected bids (and their associated task) are discarded in our *current* implementation of PAP to ensure convergence, even though the MA may want to use it again at a later time.

The diagrams in Figure 26 to Figure 28 contain an ordered list of bids for each TA for task t_3, labelled Bids(t_3), which appears directly to the right of each TA. The list is ordered by the f-value, where Bids(t_3) = {<bid_1, $f\text{-}value_1$>, …, <bid_n, $f\text{-}value_n$>}, where $f\text{-}value_1 \leq f\text{-}value_2 \leq \ldots \leq f\text{-}value_n$, hence bid_1 is the best (MA's preferred) bid. The lists will be used to explain PAP's updated bid and updated better bid features.

Figure 26 illustrates PAP's ability to reject and extract new bids from a TA if a submitted bid is found to be unsuitable when *received* (and not due to backtracking as described in Figure 25). In Figure 26 (a), the MA **announces** task t_3, which is to transport resources from Alice Spring to Cairns, and receives **bids** b_6 and b_{11} from TA_1 and TA_2, respectively. It can be seen from the ordered list associated with TA_1 and TA_2 that b_6 and b_{11}, respectively, are the best bids (based on f-value) for the task t_3. In this case, MA finds TA_1's bid b_6 unsuitable (e.g. say MA has a business policy not to transport resources through Mackay), and in Figure 26 (b), MA **provisionally rejects** b_6. TA_1 decides to send an *updated **bid*** b_7, which is the next best bid in its ordered list of bids for t_3, to replace its rejected bid b_6. TA_1 sends the update because even though TA_1's best bid was not suitable, its second best bid may still be better than other bids sent by other TA. In this example, this is the case as b_7 has an f-value of 12, and b_{11}, the only other bid MA has for t_3, has an *f-value* of 20. Therefore, in Figure 26 (c), MA **provisionally grants** b_7 and the **provisional grant is accepted**.

Figure 27 illustrates PAP's ability to allow TA to send updated bids after having bids rejected during backtracking. This provides TA with another chance of having a bid allocated, and ensures the MA has the best selection of bids for its task. In Figure 27 (a), after announcing task t_4, the TA do not submit any bids (no communication) by the deadline. The MA assumes that the task t_4 cannot be achieved (encounters an infeasible solution), and decides to backtrack in Figure 27 (b). Bid b_7 is provisionally rejected and again, TA_1 submits its next best bid for task t_3, which is b_8 (as indicated by TA_1's ordered list of bids for t_3). The MA now has two bids for its task, which are b_8 with an *f-value* of 15 and b_{11} with an *f-value* of 21. The updated bid b_8 is the preferred bid for MA, and thus is provisionally granted in Figure 27 (c), and the provisional grant is accepted. If an updated bid is not allowed in this case, then the MA would not able to select the optimal bid for task t_3.

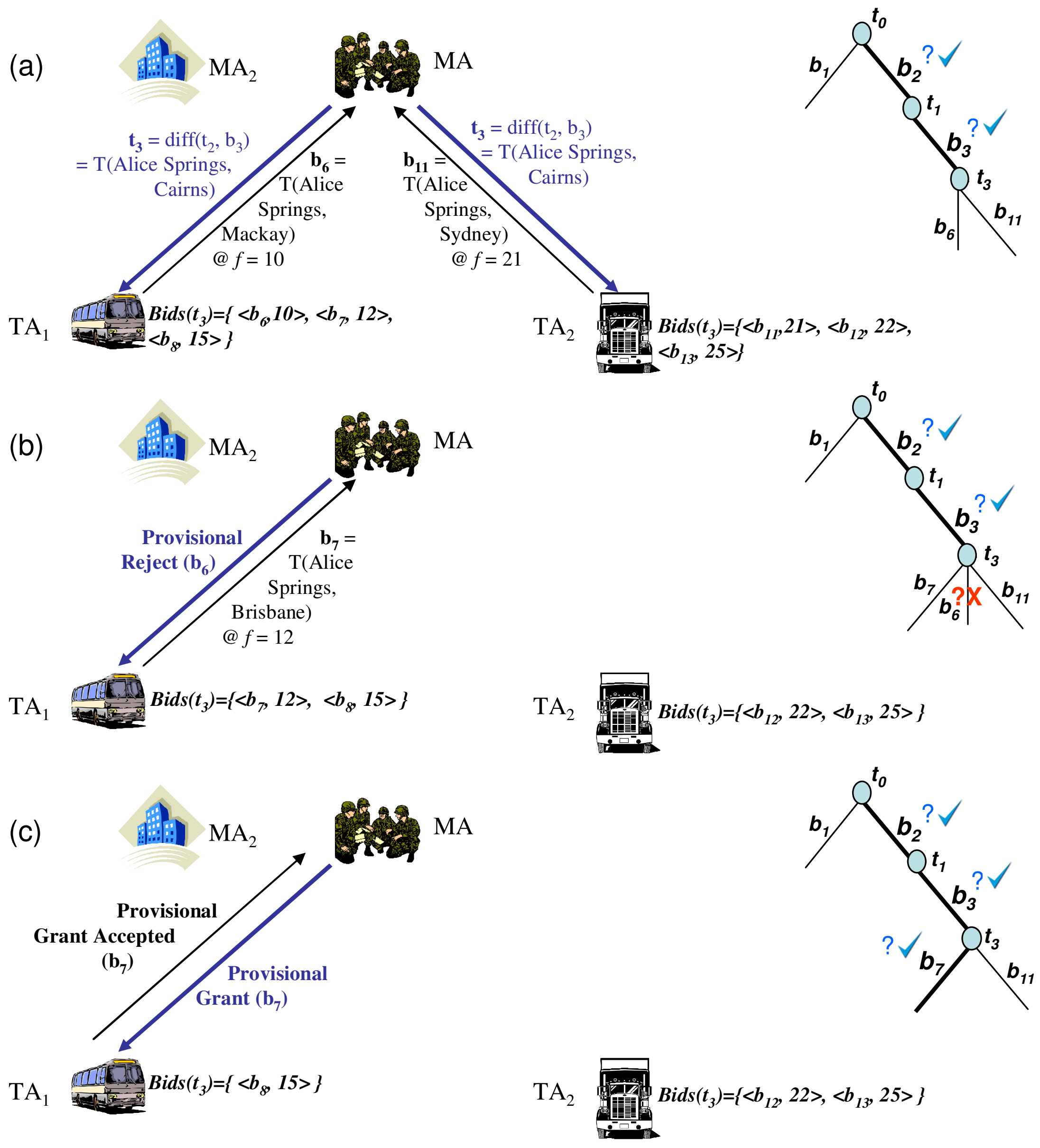

(a)
MA$_2$
MA
t_3 = diff(t_2, b_3) = T(Alice Springs, Cairns)
b_6 = T(Alice Springs, Mackay) @ f = 10
b_{11} = T(Alice Springs, Sydney) @ f = 21
t_3 = diff(t_2, b_3) = T(Alice Springs, Cairns)
TA$_1$
Bids(t_3)={ <b_6,10>, <b_7, 12>, <b_8, 15> }
TA$_2$
Bids(t_3)={<b_{11},21>, <b_{12}, 22>, <b_{13}, 25>}
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 b_6 b_{11}
(b)
MA$_2$
MA
Provisional Reject (b_6)
b_7 = T(Alice Springs, Brisbane) @ f = 12
TA$_1$
Bids(t_3)={<b_7, 12>, <b_8, 15> }
TA$_2$
Bids(t_3)={<b_{12}, 22>, <b_{13}, 25> }
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 b_7 b_6 ? X b_{11}
(c)
MA$_2$
MA
Provisional Grant Accepted (b_7)
Provisional Grant (b_7)
TA$_1$
Bids(t_3)={ <b_8, 15> }
TA$_2$
Bids(t_3)={<b_{12}, 22>, <b_{13}, 25> }
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 ? ✓ b_7 b_{11}

*Figure 26. Example continued – (a)**Task Announcement** of t_3, and TA send **Bids** b_6 and b_7. (b) MA believes b_6 is unsuitable so **Provisionally Rejects** it, and TA$_1$ sends an updated **Bid** to replace it – next best bid in list Bids(t_3). (c) b_7 is **Provisionally Granted** and the **Provisional Grant** is **Accepted**. Bids(t_3) is a function that returns an ordered list of suitable bids for task t_3 – list shown in figure as {<bid, f-value>, …}.*

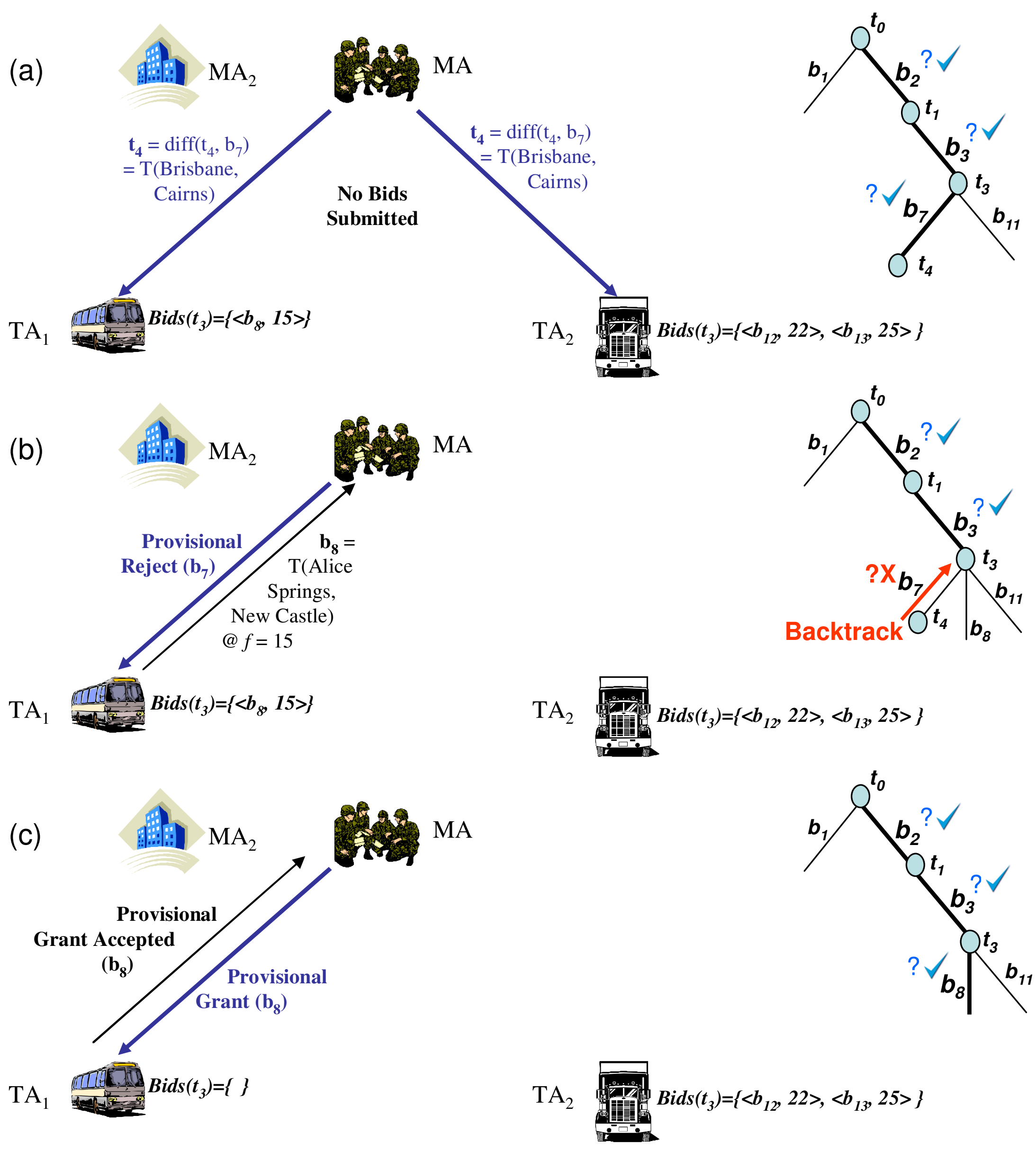
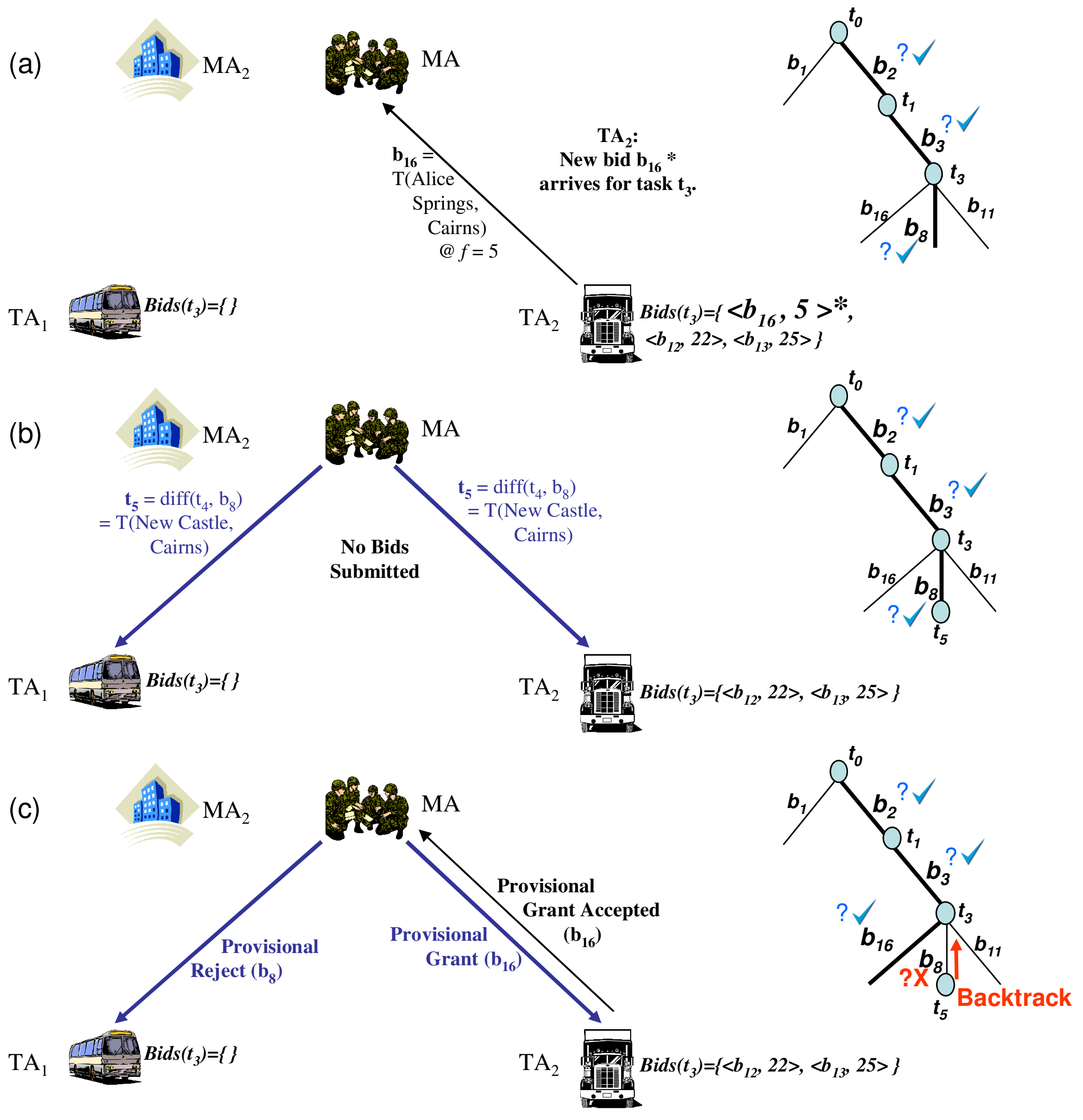

(a)
MA$_2$
MA
t_4 = diff(t_4, b_7) = T(Brisbane, Cairns)
No Bids Submitted
t_4 = diff(t_4, b_7) = T(Brisbane, Cairns)
TA$_1$
Bids(t_3)={<b_8, 15>}
TA$_2$
Bids(t_3)={<b_{12}, 22>, <b_{13}, 25> }
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 ? ✓ b_7 b_{11} t_4
(b)
MA$_2$
MA
Provisional Reject (b_7)
b_8 = T(Alice Springs, New Castle) @ f = 15
TA$_1$
Bids(t_3)={<b_8, 15>}
TA$_2$
Bids(t_3)={<b_{12}, 22>, <b_{13}, 25> }
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 ?X b_7 b_{11} Backtrack t_4 b_8
(c)
MA$_2$
MA
Provisional Grant Accepted (b_8)
Provisional Grant (b_8)
TA$_1$
Bids(t_3)={ }
TA$_2$
Bids(t_3)={<b_{12}, 22>, <b_{13}, 25> }
t_0 b_1 b_2 ? ✓ t_1 b_3 ? ✓ t_3 ? ✓ b_8 b_{11}

Figure 27. Example continued – (a) ***Task Announcement*** *of t_4 but TA have no suitable bids to submit (****No Communication****). (b) MA decides to* ***backtrack****, so* ***Provisionally Rejects*** *b_7, in which case TA$_1$ sends an updated* ***Bid*** *b_8 (its next nest bid). (c) b_8 is* ***Provisionally Granted****, and the* ***Provisional Grant*** *is* ***Accepted****.*

Figure 28 illustrates PAP's ability to keep MA's tasks up-to-date with TA's best possible bids, and even if a bid has been provisionally granted for the task the MA may benefit from this. In Figure 28 (a), after bid b_8 had been provisionally granted for task t_3 (Figure

27 (c)), an opportunity arose for TA_2 to submit a better bid (b_{16}) for t_3 than the previous bid b_{11} that was sent (e.g. because TA_2 acquired new capabilities or a TA_2's bid b_{16} had been freed (rejected) by another MA). According to the bidding policy, if a better bid arrises, then the TA sends this **bid** – an *updated better bid*. In Figure 27 (b), MA **announces** task t_5 but receives no bids for t_5 (**no communication**). MA assumes that no solution is possible for t_5, and therefore in Figure 27 (c) **backtracks** by **provisionally rejecting** b_8. Since TA_1 has no more bids for t_3 (indicated by its ordered list of bids for t_3), it cannot send an updated bid to replace the rejected bid. MA has two bids available for selection, which are b_{16} with an *f-value* of 5 and b_{11} with an *f-value* of 21. b_{16} is the preferred bid, and therefore it is **provisionally granted**, and the **provisional grant is accepted**. If TA_1 did not send the updated better bid (b_{16}) for task t_3, even though MA had already (provisionally) selected a bid for t_3, then MA would not be able to make the optimal choice (at the time) for task t_3 after backtracking.



(a)
MA$_2$
MA
TA$_2$: New bid b_{16} * arrives for task t_3.
b_{16} = T(Alice Springs, Cairns) @ f = 5
TA$_1$
$Bids(t_3)$={ }
TA$_2$
$Bids(t_3)$={ <b_{16}, 5 >*, <b_{12}, 22>, <b_{13}, 25> }
(b)
MA$_2$
MA
t_5 = diff(t_4, b_8) = T(New Castle, Cairns)
No Bids Submitted
t_5 = diff(t_4, b_8) = T(New Castle, Cairns)
TA$_1$
$Bids(t_3)$={ }
TA$_2$
$Bids(t_3)$={<b_{12}, 22>, <b_{13}, 25> }
(c)
MA$_2$
MA
Provisional Reject (b_8)
Provisional Grant (b_{16})
Provisional Grant Accepted (b_{16})
TA$_1$
$Bids(t_3)$={ }
TA$_2$
$Bids(t_3)$={<b_{12}, 22>, <b_{13}, 25> }
t_0 t_1 t_3 t_5 b_1 b_2 b_3 b_8 b_{11} b_{16}
?X
Backtrack

*Figure 28. Example continued – (a) Updated better **Bid** b_{16} is sent for task t_3. (b) **Task Announced** for t_5, but no bids submitted (**No Communication**). (c) MA **backtracks** by **Provisionally Rejecting** b_8, and then **Provisionally Grants** b_{16}, which the **Provisional Grant** is **Accepted**.*

Figure 29 illustrates how MA completes its planning and secures its distributed plan by allocating the relevant tasks, and the ability for PAP to allow MA to withdraw tasks. In Figure 29 (a), since the provisionally granted bid b_{16} fully achieves its task t_3 (diff(t_3, b_{16}) = ∅), the complete task t_0 has been achieved. In Figure 29 (b), the MA **confirm grants** all the bids that it had provisionally granted (b_2, b_3 and b_{16}) in order to secure the distributed plan and allocate the relevant tasks (i.e. the task that the bids achieve) to the respective TA. The MA is now committed to the bids, and neither MA nor TA can decommit (without penalty). The final plan to achieve task t_0 of transporting resources from Perth to Cairns is: b_2 (transport resources from Perth to Adelaide), b_3 (transport resources from Adelaide to Alice Springs) and b_{16} (transport resources from Alice Springs to Cairns). The final plan can be seen by the thick lined path (with ticked bids) in MA's search tree in Figure 29 (b).

In Figure 29 (c), after the MA has completed planning, an opportunity arises for TA_2 to submit a better **bid** for task t_1 than any bids it has previously sent for t_1. Since t_1 has been achieved, and thus is no longer available for achieving (assume the contract for t_1 cannot be terminated in this example), the MA sends a **withdrawn** message. TA_2 can therefore remove the task t_1 from it memory and not consider it for anymore updates.

Note that in the above illustrative example, we purposely use examples where the partial bid that is granted results in *one* remaining task that must be achieved. For example, for the task *T(Perth, Cairns)*, the bid *b = T(Perth, Alice Springs)* results in one remaining task *T(Alice Springs, Cairns)* that must be achieved. Partial bids could be accepted by the MA that result in *split tasks*. Thus, for the task *T(Perth, Cairns)*, a bid *b = T(Adelaide, Alice Springs)* may be selected, resulting in two remaining tasks that must be achieved. These are *T(Perth, Adelaide)* and *T(Alice Springs, Cairns)*. This issue is further discussed in chapter 7.

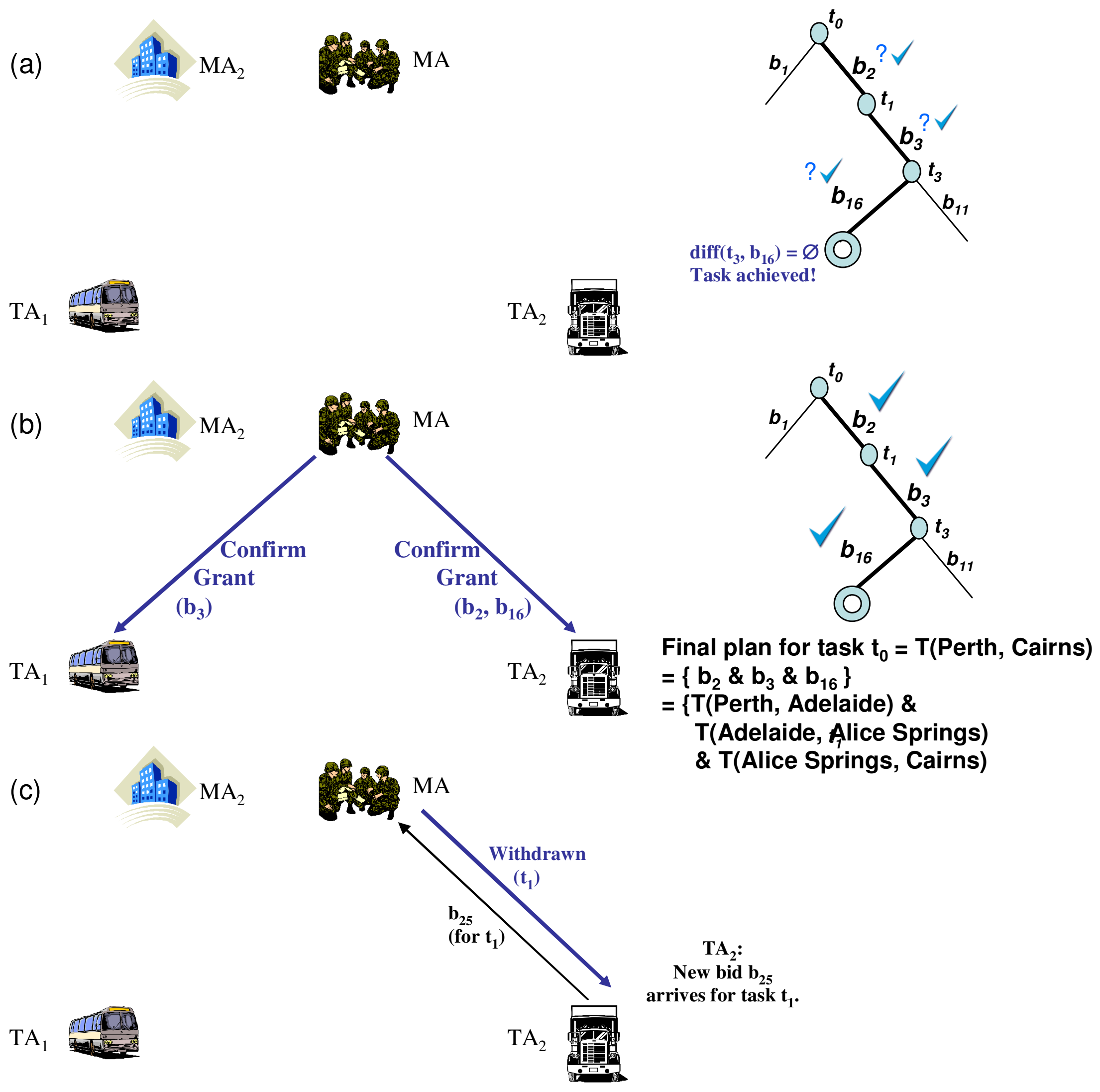


*Figure 29. Example continued – (a) Bid b_{16} fully achieves it task t_3, therefore complete task t_0 is achieved. (b) MA **Confirm Grants** (allocates) all the provisionally granted bids – b_2, b_3 and b_{16}. The distributed plan is secured. (c) TA_2 sends an updated better **Bid** b_{25} for task t_1, and MA replies with a **Withdrawn** message (as the task is achieved).*

5.2 Protocol Features

5.2.1 Greater Planning Flexibility

PAP allows backtracking of single bids, overcoming the shortfall with ECNP to provide greater flexibility with planning. PAP can address problems which require backtracking, such as our transportation domain which may encounter an infeasible solution during planning and must backtrack in order to find a solution. Other domains may also desire backtracking, for example, when a bad solution is encountered during planning. Backtracking can be used to search for a better solution. Later, we will show that PAP runs a decentralised depth-first search.

PAP is able to address planning and task allocation problems that CNP, CNP-ext and ECNP can address. These protocols are not always able to address problems that PAP is able to. Formal proofs are provided in section 5.3.3.

5.2.2 Multiple Auctioneers/MA and the Eager Bidder Problem

As discussed in the previous chapter, the eager bidder problem occurs when bidders submit bids that they are committed to, and therefore cannot submit the same or a conflicting bid elsewhere until the bid is rejected. This may result in the auctioneer missing potential contracts when dealing with multiple auctioneers. PAP's commitment policy overcomes the eager bidder problem because bidders are not committed to submitted bids until it is provisionally granted. This is similar to an approach suggested by Schillo *et al.*, except that they do not allow the auctioneer to (provisionally) reject the bid once the bidder has accepted the (provisional) grant (Schillo, Kray et al. 2002). Therefore, a bidder may submit bids for many tasks, from the same or *multiple auctioneers*, *simultaneously* as they are received, allowing the bidder to be involved in all potential contracts. The bidder only needs to commit to its bid when an auctioneer has a *genuine interest* in the bidder's bid and provisionally grants the bid, in which case the bidder may accept the provisional grant in order to commit to the bid. After a provisional grant, if the bidder, for whatever reason, no longer wants to commit to the bid, i.e. the bid is (provisionally) withdrawn, then the bidder may continue negotiations by sending an updated bid, if desired, as an alternative to its previous bid for the auctioneer's consideration.

CNP-ext allows bidders to manage multiple auctioneers. The protocol also allows bidders to submit bids without commitment during the "pre-bidding" phase. Bidders' bids are not committed until their bids are pre-accepted and the bidder sends a definitive bid. Although CNP-ext does not require bidders to be committed to initial bids, in its specification, they do not necessarily submit conflicting bids. Bidders sort tasks received from auctioneers and make bids based on this order, where tasks (and their bids) lower in the order are dependent on tasks (and their bids) higher in the order. Tasks that commenced lower in the order and whose bids are rejected are moved higher in the order, replacing higher order tasks whose bids were rejected, so that the bidder can submit an improved (updated) bid for the lower order task.

There are a couple of issues with this approach. In an agent system, there may be a greater likelihood that a bidder's bids will be rejected than granted, assuming all bidders have similar capabilities, and thus are all equally likely of having their bid granted. For example, if there are only two bidders per auctioneer, then one would expect a 50% chance that each bid will be granted. If there are any more bidders per auctioneer, which in many applications is likely, then the chance of a grant reduces below 50%, which for *nb* number of bidders the expected chance of acceptance for a bid would be $(1/nb) \times 100\%$. Therefore, for only 10 bidders, only 10% of bids are likely to granted, i.e. a 90% rejection rate. With a large number of rejections and a large number of tasks in the task order, CNP-ext would require a potentially large number of re-orderings of the task order and updated bids. This may result in the auctioneer waiting for a large number of updating iterations as its task is re-ordered and updated by bidders before it converges to a solution. Additionally, an auctioneer may need to wait many iterations for its task to move high enough in the task order of a bidder such that its updated bid is suitable for the auctioneer. Even if the bid is finally granted (pre-accepted), due to the dependency of tasks lower in the task order with those higher in the tasks order, a bidder is unable to commit to the bid (send a definitive bid) until it receives grants for bids for all tasks higher in the task order. Therefore, CNP-ext initial bidding phase (before dependent bids are granted and committed) suffers from a problem similar to the eager bidder problem, where a negotiation regarding one bidder's bid is dependent on the results of negotiations regarding another bid(s).

With PAP, no order is formed in the initial bidding phase; rather the best bid for each task is submitted, even if they all conflict. The first auctioneer to grant their bid will be allocated the bid. Following this, if any other auctioneer tries to provisionally grant its bid, which is likely to be a small number if there are many bidders per auctioneer [13], and the bid conflicts with the provisionally granted bid, then the bidder will (provisionally) withdraw the bid and submit an updated bid, which is its next best available bid. Again, the auctioneer that provisionally grants the updated bid first will be allocated the bid, and any others that do so after this, which is likely to be a smaller number again [14], will receive an updated bid. This process continues until the relevant tasks have been allocated bids. For all the bids that were not successful in the auctioneer provisionally granting it (i.e. the auctioneer rejects the bid), there is no updating, and hence no communication, required. With CNP-ext, if the task for a bid is initially lower in the order, then the bidder could potentially send many updated bids for its rejected bids for the task until the task is at the top of the order and still be unsuccessful in being allocated.

With PAP, bidders' bids are not dependent on negotiations for other bids in the initial bidding phase before they can commit to the bid. Additionally, with many bidders, and thus rejections, and a large number of tasks in the task order, there are likely to be less updating iterations from many agents before an auctioneer converges to a solution or receives a suitable bid from a bidder. Bidders in PAP submit their best bid first, and auctioneers only wait for updates from the individual bidder that (provisionally) withdraws its bid that the auctioneer tries to provisionally grant.

[13] E.g. if 100 bids are sent and there are 10 bidders per auctioneer, then number of bids likely to be granted are $1/10 \times 100 = 10\%$ of 100 bids, which is 10 bids granted. 1 is allocated the bid and 9 are sent updates (bids withdrawn).

[14] E.g. with our previous example, of the 9 auctioneers that are sent updated bids, only 10% of the 9 updated bids are likely to be granted, therefore only 1 is expected to grant the bid, which will get the allocation.

A bidder's ordering of tasks (and their bids) in PAP is determined by the interaction with the auctioneers, such as the type of bids that they require and when they request and secure them, rather than the bidder itself. In our experiments we assume that bidders have no preference for one bid for a task over another because the profit margins are the same for all bids. If this is not the case, as with CNP-ext that orders tasks based on bidders' preferences, then bidders may still submit their best bid for all tasks, even lower priority tasks. If a bid for a lower priority task is provisionally granted, then the bidder may hold the acceptance of the provisional grant until the end of the provisional grant acceptance deadline (*pgad*), in the hope that the higher priority bids may be granted. If not, then the bidder may accept the provisional grant for the lower priority bid, assuming that there is no interest in its higher priority bids, or withdraw the bid if it still believes there is a reasonable chance that its higher priority bids will be granted.

Although PAP does not suffer from the problem of dependence of negotiations between bids in the initial bidding phase, both PAP and CNP-ext suffer from this problem in the later bidding phases when bids are committed by the bidder. If a bidder commits to a bid b_1 which is provisionally granted, and then commits to another bid b_2 which is dependent on b_1, then if b_1 is provisionally rejected, b_2 may not be achievable or be achievable but at a loss. We see an instance of this in the next chapter where the pricing of one bid is dependent on another provisionally granted bid which may be rejected. As we will discuss in the next chapter, we address this by calculating the price for a bid as the mathematical expectation, with the value function being the price of the bid if the dependent bids remains, and using the probabilities of the dependent bids remaining. Therefore, the price is adjusted to allow for the rejection of dependent bids, minimising loss of profit. In the case that a rejection of a dependent bid results in another committed bid being unachievable, the bidder must de-commit from the bid, paying the penalty for de-commitment.

5.2.3 Planning in a Dynamic Environment

PAP is able to cope with a dynamic environment, where old bids and tasks are withdrawn and new bids and tasks arise. With auctioneers' tasks, when a new task arises, because, for example, the selection of a bid for an existing task did not achieve the complete task or the auctioneer received a request to achieve a new (business) goal, the auctioneer starts a new protocol process by announcing the task to bidders, and uses PAP to achieve the task. If a bidder submits a bid for a task which is no longer available, because, for example, the task has been achieved or is unachievable as it was involved in backtracking, then the auctioneer may withdraw the task by sending the bidder a withdrawn message. The bidder may then remove the task from memory so that it does not consider the task for any more bids. With bidders' bids, if an auctioneer tries to provisionally grant a bid that is not longer available, because, for example, the bid (or a conflicting bid) has been granted elsewhere, then the bidder may withdraw the bid by sending a (provisionally) withdrawn message. If a new opportunity arises for the bidder to submit a better bid than it had previously for a task, because, for example, a provisionally granted bid was rejected that released the availability of another bid, then the bidder is able to send the (updated better) bid for the task.

CNP and ECNP do not consider withdrawn tasks and bids, and updated bids. CNP-ext does not explicitly consider withdrawn tasks and bids, although if a bid is withdrawn, a bidder may submit a different definitive (final committed) bid from the (pre-)bid that was granted. This effectively withdraws the granted bid and updates with a new bid, but can only occur once as the bidder is committed to this new (definitive) bid. PAP allows bidders to withdraw many consecutive bids as updated bids that replace withdrawn bids that are not committed to by the bidder until it accepts a provisional grant for it. We do not allow bidders to propose and commit to a final bid that is different to the bid that the auctioneer and bidder have been negotiating over. If a bid is no longer available, the bid must be withdrawn and a new bid sent to recommence the negotiation process. CNP-ext allows bidders to send updated bids when their bid are rejected, resulting in their tasks being re-ordered so that the bidder can submit a better bid for the task.

5.2.4 Updated Bids and Updated Better Bids

Figure 26 to Figure 28 shows the utility of PAP's updated bid and updated better bid features. The updated bids feature allows the bidder to submit its next best bid to replace a bid that has been withdrawn or rejected. The updated bid may be better than bids currently held by the auctioneer, and therefore without this feature, the auctioneer may not be able to access the best potential bid for its task. The updated better bids feature allows the bidders to submit bids which are better than its currently submitted bids, if the opportunity arises, e.g. it acquires new capabilities or a provisionally granted bid is rejected resulting in a better bid for the task becoming available again. Using the updated better bids feature, the auctioneer is ensured that there are no other bids it has not got access to which is better than the bids it currently holds. If one does become available, it will be submitted. In Figure 28 we have shown that this can be of use even if a bid for a task had been provisionally granted because the auctioneer may backtrack, and thus revisit bids for that task, which the updated better bid may be the best bid at the time. Therefore, the updated bids and updated better bids features allow auctioneers to hold the best bids for their tasks at any time, potentially enabling them to obtain the best plan to achieve their task.

CNP-ext uses updating of bids in a slightly different way. CNP-ext does not require that bidders submit their best bid. As a result, CNP-ext uses updating of bids to allow bidders to submit a better bid to replace a rejected bid in order to keep the negotiation of a task continuing and improve their chances of an allocation. PAP uses updating of bids to allow bidders to provide auctioneers with their best set of possible bids for a task at any time.

5.2.5 Consistent with Contracting

An important feature of PAP is its initial bidding phase, namely the bidding without commitment and the provisional grant. Although others, such as ECNP and CNP-ext, have used similar concepts, in applying PAP to real world open-market and e-commerce applications, these concepts must be consistent with real world contracting and agreement processes. We believe that PAP is predominantly consistent with these processes.

The task announcement and bidding in PAP is an *information exchange phase*, as opposed to a formal contracting negotiation with CNP. A task announcement informs bidders of an auctioneer's desire to achieve some task, which could be to acquire or provide some service. Bidding is a way for bidders to inform the auctioneer of possible services they may *desire* to provide or request from the auctioneer. An auctioneer sending a provisional reject after a bid is received informs the bidder that its bid is not suitable for its task, allowing the bidder to submit an updated bid. Therefore, this information phase provides the auctioneers with knowledge of bids (or resources) that could possibly achieve its task. With this information, the auctioneer is able to make more informed decisions regarding taking the next step of a formal contracting negotiation.

The formal contracting negotiations phase begins with a provisional grant, which is equivalent to the auctioneer asking the bidder if it would formally propose its bid, and thus commit to it. The bidder may respond by formally proposing and committing to the bid, by sending a provisional grant accepted message. Alternatively, it may retract its desire to perform the bid, by (provisionally) withdrawing the bid, in which case the bidder may send an updated bid that indicates its new desire of services it is now willing to provide or request. Once the bidder accepts the provisional grant, the contracting agreement is not yet complete – it must be accepted by both parties. The auctioneer must also accept the agreement, which is the allocation of the bid (or the task that the bid achieves) to the bidder. This occurs when the auctioneer confirm grants the bid, which is the final signing of the contract. If the bid does not fully achieve the task, then the auctioneer delays the confirm granting of the bid until it finds a suitable plan to achieve the rest of the task that the bid does not achieve. When a plan is found, the confirm grant is sent, and the contract is now formed. Both the auctioneer and the bidder are committed to the bid. If the auctioneer finds that the bid is not suitable, it may reject the formal offer for the bid, by sending a provisional reject message. The bidder will no longer be committed to the bid.

The information exchange phase mentioned above occurs quite often in the real world. People often discuss their desires, i.e. tasks that they will like to perform or have performed, to others without committing to it. If an utterance of a desire resulted in an individual being committed to it, then there would likely be many broken contracts in the real world. The concept of provisional grant is not new for some industries. Many hotels, for example, will allow an individual (an auctioneer) to place a tentative booking (provisional grant) on a room. The individual is not committed to the booking (service), and thus is allowed to cancel (provisional reject) at a later date. The individual is only committed to the booking when it accepts the service (confirm grant), for example, by paying for it. Organisations which allow this flexibility would likely be preferred over those that do not.

Sandholm and Lesser state that offers (bids) without commitment strategically meaningless (Sandholm and Lesser 2002). The initial bidding in PAP is considered information exchange rather than a formal offer, which occurs during a provisional accept. Reeves *et al.*'s ContractBot generates contracts by automating the discovery, negotiation and execution of contracts (Reeves, Wellman et al. 2002). The contract negotiation is auction-based, used to determine the price and other terms of the contract. We assume contract terms, such as price, in PAP are fixed and are not negotiated over. The terms are defined by the bids sent by bidders, where the auctioneer may or may not accept the terms.

An element of the legal contracting process that PAP and other aforementioned Contract-Net based protocols do not accommodate is the ability for bidders to revoke their offer (withdraw a provisionally granted bid) before the auctioneer accepts it. Excluding revocation from PAP offers computation efficiency, preventing bidders from revoking bids during a search. This will force the auctioneer to backtrack to the revoked bid. Extending PAP to include revocation is the subject of future work.

5.2.6 Reduced Broken Contracts

PAP's information phase reduces the likelihood of broken contracts compared with the traditional CNP, such as Sandholm's CNP (Sandholm and Lesser 1996; Sandholm and Lesser 2002). Sandholm's CNP allows agents to break contracts, or their commitments, and take alternative options (backtrack), but pay a penalty in order to do so. PAP allows agents to test different options first without the commitment, by exchanging information, and when a suitable solution is found, only then do they proceed with the formal contracting negotiations (commitment). Bidders may submit bids for many tasks at the same time. When bids are provisionally granted by the auctioneer, bidders need only commit to bids that they find suitable, and withdraw those that are not. Auctioneers are able to "search" for a suitable plan for its task, testing different options (bids) and backtracking when required, before entering into a full contract.

5.2.7 Reduced Communication and Distributed Processing

PAP was developed to minimise communication since communication is vital in order for PAP to operate, and can be expensive or limited (e.g. bandwidth). Additionally, PAP could be applied to large scale applications, increasing the communication requirements, potentially becoming a bottleneck. Furthermore, computational processing is an issue with large scale systems, which can also be a potential bottleneck. PAP was developed to take advantage of distributed processing, by constraining the processing required by the auctioneer that is performing the planning.

Allowing the auctioneer to submit the bid evaluation function enables bidders to determine their best bid for the auctioneer. Therefore, auctioneers need only communicate one bid, which is their best bid. If their best bid is not preferred by the auctioneer, then any other bid which is worse should not be either. This prevents the bidders from communicating many bids.

With a centralised approach, the auctioneer receives all bidders' bids at the start and processes them in order to find a suitable plan. Although this saves the auctioneer from having tasks, bids, and other speech acts communicated during planning, it has some disadvantages. If the number of possible bids submitted by each bidder is large, then communication could be greater than with PAP (see following sections). There may be potentially complex dependencies between bids that the bidders submit. For example, selecting one bid may result in other bids' price changing or becoming unavailable. These dependencies could become complex and difficult to define. Furthermore, the bidder may not want to release this potentially private information regarding its many/all bids and their dependencies. As we will show later, a centralised approach may not allow the auctioneer take advantage of a dynamic environment. This may result in a worse or infeasible solution compared with PAP, which interacts with the environment during planning.

PAP's feature of allowing auctioneers to send the bid evaluation function and receive one bid from bidders, rather than all bids for processing as in the centralised approach, takes advantage of distributed processing. In the centralised approach, the single auctioneer must process the potentially large number of bids from many bidders, and their dependencies, in order to find suitable bids for its tasks. With PAP, the auctioneer sends the bid evaluation function to the bidders to *allow the bidders to process their bids* and submit the best one for the task. Therefore, the processing of suitable bids for a task is transferred to the potentially many bidders, and hence many distributed processes. This has the additional benefit of decentralisation, allowing the bidders to have greater control over their actions (the bids that they perform for the task at hand) and minimises the information that they need to release.

Communication is further reduced with PAP as it does not require rejection of bids that the auctioneer does not intend to use. This can potentially be a considerable saving in communication if many bids are submitted for a task because typically with contract net approaches, such as CNP, ECNP and CNP-ext, all bids but the one bid that is selected are rejected. With PAP, it does not matter how many bids are submitted, communication (provisional grant, etc.) is only required for the one selected bid (unless backtracking is required, or a submitted bid is unsuitable, and thus is provisionally rejected when received). Not rejecting unused bids during planning also has the benefit of allowing bids to remain available later when required during backtracking. Additionally, not rejecting bids after a plan is found (confirm granted) allows the bids to remain available for replanning if the contracts are broken.

The persistence policy also assists in minimising communication. Bidders and auctioneers in PAP do not inform others that their bids and tasks have changed status, respectively, unless other agents try to *use*, and therefore show interest, in them. If a bidder *uses* an unavailable task by submitting an updated bid for it, only then does the auctioneer inform the bidder that the task is withdrawn. If an auctioneer *uses* an unavailable bid by trying to provisionally granting it, only then does the bidder inform the auctioneer that the bid is withdrawn. We believe this is likely to save communication because in our applications of interest the instances of agents using tasks and bids after the initial task announcement and bidding phase is likely to be low compared with the number of agents that need to be informed of status changes and the number of possible status changes over time. If a bidder sends a bid for a task, then the chance that the auctioneer does use the bid could be small if the auctioneer has many bids. In the chance that the auctioneer does use the bid, it will likely use the bid shortly after it is submitted, in which case the bid is likely to still be available after this short time anyway. If the auctioneer does not use the bid in the first instance, then the chances of using the bid reduces even further as the auctioneer will need to backtrack and again select the bid over many others. If the bidder did keep the auctioneer informed of its bid's status, then it may communicate often as the bid's status may change often over time, possibly being dependent on other bids in the bidder's local plan that are changing over time. Therefore, it is beneficial to inform the auctioneer regarding the status of the bid only at the particular time that the auctioneer wants to use the bid.

When an auctioneer announces a task, the bidders should submit their best bid for the task. We believe it is unlikely that the bidder will find an opportunity to submit a better bid for the task after the length of time it takes the auctioneer to withdraw the task (achieve it or find it unachievable). In the short term, a bidder may be negotiating with many auctioneers, which may, for example, release (provisionally reject) bids in order to create opportunities for updated better bids – and the likelihood that these bid releases could created an updated better bid could be small. After a short time, negotiations should have completed and any bids whose release may allow an updated better bid would have been released, with the task likely to be still available. Therefore, the bidder is not likely to submit many updated bids when the task is withdrawn. If the auctioneer was to keep the bidders informed when its task changes status, then it would have to communicate with all the bidders, which could potentially be a large number.

We will later show that PAP requires less communication than CNP, CNP-ext, ECNP, and the centralised combinatorial auction approaches (centralised auctioneer described above) under certain circumstances. We believe these circumstances are valid in many real world applications. Additionally, in the chapter 7 we will show how communication is constrained in the transportation domain when tasks in one node of an auctioneer's search tree (or PAP planning process) is repeated in its child nodes. The repeated tasks do not have to be re-announced, and the bidders are still able to submit their best bids – as long as conditions on the auctioneer's bid evaluation function are met.

5.3 Formal Analysis

5.3.1 Problems Addressed by CNP, CNP-ext, ECNP and PAP

In section 4.1, we presented three types of planning and task allocation problems:

1. *Task allocation* – an auctioneer's set of tasks T^{alloc} is allocated to a single bidder, by the acceptance of one of its bids, to attain the auctioneer's goal *G*.
2. *Task allocation with bid planning* – an auctioneer's set of tasks T^{alloc} is allocated to a set of bidders, by finding (bid planning) and accepting a set of bids held by the bidders, to attain the auctioneer's goal *G*.
3. *Planning and task allocation* – an auctioneer must find a suitable set of tasks T^{alloc} among a collection of sets of tasks, and allocate T^{alloc} to a set of bidders, by finding (bid planning) and accepting a set of bids held by the bidders, to attain the auctioneer's goal *G*.

CNP and CNP-ext were developed to achieve the problem definition (1) above, which is the *task allocation* problem of allocating (all or some of) an auctioneer's set of tasks to a single bidder. ECNP and PAP are also able to address this problem if the auctioneer only receives bids that fully achieve the announced set of tasks T^{alloc}. Problem definition (2) above, which is *task allocation with bid planning*, involves an auctioneer with a predefined plan, i.e. the (single) set of tasks, to achieve its goal *G*. The auctioneer must allocate elements within the set of tasks to different bidders as a single bid, and hence a single bidder cannot achieve the auctioneer's set of tasks. It is a bid planning problem as the auctioneer must plan over the possible set of bids that can be used to achieve its set of tasks in order to find a suitable bid allocation plan. The combinatorial auctions problem, which we apply PAP to in chapter 6, is of the same class of problems.

Although there is planning with respect to the bids in problem definition (2), from the auctioneers perspective, it is essentially a task allocation problem as the auctioneer is allocating its predefined set of tasks to bidders. There is no planning with respect to the auctioneer's goal that must be achieved. With problem definition (3), *planning and task* *allocation*, there is planning with respect to the auctioneer's goal as the set of tasks to achieve the goal is not predefined. Therefore, the auctioneer must solve the problem of finding a suitable plan (or task-decomposition) for its goal, in addition to the bid planning problem of finding and accepting suitable set of bids from bidders that can achieve the plan (set of tasks) for the goal. Our transportation problem that we apply PAP to in the next chapter is associated this class of problems. Both ECNP and PAP are able to address both problems (2) and (3), but due to PAP's flexible planning and ability to plan with multiple auctioneers simultaneously and in a dynamic environment, PAP is able to address a larger range of problems of this class than ECNP.

5.3.2 PAP Negotiation

We will define PAP using the negotiation (or protocol process) based on the description presented in section 4.1, and the Protocol Flow Diagram illustration in Figure 21. Since the description of PAP is complicated with new protocol processes spawned due to the partial fulfilment of its set of tasks $\Omega \subseteq$ T, multiple negotiations between an auctioneer and bidder may be required to achieve T. We use the subscript x to indicate the x^{th} protocol process, and hence x^{th} negotiation, between auctioneer α and bidder γ in order to achieve T. We have a negotiation defined as γ

$$Neg_x = < \alpha, \gamma, \Omega_x, \Sigma_x, S_x, A, \delta_x, s_x, F_x >$$

where α is an auctioneer in the negotiation, and γ is a bidder in the negotiation. Other tuples are dependent on the protocol:

- $\Omega_x \subseteq T$, is the set of tasks that the agents α and γ are negotiating over in order to fully or partially achieve it, and $\Omega_{x+1} \subseteq \Omega_x \subseteq \ldots \subseteq \Omega_0 = T$. $x = 0$ denotes the negotiation corresponding to the initial (PAP) protocol process to achieve T. $x + 1$ denotes the negotiation (Neg_{x+1}) corresponding to the protocol process to achieve Ω_{x+1}, spawned from the x^{th} protocol process negotiation (Neg_x) due to the *partial* achievement of Ω_x.
- Σ_x is the set of bids that the agents α and γ are negotiating over to fully or partially achieve the set of tasks Ω_x.

- S_x = {step1$_x$, step2$_x$, step3$_x$, step4$_x$, step5$_x$, step1$_{x+1}$, step5$_{x-1}$} ∪ {exit$^{NC}_x$, exit$^{C}_x$} is the set of possible states that are accessible by the protocol in the current (x^{th}) protocol process. exit$^{NC}_x$ is the state which exits with *no* contract and exit$^{C}_x$ is the state which exits with a contract.

- $A = A_\alpha \cup A_\gamma \cup A_e$ is the set of possible speech acts or events where: A_α is the set of speech acts associated with the auctioneer; A_γ is the set of possible speech acts associated with the bidder; and A_e is the set of events (common to both agents). A_α = ({**task announcement**}× Ω) ∪ ({**provisional grant**}× Σ) ∪ ({**withdrawn**}× Ω) ∪ ({**provisional reject**}× Σ) ∪ ({**confirm grant**}× Σ); A_γ = ({**bid**}× Σ) ∪ ({**provisional grant accepted**}× Σ) ∪ ({**provisionally withdrawn**}× Σ) ∪ ({**withdrawn**}× Σ); A_e = ({**no communication**}) ∪ ({**backtracking**}).

- δ ⊆ (S_x×A×S) specifies the transitions between the possible states in the x^{th} protocol process, such that δ is a function, with δ(step1$_x$, < **task announcement**, ω>) = step2$_x$; δ(step2$_x$, <**bid**, σ>) = step3$_x$; δ(step2$_x$, <**no communication**>) = step3$_x$; δ(step2$_x$, <**no communication**>) = exit$^{NC}_x$; δ(step3$_x$, <**provisional grant**, σ>) = step4$_x$; δ(step3$_x$, < **withdrawn**, ω>) = exit$^{NC}_x$; δ(step3$_x$, <**provisional reject**, σ>) = step2$_x$; δ(step3$_x$, <**no communication**>) = exit$^{NC}_x$; δ(step3$_x$, <**backtracking**>) = step5$_{x-1}$; δ(step4$_x$, <**provisional grant accepted**, σ>) = step5$_x$; δ(step4$_x$, <**provisional grant accepted**, σ>) = step1$_{x+1}$; δ(step4$_x$, <**provisionally withdrawn**, σ>) = step2$_x$; δ(step4$_x$, <**withdrawn**, σ>) = step2$_x$; δ(step5$_x$, <**confirm grant**, σ>) = exit$^{C}_x$ & step5$_{x-1}$ (*see below*); δ(step5$_0$, <**confirm grant**, σ>) = exit$^{C}_0$; δ(step5$_x$, <**provisional reject**, σ>) = step2$_x$.

- s_x = step1, is the start state.

- F_x = {exit^{NC}, exit^{C}} is the set of final states.

(exit$^{C}_x$ & step5$_{x-1}$) asserts that the participating agents exit the current (x^{th}) protocol process with a contract, and then proceeds to state step5 of the previous (x-1^{th}) protocol process.

5.3.3 Planning Flexibility

We formally compare the planning flexibility of PAP with CNP, CNP-ext and ECNP. Our analysis shows that PAP is able to perform the planning and task allocation that these contract net-based approaches can. However, there are planning problems that CNP, CNP-ext and ECNP cannot address that PAP is able to solve. Before we commence the analysis, we introduce some notation.

Definition 1: An agent system consisting of a set of agents $\mathscr{A}$ to solve a planning problem comprises an auctioneer $\alpha \in \mathscr{A}$ and a set of bidders $\Gamma \subseteq \mathscr{A} \setminus \{\alpha\}$. The auctioneer comprises a set of tasks T_j, where T_0 is the initial set of tasks. The index $j > 0$ is used to denote new sets of tasks that are created in achieving T_0 as the protocol (planning) proceeds. The auctioneer allocates bid(s) submitted by bidders in order to achieve T_0. For each announced set of tasks T_j, the auctioneer receives a set of bids B_j from a set of bidders in order to fully or partially achieve T_j. $B_j = \{b_{1,j}, \ldots, b_{n,j}\}$, where n is the number of bids submitted for the auctioneer's j^{th} set of tasks T_j. The auctioneer's preference among the bids B_j for T_j is denoted $prefer(T_j, B_j)$. $diff(T_j, b_{i,j})$ is a function that returns a set of tasks T_{j+1}, which is the remaining set of tasks that the bid $b_{i,j}$ does not achieve from the set of tasks T_j. If $T_{j+1} = \varnothing$, then the bid $b_{i,j}$ fully achieves the set of tasks T_j.

Theorem 1: PAP finds an equivalent task allocation to CNP for an agent system $\mathscr{A}$ if:

- $\mathscr{A}$ comprises a single auctioneer
- bidders can only submit bids that can fully achieve the auctioneer's announced set of tasks, and
- PAP precludes withdrawals, the updating of bids, rejection of bids and backtracking.

Proof:

<u>Task allocation with CNP:</u>

Refer to the CNP specification in section 4.2 (Figure 11), and Definition 1. The auctioneer α commences at step 1 by **announcing** its initial set of tasks T_0, and proceeds to step 2. The bidders Γ submit **bids** B_0 in order to achieve T_0 such that they are full bids, i.e. $\forall k\ [b_{k,0} \in B_0 \Rightarrow diff(T_0, b_{k,0}) = \varnothing]$, ignoring bidders that submit refuse messages. Control proceeds to step 3. The auctioneer α selects its most preferred bid, which is *prefer(*T_0*,* B_0*),* **grants** the bid, and **rejects** all other bids $B_0 \backslash \{ prefer(T_0, B_0)\}$. The auctioneer α has bid *prefer(*T_0*,* B_0*)* allocated to achieve the set of tasks T_0.

Task allocation with PAP:

Refer to the PAP specification in section 5.1 (Figure 21), and Definition 1. The auctioneer α commences at step 1 by **announcing** the set of tasks T_0, and proceeds to step 2. The bidders Γ submit **bids** B_0 in order to achieve T_0 such that the bids fully achieve the set of tasks, i.e. $\forall k\ [b_{k,0} \in B_0 \Rightarrow diff(T_0, b_{k,0}) = \varnothing]$, ignoring bidders that do not submit bids – **no communication**. PAP proceeds to step 3. Since we do not consider withdrawals and backtracking in the context of the theorem, these speech acts or events (e.g. withdrawn, provisionally reject and backtracking) can be ignored at step 3. Therefore, at step 3, the auctioneer α selects its preferred bid, *prefer(*T_0*,* B_0*)* and **provisionally grants** the bid. PAP proceeds to step 4, where the bidder of bid *prefer(*T_0*,* B_0*)* **accepts the provisional grant**, as the bidder is unable to (provisionally) withdraw in the context of the theorem. Since the bid *prefer(*T_0*,* B_0*)* fully achieves the task T_0 in the context of the theorem, PAP takes control path (f) to step 5. The auctioneer must now **confirm grant** the bid *prefer(*T_0*,* B_0*)*, as provisional rejects are precluded in the context of the theorem. Since bidders are not committed to their submitted bids in PAP, rejection of the other bids $B_0 \backslash \{prefer(T_0, B_0)\}$ is not required. The auctioneer α has bid *prefer(*T_0*,* B_0*)* allocated to achieve task T_0.

Final allocation for both protocols

The resulting task allocation for both CNP and PAP are equivalent in the context of the theorem, with both allocating bid *prefer(*T_0*,* B_0*)* for task T_0.

Q.E.D.

Definition 2: For any set of bids $\boldsymbol{B}$ and an auctioneer $\alpha \in \mathscr{A}$, we write *suitable(*α*,* $\boldsymbol{B}$*)* when α regards the set of bids (plan) $\boldsymbol{B}$ as suitable.

Theorem 2: PAP finds an equivalent plan and task allocation as ECNP for an agent system $\mathscr{A}$ if

- $\mathscr{A}$ has a single auctioneer
- PAP precludes withdrawals, the updating of bids, and the rejection of bids on submission, and
- PAP only backtracks when a plan is found and deemed unsuitable.

Proof:

<u>Task allocation with ECNP:</u>

Refer to the ECNP specification in section 4.7 (Figure 18), Definition 1 and Definition 2. The auctioneer α commences at step 1 by **announcing** the set of tasks T_0, and proceeds to step 2. The bidders Γ submit **bids** B_0 in order to achieve T_0, and ECNP proceeds to step 3. The auctioneer α selects its most preferred bid, which is *prefer*(T_0, B_0) and **provisionally grants** the bid. The other bids B_0\{*prefer*(T_0, B_0)} are **provisionally rejected**. If the bid *prefer*(T_0, B_0) does not fully achieve the set of tasks T_0, then the remaining set of tasks that are not achieved, T_1 = *diff*(T_0, *prefer*(T_0, B_0)), becomes the new set of tasks to announce at step 1 (proceed back to step 1, where a new protocol process is created to achieve T_1). This process continues, e.g. receive bids B_1 for T_1, provisionally grant *prefer*(T_1, B_1) for T_1 and rejecting the others B_1\ {*prefer*(T_1, B_1)}, and re-announcing the task T_2 = *diff*(T_1, *prefer*(T_1, B_1)) at step 1 again, etc., until we arrive at task T_j such that *diff*(T_j, *prefer*(T_j, B_j)) = ∅, i.e. the task T_j (and hence T_0) is fully achieved. In this case, ECNP proceeds to step 4.

The auctioneer now has the set of bids (a plan) ***B*** = {*prefer*(T_0, B_0), *prefer*(T_1, B_1), …, *prefer*(T_j, B_j)} that are provisionally granted. If the plan is suitable, i.e. *suitable*(α, ***B***), then the auctioneer α **confirm grants** all the bids in ***B*** in order to allocate the tasks. If the plan is not suitable, i.e. ¬ *suitable*(α, ***B***), then the auctioneer **confirm rejects** all the bids in ***B***. Therefore, the auctioneer α has a plan and allocation for its set of tasks T_0 of ***B*** if the plan is suitable, or a rejected plan if ***B*** is not suitable.

Task allocation with PAP:

Refer to the PAP specification in section 5.1 (Figure 21), Definition 1 and Definition 2. The auctioneer α commences at step 1 by **announcing** the set of tasks T_0, and proceeds to step 2. The bidders Γ submit **bids** B_0 in order to achieve T_0, ignoring bidders that do not submit bids – **no communication**. PAP proceeds to step 3. Since we do not consider withdrawals, rejection on bid submission and backtracking until the end, in the context of the theorem, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. Therefore, at step 3, the auctioneer α selects its preferred bid, which is $prefer(T_0, B_0)$ and **provisionally grants** the bid. Since bidders are not committed to their bids, the auctioneer does not need to reject the other bids $B_0 \backslash \{prefer(T_0, B_0)\}$. PAP proceeds to step 4, the bidder of bid $prefer(T_0, B_0)$ **accepts the provisional grant** of its bid as (provisional) withdrawals are not permitted in the context of the theorem. If the bid $prefer(T_0, B_0)$ does not fully achieve the set of tasks T_0, then the remaining set of tasks that is not achieved, $T_1 = diff(T_0, prefer(T_0, B_0))$, becomes the new set of tasks to announce at step 1 (proceed back to step 1, where a new protocol process is created to achieve T_1). This process continues, e.g. receive bids B_1 for T_1, provisionally grant bid $prefer(T_1, B_1)$ for T_1, and re-announcing the set of tasks $T_2 = diff(T_1, prefer(T_1, B_1))$ at step 1 again, etc., until we arrive at set of tasks T_j such that $diff(T_j, prefer(T_j, B_j)) = \emptyset$, i.e. the set of tasks T_j (and hence T_0) is fully achieved. In this case, PAP proceeds to step 5.

The auctioneer now has the set of bids (a plan) $\boldsymbol{B} = \{prefer(T_0, B_0), prefer(T_1, B_1), \ldots, prefer(T_j, B_j)\}$ provisionally granted. If the plan is suitable, i.e. $suitable(\alpha, \boldsymbol{B})$, then the auctioneer **confirm grants** all the bids in $\boldsymbol{B}$ in order to allocate the tasks. If the plan is not suitable, i.e. $\neg suitable(\alpha, \boldsymbol{B})$, the auctioneer rejects the complete plan $\boldsymbol{B}$ by doing the following. The auctioneer **provisionally rejects** $prefer(T_j, B_j)$. This is the last bid that was provisionally granted, as this will be the current protocol process that the auctioneer is executing (to achieve T_j). According to the PAP specification, if this occurs, PAP proceeds to step 2. Since all bids are already submitted, as updated bids are precluded in the context of the theorem, the protocol proceeds to step 3 (path (b) with **no communication**). PAP may use **backtracking** at step 3. This results in PAP proceeding to step 5 of the previous protocol process for T_{j-1} (takes control path (d) since it is not at the initial protocol process, which is for T_0). Again, *prefer*(T_{j-1}, B_{j-1}) is **provisionally rejected** and PAP proceeds to step 2, etc., which continues until PAP arrives to step 5 of the initial protocol process for T_0 and provisionally rejects *prefer*(T_0, B_0). PAP proceeds to step 2, takes path (b) with **no communication**, and **backtracks** again. Since PAP is backtracking from the initial (root) set of tasks T_0, PAP takes path (c) to exit the protocol. The complete plan ***B*** has been rejected.

Note that ECNP confirm rejects the provisionally granted bids in ***B***, which has the effect of decommitting the bidders from their provisionally granted bids. PAP's *provisional reject* does the same, except it is *provisional* in case the auctioneer wants to grant the bid again at a later time. Using PAP, the auctioneer has a plan and allocation for its set of tasks T_0 of ***B*** if the plan is suitable, or a rejected plan if ***B*** is not suitable.

Final allocation for both protocols

Both ECNP and PAP produce the same result, which is a plan ***B*** = {*prefer*(T_0, B_0), *prefer*(T_1, B_1), …, *prefer*(T_j, B_j)} for the set of tasks T_0 that is either allocated if suitable or rejected if unsuitable.

Q.E.D.

In order to analyse PAP with CNP-ext, we need to extend Definition 1 to include a time component to the set of bids *B*, as this may change with time.

Definition 3: An agent system consisting of a set of agents $\mathscr{A}$ to solve a planning problem comprises an auctioneer $\alpha \in \mathscr{A}$ and a set of bidders $\Gamma \subseteq \mathscr{A} \setminus \{\alpha\}$. The auctioneer comprises a set of tasks T_j, where T_0 is its initial set of tasks. The index $j > 0$ is used to denote new sets of tasks that are created in achieving T_0 as the protocol (planning) proceeds. The auctioneer allocates bid(s) submitted by bidders in order to achieve T_0. At time τ, for each announced set of tasks T_j, the auctioneer has a set of bids $B_{\tau,j}$ that are submitted by the set of bidders in order to fully or partially achieve T_j. The submitted set of bids $B_{\tau,j}$ for the set of tasks T_j may change over time because bidders may send updated bids throughout the protocol (planning) process. $B_{\tau,j} = \{b_{1,j}, \ldots, b_{n,j}\}$ where n is the number of bids submitted up to time τ for the auctioneer's j^{th} set of tasks T_j. The auctioneer's preference among the bids $B_{\tau,j}$ for T_j is denoted *prefer*(T_j, $B_{\tau,j}$). *diff*(T_j, $b_{i,j}$) is a function that returns a set of tasks T_{j+1}, which is the remaining set of tasks that the bid $b_{i,j}$ does not achieve from the set of tasks T_j. If $T_{j+1} = \varnothing$, then the bid $b_{i,j}$ fully achieves the set of tasks T_j. $bidder(b_{i,j})$ is a function which returns a bidder $\gamma \in \Gamma$ that submitted the bid $b_{i,j}$.

Definition 4: With CNP-ext, δ_τ is a definitive bid submitted by bidder $bidder(\delta_\tau) \in \Gamma$ after its pre-bid $b \in B_{\tau,j}$ is provisionally granted by the auctioneer α from a list of submitted pre-bids $B_{\tau,j}$ for the set of tasks T_j. After a definitive bid δ_τ is submitted at time $\tau' > \tau$ for $bidder(\delta_\tau)$'s pre-bid b, δ_τ replaces its associated (pre-) bid b in the bid list $B_{\tau',j}$, i.e. $\delta_\tau \in B_{\tau',j}$ where $[b \neq \delta_\tau \Rightarrow b \notin B_{\tau',j}]$ & $[b = \delta_\tau \Rightarrow b \in B_{\tau',j}]$.

Theorem 3: PAP finds an equivalent task allocation as CNP-ext if

- PAP only allows bids that can fully achieve the auctioneer's announced set of tasks,
- PAP precludes withdrawal of tasks, rejection of bids on submission, provisionally withdrawn bids, and backtracking,
- the condition used for provisionally rejecting a provisionally granted bid with PAP is that the provisionally granted bid is no longer preferred over a newly submitted updated bid, and
- if a bidder's updated bid submitted to replace its withdrawn bid is provisionally granted before any other submitted bids, then the bidder using PAP is not able to withdraw the bid for the second time.

Proof:

Task allocation with CNP-ext:

Refer to the CNP-ext specification in section 4.5 (Figure 13), Definition 3 and Definition 4. The auctioneer α commences at step 1 by **announcing** the set of tasks T_0, and the protocol proceeds to step 2. At time $t1$, the bidders Γ submit **PreBids** $B_{t1,0}$ in order to achieve T_0 such that they are full bids, i.e. $\forall k\ [b_{k,0} \in B_{t1,0} \Rightarrow diff(T_0, b_{k,0}) = \varnothing]$, ignoring bidders that do not submit any bids (**no communication**). CNP-ext proceeds to step 3. The auctioneer α selects its most preferred pre-bid, which is $prefer(T_0, B_{t1,0})$ and sends the bidder $bidder(prefer(T_0, B_{t1,0}))$ a **provisional grant**, and the protocol proceeds to step 4 for agents α and $bidder(prefer(T_0, B_{t1,0}))$. The other bidders $\Gamma \backslash \{bidder(prefer(T_0, B_{t1,0}))\}$ associated with the other pre-bids $B_{t1,0} \backslash \{prefer(T_0, B_{t1,0})\}$ are sent **provisional reject** messages, and the protocol proceeds to step 2 where these bidders may submit an updated pre-bid. The **confirm reject** at step 3 is used for bidders still at this step in the negotiation when the auctioneer α finds a solution (a bid is confirm granted). In this case, negotiations to achieve the set of tasks T_0 ends between these bidders and the auctioneer α (discussed later). At step 4, the bidder $bidder(prefer(T_0, B_{t1,0}))$ sends a **definitive bid** δ_{t1} for provisionally granted pre-bid $prefer(T_0, B_{t1,0})$, which may be different to the bid that was provisionally granted. From Definition 4, δ_{t1} replaces the pre-bid $prefer(T_0, B_{t1,0})$ in the bid list $B_{t2,0}$ for $t2 > t1$. $bidder(\delta_{t1}) = bidder(prefer(T_0, B_{t1,0}))$.

At step 5, there are four situations: **(i)** The definitive bid δ_{t1} is the same as the provisionally granted pre-bid $prefer(T_0, B_{t1,0})$ and *no other* bidder $\gamma \in \Gamma$ where $\gamma \neq bidder(\delta_{t1})$ sends an updated pre-bid by time $t2 > t1$ that is better than δ_{t1}, i.e. $prefer(T_0, B_{t1,0}) = \delta_{t1} = prefer(T_0, B_{t2,0})$, and $prefer(T_0, B_{t1,0}) \in B_{t2,0}$. **(ii)** The definitive bid δ_{t1} is the same as the provisionally granted pre-bid $prefer(T_0, B_{t1,0})$ but *some other* bidder $\gamma \in \Gamma$ where $\gamma \neq bidder(\delta_{t1})$ submits a pre-bid by time $t2 > t1$ that is better than δ_{t1}, i.e. $prefer(T_0, B_{t1,0}) = \delta_{t1} \neq prefer(T_0, B_{t2,0})$, and $prefer(T_0, B_{t1,0}) \in B_{t2,0}$. **(iii)** The definitive bid δ_{t1} differs to that of the provisionally granted pre-bid $prefer(T_0, B_{t1,0})$ [15] but is still better than *any other* submitted pre-bids, including updated pre-bids, submitted by time $t2 > t1$ by bidders $\gamma \in \Gamma$ where $\gamma \neq bidder(\delta_{t1})$, i.e. $prefer(T_0, B_{t1,0}) \neq \delta_{t1} = prefer(T_0, B_{t2,0})$, where $prefer(T_0, B_{t1,0}) \notin B_{t2,0}$, and hence $prefer(T_0, B_{t1,0})$ is withdrawn. **(iv)** The definitive bid δ_{t1} differs from the provisionally granted pre-bid $prefer(T_0, B_{t1,0})$ [16] and *some other* bidder $\gamma \in \Gamma$ where $\gamma \neq bidder(\delta_{t1})$ submits a pre-bid by time $t2 > t1$ that is better than δ_{t1}, i.e. $prefer(T_0, B_{t1,0}) \neq \delta_{t1} \neq prefer(T_0, B_{t2,0})$ where $prefer(T_0, B_{t1,0}) \notin B_{t2,0}$, and hence $prefer(T_0, B_{t1,0})$ is withdrawn.

[15] This is equivalent to withdrawing the bid and submitting a new updated bid in the PAP.

[16] This is equivalent to withdrawing the bid and submitting a new updated bid in the PAP.

The **confirm reject** at step 5 is used primarily to eliminate the bidder *bidder*(δ_{t1}) from future negotiations, as a means to discourage bidders from undesirable behaviour. This may occur, for example, if the bidder submits a definitive bid δ_{t1} much lower than its pre-bid *prefer*(T_0, $B_{t1,0}$). The confirm reject does not affect the proof, but results in a reduced set of bids $B_{t2,0}\backslash\{\delta_{t1}\}$ at time $t2 > t1$ for T_0, from a reduced set of bidders $\Gamma\backslash\{bidder(\delta_{t1})\}$. We now investigate all of the four situations described above.

Situation (i): The auctioneer will send the bidder *bidder*(δ_{t1}) a **confirm grant** for its definitive bid δ_{t1}, and as mentioned above, all other bidders $\Gamma\backslash\{bidder(\delta_{t1})\}$ which are still at step 3 will receive a **confirm reject** for their unsuccessful pre-bids $B_{t2,0}\backslash\{\delta_{t1}\}$. The auctioneer now has bid $\delta_{t1} = prefer(T_0, B_{t2,0})$ allocated for the set of tasks T_0, where $\delta_{t1} = prefer(T_0, B_{t1,0})$.

Situation (ii): The auctioneer sends the bidder *bidder*(δ_{t1}) a **provisionally reject** message for its definitive bid δ_{t1}, resulting in the protocol proceeding to step 2, allowing the bidder *bidder*(δ_{t1}) to submit an updated **pre-bid** if it desires. The auctioneer α who is at step 3 with another bidder $bidder(prefer(T_0, B_{t2,0})) \neq bidder(\delta_{t1})$ sends this bidder a **provisional grant** for its now preferred bid $prefer(T_0, B_{t2,0})$. The protocol proceeds to step 4, and $bidder(prefer(T_0, B_{t2,0}))$ submits a **definitive bid** δ_{t2} for its provisionally granted bid $prefer(T_0, B_{t2,0})$. $bidder(\delta_{t2}) = bidder(prefer(T_0, B_{t2,0}))$. Again, there are four situations from step 5 as described above. Set $t1 \leftarrow t2$ and select the situation (i), (ii), (iii) or (iv) as appropriate.

Situation (iii): The auctioneer will send the bidder *bidder*(δ_{t1}) a **confirm grant** for its definitive bid δ_{t1}, and as mentioned above, all other bidders $\Gamma\backslash\{bidder(\delta_{t1})\}$ which are still at step 3 will receive a **confirm reject** for their unsuccessful pre-bids $B_{t2,0}\backslash\{\delta_{t1}\}$. The auctioneer now has the bid $\delta_{t1} = prefer(T_0, B_{t2,0})$ allocated for the set of tasks T_0, where $\delta_{t1} \neq prefer(T_0, B_{t1,0})$.

Situation (iv): The auctioneer sends the bidder *bidder*(δ_{t1}) a **provisionally reject** message for its definitive bid δ_{t1}, resulting in the protocol proceeding to step 2 for bidder *bidder*(δ_{t1}) to submit an updated **pre-bid** if it desires. The auctioneer α who is at step 3 with another bidder $bidder(prefer(T_0, B_{t2,0})) \neq bidder(\delta_{t1})$ sends this bidder a **provisional grant** for its now preferred bid $prefer(T_0, B_{t2,0})$. The protocol proceeds to step 4, the bidder *bidder(prefer(T_0, $B_{t2,0}$))* submits a **definitive bid** δ_{t2} for its provisionally granted bid *prefer(T_0, $B_{t2,0}$)*. *bidder(δ_{t2}) = bidder(prefer(T_0, $B_{t2,0}$))*. Again, there are four situations from step 5 as described above. Set $t1 \leftarrow t2$ and select the situation (i), (ii), (iii) or (iv) as appropriate.

Task allocation with PAP:

Refer to the PAP specification in section 5.1 (Figure 21), and Definition 3. The auctioneer α commences at step 1 by **announcing** the set of tasks T_0, and the protocol proceeds to step 2. At time *t1*, the bidders Γ submit **bids** $B_{t1,0}$ in order to achieve T_0 such that they are full bids, i.e. $\forall k\ [b_{k,0} \in B_{t1,0} \Rightarrow diff(T_0, b_{k,0}) = \emptyset]$, ignoring bidders that do not submit any bids (**no communication**). The protocol proceeds to step 3. Since we do not consider task withdrawals, rejection on bid submission and backtracking, in the context of the theorem, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. The auctioneer α selects its most preferred bid, which is *prefer(T_0, $B_{t1,0}$)* and sends the bidder *bidder(prefer(T_0, $B_{t1,0}$))* a **provisional grant**. The protocol proceeds to step 4 for α and *bidder(prefer(T_0, $B_{t1,0}$))*. Since bidders are not committed to their bids, the auctioneer does not need to reject the other bids $B_{t1,0}\backslash\{prefer(T_0, B_{t1,0})\}$ by bidders $\Gamma\backslash\{bidder(prefer(T_0, B_{t1,0}))\}$. As a result of PAP's bidding policy, bidders may submit *updated better bids* at time $\tau > t1$ if they have an opportunity to submit a better bid than a previously submitted bid. Since we do not consider provisionally withdrawn bids (or can assume it is equivalent to withdrawn bids, indicating that a bid is not available), in the context of the theorem, this speech act can be ignored at step 4.

At steps 4 and 5, there are three situations (i), (ii), and (iii) that are possible, which are analogous to those situations (i), (ii), and both (iii) and (iv) together, respectively, described above for CNP-ext: **(i)** At step 4 the bidder *bidder(prefer(T_0, $B_{t1,0}$))* sends a **provisional grant accepted** to commit to its bid *prefer(T_0, $B_{t1,0}$)*. The protocol proceeds to step 5. In this situation, *no other* bidder $\gamma \in \Gamma$ where $\gamma \neq bidder(prefer(T_0, B_{t1,0}))$ has sent an *updated better bid* by time $t2 > t1$ that is better than the provisionally granted bid *prefer(T_0, $B_{t1,0}$)*, i.e. $prefer(T_0, B_{t1,0}) = prefer(T_0, B_{t2,0})$, and $prefer(T_0, B_{t1,0}) \in B_{t2,0}$. The condition in the theorem for provisionally rejecting the provisionally granted bid

$prefer(T_0, B_{t1,0})$ is not satisfied; **(ii)** At step 4 the bidder *$bidder(prefer(T_0, B_{t1,0}))$* sends a **provisional grant accepted** to commit to its bid *$prefer(T_0, B_{t1,0})$*. The protocol proceeds to step 5. In this situation, some other bidder $\gamma \in \Gamma$ where $\gamma \neq$ *$bidder(prefer(T_0, B_{t1,0}))$* submits an *updated better bid* by time *$t2 > t1$* that is better than *$prefer(T_0, B_{t1,0})$*, i.e. *$prefer(T_0, B_{t1,0}) \neq prefer(T_0, B_{t2,0})$*, and *$prefer(T_0, B_{t1,0}) \in B_{t2,0}$*. The condition for provisionally rejecting the provisionally granted bid *$prefer(T_0, B_{t1,0})$* is satisfied; **(iii)** At step 4 the bidder *$bidder(prefer(T_0, B_{t1,0}))$* sends a **withdrawn** message to withdraw the bid *$prefer(T_0, B_{t1,0})$*. Therefore, *$prefer(T_0, B_{t1,0}) \notin B_{t2,0}$* at time *$t2 > t1$* since *$prefer(T_0, B_{t1,0})$* is withdrawn.

Situation (i): The auctioneer sends the bidder *$bidder(prefer(T_0, B_{t1,0})) = bidder(prefer(T_0, B_{t2,0}))$* a confirm grant for its bid *$prefer(T_0, B_{t1,0}) = prefer(T_0, B_{t2,0})$*. Since bidders are not committed to their bids, the auctioneer does not need to reject the other bids $B_{t2,0}\backslash\{prefer(T_0, B_{t2,0})\}$ by bidders $\Gamma\backslash\{bidder(prefer(T_0, B_{t2,0}))\}$. The auctioneer now has bid *$prefer(T_0, B_{t2,0})$* allocated for the set of tasks *T_0*.

Situation (ii): The auctioneer sends the bidder *$bidder(prefer(T_0, B_{t1,0}))$* a **provisionally reject** message for its bid *$prefer(T_0, B_{t1,0})$*. The protocol proceeds to step 2, allowing the bidder *$bidder(prefer(T_0, B_{t1,0}))$* to submit an updated **bid** to replace its rejected bid *$prefer(T_0, B_{t1,0})$*. The protocol proceeds to step 3. Again, in the context of the theorem, withdrawn, reject and backtracking speech acts can be ignored. The auctioneer α sends the bidder *$bidder(prefer(T_0, B_{t2,0}))$* a **provisional grant** for its now preferred bid *$prefer(T_0, B_{t2,0})$*. The protocol proceeds to step 4, and again, there are three situations from step 4 as described above. Set *$t1 \leftarrow t2$* and select the situation (i), (ii) or (iii) as appropriate.

Situation (iii): The protocol proceeds to step 2, allowing the bidder *$bidder(prefer(T_0, B_{t1,0}))$* to send an updated **bid** to replace the withdrawn bid *$prefer(T_0, B_{t1,0})$*. The protocol proceeds to step 3. Again, in the context of the theorem, withdrawn, reject and backtracking speech acts can be ignored. The auctioneer α selects its most preferred bid, which is *$prefer(T_0, B_{t2,0})$* at time *$t2 > t1$* and sends bidder *$bidder(prefer(T_0, B_{t2,0}))$* a **provisional grant**. The protocol proceeds to step 4, and again, there are three situations from step 4 as described above. Set *$t1 \leftarrow t2$* and select the situation (i), (ii) or (iii) as appropriate. As mentioned above, bidder *bidder(prefer*(T_0, $B_{t1,0}$)) that withdrew its bid *prefer*(T_0, $B_{t1,0}$) sends an updated bid, which we will refer to as *u*, where *bidder(u)* = *bidder(prefer*(T_0, $B_{t1,0}$)). We have:

(iii-a) the updated bid *u* is the most preferred bid, i.e. *u* = *prefer*(T_0, $B_{t2,0}$). *u* is then confirm granted at situation (i) after it is provisionally granted. We then have the same situation as situation (iii) in CNP-ext where a definitive bid that is submitted is different to the provisionally granted bid (which is effectively withdrawing the provisionally granted bid *prefer*(T_0, $B_{t1,0}$) and submitting a new preferred bid *u* = *prefer*(T_0, $B_{t2,0}$)) that is still better than other submitted bids at time *t2*. Thus this definitive bid *u* is confirm granted.

(iii-b) In all other cases, which are: **(1)** The updated bid *u* is the preferred bid *u* = *prefer*(T_0, $B_{\tau,0}$)) at time $t1 < \tau < t2$ and provisionally granted, but is not confirm granted due to situation (ii) where another *updated better bid* is submitted that is preferred, i.e. $u \neq$ *prefer*(T_0, $B_{t2,0}$)). We then have situation (iv) in CNP-ext, where a definitive bid submitted is different to the provisionally granted bid (which is effectively withdrawing the provisionally granted bid *prefer*(T_0, $B_{t1,0}$) and submitting a new bid $u \neq$ *prefer*(T_0, $B_{t2,0}$)) which is not preferred at time *t2*. Thus *u* is provisionally rejected and *prefer*(T_0, $B_{t2,0}$) by bidder *bidder(prefer*(T_0, $B_{t2,0}$)) is provisionally granted; **(2)** The updated bid *u* is the preferred bid *u* = *prefer*(T_0, $B_{\tau,0}$)) at time $t1 < \tau < t2$ and provisionally granted, but is not confirm granted due to situation (iii) where it is withdrawn again. In the context of the theorem, we ignore this situation because CNP-ext does not allow the situation where a bid is withdrawn more than once consecutively. CNP-ext only allows "withdrawn" bids when a bidder submits a different definitive bid to that which is provisionally granted (which is effectively withdrawing the provisionally granted bid *prefer*(T_0, $B_{t1,0}$) and submitting a new *definitive* bid *u*). Since agents are committed to the updated definitive bid *u*, they cannot withdraw from it; **(3)** The updated bid *u* is the not the most preferred bid $u \neq$ *prefer*(T_0, $B_{t2,0}$), then we have situation (iv) in CNP-ext, where a definitive bid submitted is different to the provisionally granted bid (which is effectively withdrawing the provisionally granted bid *prefer*(T_0, $B_{t1,0}$) and submitting a new bid *u*

$\neq$ *prefer*$(T_0, B_{t2,0})$) which is not preferred at time *t2*, and thus is *u* is not provisionally granted. Rather, *prefer*$(T_0, B_{t2,0})$ by bidder *bidder*(*prefer*$(T_0, B_{t2,0})$) is provisionally granted.

<u>Final allocation for both protocols</u>

Both CNP-ext and PAP produce the same result from the same situations:

- For situation (i) for both CNP-ext and PAP, the bid *prefer*$(T_0, B_{t1,0})$ is provisionally granted at time *t1*, the same bid is confirm granted at time *t2*, such that the final allocation is *prefer*$(T_0, B_{t1,0})$ = *prefer*$(T_0, B_{t2,0})$.

- For situation (iii) for CNP-ext and situation (iii-a) for PAP, after the bid *prefer*$(T_0, B_{t1,0})$ is provisionally granted at time *t1*, it is withdrawn with a new bid *b* submitted, where *b* is the preferred bid at time *t2*. For CNP-ext, *b* is the definitive bid submitted that is different to *prefer*$(T_0, B_{t1,0})$. For PAP, *b* is the updated bid send after the bidder withdraws the bid *prefer*$(T_0, B_{t1,0})$. Thus *b* is confirm granted, and the final allocation is *b* = *prefer*$(T_0, B_{t2,0}) \neq$ *prefer*$(T_0, B_{t1,0})$, and *bidder*(*b*) = *bidder*(*prefer*$(T_0, B_{t1,0})$) = *bidder*(*prefer*$(T_0, B_{t2,0})$).

- For situation (ii) for both CNP-ext and PAP, the after a bid *prefer*$(T_0, B_{t1,0})$ is provisionally granted at time *t1*, another bid is preferred at time *t2* before *prefer*$(T_0, B_{t1,0})$ is confirm granted. So both protocols provisionally reject the bid *prefer*$(T_0, B_{t1,0})$ and provisionally grant a new bid *prefer*$(T_0, B_{t2,0})$. The protocol process continues until the situations corresponding to the first two dot points above arise and an allocation results.

- For situation (iv) for CNP-ext and situation (iii-b) for PAP, after the bid *prefer*$(T_0, B_{t1,0})$ is provisionally granted at time *t1*, it is withdrawn with a new bid *b* submitted where *b* is not the preferred bid *prefer*$(T_0, B_{t2,0})$ at time *t2*. For CNP-ext, *b* is the definitive bid submitted that is different to *prefer*$(T_0, B_{t1,0})$. For PAP, *b* is the updated bid send after the bidder withdraws the bid *prefer*$(T_0, B_{t1,0})$). Therefore, the preferred bid *prefer*$(T_0, B_{t2,0}) \neq b$ by bidder *bidder*(*prefer*$(T_0, B_{t2,0})$) is provisionally granted. The protocol process continues until the situations corresponding to the first two dot points above arise and an allocation results.

Therefore, both CNP-ext and PAP produce the same allocation, as indicated by the first two dot points above, for the set of tasks T_0.

Q.E.D.

We have shown that PAP can solve planning problems that CNP, ECNP and CNP-ext are able to. The example in Definition 5 is used to show that CNP, ECNP and CNP-ext may not be able to solve planning problems that PAP is able to.

Definition 5: An agent system consisting of a set of agents $\mathscr{A}$ to solve a planning problem comprises an auctioneer $\alpha \in \mathscr{A}$ and a set of bidders $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\} \subseteq \mathscr{A} \setminus \{\alpha\}$. The auctioneer α requires a set of tasks T_0 achieved, such that T_0 has two possible plans: $T_0 = \{t_1, t_2, t_3, t_4\}$ or $T_0 = \{t_5, t_6\}$. The index $j > 0$ is used to denote new sets of tasks that are created in achieving T_0 as the protocol (planning) proceeds. The three bidders γ_1, γ_2 and γ_3 are able to perform $b_1 = \{t_1, t_3\}$, $b_2 = \{t_2, t_4\}$ and $b_3 = \{t_5\}$, respectively, for the auctioneer's announced set of tasks T_0. Therefore, only the plan $T_0 = \{t_1, t_2, t_3, t_4\}$ can be fully achieved with the available bids. The auctioneer's preference among a set of submitted bids B_j for T_j is denoted $prefer(T_j, B_j)$. In this example $prefer(T_0, \{b_1, b_2, b_3\}) = b_3$ and $prefer(T_0, \{b_1, b_2\}) = b_1$. $diff(T_j, b_i)$ is a function that returns a set of tasks T_k where $k > j$ which is the remaining set of tasks that the bid b_i does not achieve from the set of tasks T_j. If $T_k = \varnothing$, then the bid b_i fully achieves the set of tasks T_j.

Lemma 1: PAP is able to solve the planning problem specified in Definition 5, if:

- $\mathscr{A}$ comprises a single auctioneer
- PAP precludes withdrawals of bids and tasks (a static environment) and the rejection of bids on submission, and
- the final plan found is suitable.

Proof:

Refer to the PAP specification in section 5.1 (Figure 21). PAP commences at step 1 where the auctioneer α **announces** the set of tasks T_0 and proceeds to step 2. The bidders γ_1, γ_2 and γ_3 submit **bids** b_1, b_2 and b_3 to partially achieve the set of tasks T_0. The protocol proceeds to step 3. Since we do not consider task withdrawals and rejection on bid submission, in the context of the lemma, these speech acts (e.g. withdrawn and provisional reject) can be ignored at step 3. Since bids were submitted for the announced set of tasks T_0, backtracking is not required. Therefore, the auctioneer α selects its preferred bid, which is $prefer(T_0, \{b_1, b_2, b_3\}) = b_3$, and sends bidder γ_3 a **provisional grant** for its bid b_3. The protocol proceeds to step 4 for the auctioneer α and bidder γ_3 – the other bidders γ_1 and γ_2 remain at step 3 in the negotiations with the auctioneer α. Since we do not consider bid withdrawals, in the context of the lemma, these speech acts (e.g. withdrawn and provisionally withdrawn) can be ignored at step 4. The bidder γ_3 **accepts the provisional grant**. Since the set of tasks T_0 has not been completely achieved, the protocol takes control option (e) where the remaining set of tasks left to achieve T_1 is **re-announced** back at step 1 in a new protocol process, where $T_1 = diff(T_0, b_3) = diff(\{t_5, t_6\}, \{t_5\}) = \{t_6\}$. The protocol proceeds to step 2, where no bidders are able to fully or partially perform the set of tasks T_1, and therefore all bidders submit nothing – **no communication**. The protocol process for this set of tasks T_1 therefore ends for the bidders (control option (a)). The protocol proceeds to step 3 for the auctioneer (control option (b)) and since no bids are available for the set of tasks T_1, the auctioneer **backtracks**, and thus proceeds to step 5 of the protocol process for the set of tasks T_0, and provisionally rejects b_3 for T_0. The protocol proceeds to step 2, allowing the bidder γ_3 to submit an updated bid for it rejected bid b_3, where in the context of the lemma, bidder γ_3 is unable to (b_3 is its only bid). The protocol process for T_0 ends for the bidder γ_3 – **no communication**, control option (a).

The auctioneer α is still at step 3 with bidders γ_1 and γ_2, with bids b_1 and b_2, respectively. Again, since we do not consider task withdrawals and rejection on bid submission, in the context of the lemma, these speech acts (e.g. withdrawn and provisional reject) can be ignored at step 3. Since bids remain for the announced set of tasks T_0, backtracking is not required. Therefore, the auctioneer α selects its preferred bid, which is $prefer(T_0, \{b_1, b_2\}) = b_1$, and sends bidder γ_1 a **provisional grant** for its bid b_1. The protocol proceeds to step 4 for the auctioneer α and bidder γ_1. Again, since we do not consider bid withdrawals, in the context of the lemma, these speech acts (e.g. withdrawn and provisionally withdrawn) can be ignored at step 4. The bidder γ_1 **accepts the provisional grant**. Since the set of tasks T_0 has not been completely achieved, we task protocol control option (e) where the remaining set of tasks left to achieve T_2 is **re-announced** back at step 1 in a new protocol process, where $T_2 = diff(T_0, b_1) = diff(\{t_1, t_2, t_3, t_4\}, \{t_1, t_3\}) = \{t_2, t_4\}$. The protocol proceeds to step 2 where the bidder γ_2 submits **bid** b_2 for T_2, and the other bidders submit nothing – **no communication**, and task control option (a) to exit the protocol process for the set of tasks T_2. The protocol proceeds to step 3. Again, since we do not consider task withdrawals and rejection on bid submission, in the context of the lemma, these speech acts (e.g. withdrawn and provisional reject) can be ignored at step 3. Since bids were submitted for the announced set of tasks T_2, backtracking is not required. Therefore, the auctioneer α selects its preferred bid, which is $prefer(T_2, \{b_2\}) = b_2$, and sends bidder γ_2 a **provisional grant** for its bid b_2. The protocol proceeds to step 4 for the auctioneer α and bidder γ_2. Again, since we do not consider bid withdrawals, in the context of the lemma, these speech acts (e.g. withdrawn and provisionally withdrawn) can be ignored at step 4. The bidder γ_2 **accepts the provisional grant**.

At this stage, the set of tasks T_2, and thus T_0, has been completely achieved, i.e. $diff(T_2, b_2) = diff(\{t_2, t_4\}, \{t_2, t_4\}) = \varnothing$. Therefore, the protocol proceeds to step 5 of the protocol process for T_2. In the context of the lemma, the final solution found is suitable, and therefore, the provisional reject speech act at step 5 can be ignored. The auctioneer α can now **confirm grant** the provisionally granted bid b_2 by bidder γ_2 for the set of tasks T_2. The protocol process then proceeds to step 5 of the previous protocol process for T_0, where the auctioneer α **confirm grants** the provisionally granted bid b_1 by bidder γ_1 for the set of tasks T_0. Since there are no more previous protocol processes, the protocol exits, and a solution is found.

Therefore, using PAP, we have found a solution to the planning and task allocation problem specified in Definition 5. The final plan for the set of tasks is $T_0 = \{t_1, t_2, t_3, t_4\}$, with the plan of bids to achieve T_0 is $b_1 = \{t_1, t_3\}$ and $b_2 = \{t_2, t_4\}$, and thus the final allocation of bids b_1 and b_2 results.

Q.E.D.

Theorem 4: There exist planning problems that PAP is able to address which CNP and CNP-ext cannot.

Proof:

Refer to the CNP and the CNP-ext specification in section 4.2, and Definition 5. At step 1, the auctioneer α announces a set of tasks T_0, and the protocols proceed to step 2. At step 2 of both protocols, bidders must submit bids that fully achieve the set of tasks T_0, which none of the bidders γ_1, γ_2 and γ_3 are able to do. Therefore, the protocol exits with no task allocation for the set of tasks T_0.

According to Lemma 1, PAP is able to solve the planning (and task allocation) problem specified in Definition 5.

Therefore PAP is able to address planning problems that CNP and CNP-ext are not.

Note that in this proof, we have not considered examples of planning problems that require:

1. Negotiations with multiple auctioneers simultaneously, withdrawals of tasks and bids, the updating of bids, rejection of bids and backtracking, which CNP does not allow and PAP does.
2. Withdrawal of tasks, rejection of bids on submission, provisionally withdrawn bids, backtracking, and more than one consecutive withdrawal of bids by bidders, which CNP-ext does not allow and PAP does.

Q.E.D.

Theorem 5: There exists planning problems that PAP is able to address which ECNP cannot.

Proof:

Refer to the ECNP specification in section 4.7 (Figure 18), and Definition 5. At step 1, the auctioneer α announces a set of tasks T_0, and the protocols proceed to step 2. The bidders γ_1, γ_2 and γ_3 submit **bids** b_1, b_2 and b_3 to partially achieve the set of tasks T_0. The protocol proceeds to step 3. The auctioneer α selects its preferred bid, which is $prefer(T_0, \{b_1, b_2, b_3\}) = b_3$, and sends bidder γ_3 a **provisional grant** for its bid b_3. The other bidders γ_1 and γ_2 receive a **provisional reject** for their bids b_1 and b_2, respectively. Since the set of tasks T_0 has not been completely achieved, the protocol takes control option (a) where the remaining set of tasks left to achieve T_1 is **re-announced** back at step 1 in a new protocol process, where $T_1 = diff(T_0, b_3) = diff(\{t_5, t_6\}, \{t_5\}) = \{t_6\}$. The protocol proceeds to step 2, where no bidders are able to fully or partially perform the set of tasks T_1. As ECNP does not allow backtracking at any stage in the planning (it only allows backtracking when a solution/plan is found and is deemed unsuitable) and expects a bid at step 2 in the protocol, ECNP is unable to continue with planning. Therefore, ECNP does not find a plan and task allocation for the set of tasks T_0.

According to Lemma 1, PAP is able to solve the planning (and task allocation) problem specified in Definition 5.

Therefore PAP is able to address planning problems that ECNP is not.

Note that in this proof, we have not considered examples of planning problems that require negotiations with multiple auctioneers simultaneously, withdrawals of tasks and bids, the updating of bids, and the rejection of bids on submission, which ECNP does not allow and PAP does.

Q.E.D.

5.3.4 Decentralised Depth-First Search with Dynamic Search Tree

PAP allows auctioneers to perform a search equivalent to a centralised depth first search, i.e. selects the same bids/branches and passes through the same tasks/nodes, where the search tree (the bids/branches available with each task/node) may change over time, except that PAP is able to perform it in a decentralised (distributed) fashion.

Definition 6: An agent system consisting of a set of agents $\mathscr{A}$ to solve a planning problem comprises an auctioneer $\alpha \in \mathscr{A}$ and a set of bidders $\Gamma \subseteq \mathscr{A} \setminus \{\alpha\}$. The auctioneer comprises a set of tasks T_j, where T_0 is its initial set of tasks that it would like to achieve. The index $j > 0$ is used to denote new sets of tasks that are created in achieving T_0 as the protocol (planning) proceeds. With the centralised planning case, the auctioneer α at time τ has available a set of n suitable bids $B^c_{\tau,j} = \{b_1, \ldots, b_n\}$ that it may use in order to achieve a set of tasks T_j. With the decentralised case, the auctioneer α at time τ has the and same set of bids for T_j, except the bids are distributed among p bidders

$\Gamma_\tau = \{\gamma_1, \ldots, \gamma_p\}$, i.e. bidder γ_i at time τ contains a set of suitable bids $B_{\tau,j,i} = \{b_{1,i}, \ldots, b_{q,i}\}$, $q \leq n$, for the set of tasks T_j, where $B_{\tau,j,1} \cup B_{\tau,j,2} \cup \ldots \cup B_{\tau,j,p} = B^c_{\tau,j}$. At time τ, the auctioneer α using PAP (decentralised case) contains a set of submitted bids $B^s_{\tau,j}$ by bidders $\Gamma_{\tau'}$, $\tau' \leq \tau$ for T_j, where $B^s_{\tau,a}$ excludes any previously withdrawn or rejected bids (the auctioneer α discards these bids in our current implementation). *task_achieved*(T_j, b_i) is a predicate that is true if bid b_i achieves the set of tasks T_j. *diff*(T_j, b_i) is a function that returns a set of tasks T_k where $k > j$, which is the remaining set of tasks that the bid b_i does achieve from the set of tasks T_j. *plan_suitable* is a predicate that is false if the current plan (selected bids) for T_0 is not suitable (infeasible or bad solution), and hence backtracking is required. During the time that a bid b_i is selected/granted for a set of tasks T_j, it cannot be withdrawn, i.e. if $b_i \in B_{\tau 1,j}$ was selected/granted at time $\tau 1$ for T_j, then $\forall y[b_i \in B_{y,j} \ \& \ \tau 1 \leq i \leq \tau 2]$ where $\tau 2$ is the time that b_i is either rejected or executed to perform all or part of T_j. We assume that the bidding deadline in PAP is set large enough such that all bidding in the current bidding phase (step 2 of PAP) between bidders Γ_τ and the auctioneer α are complete before PAP continues. Withdrawn bids and provisionally withdrawn bids are assumed equivalent, as in our current implementations. If bidders enter the system at time τ^e, we assume they acquire a list of all the previously announced tasks (if any) at time $\tau' < \tau^e$.

Definition 7: *order_bids*(T_j, B_k) is a time independent function that will return a set of bids $\{b_1, \ldots, b_m\}$ from the set of bids B_k for the set of tasks T_j that are ordered from best to worse, which we denote $b_1 \geq b_2 \geq \ldots \geq b_m$, where b_1 is the most preferred bid and b_m is the least preferred bid for the set of tasks T_j. $B^r_{\tau,j,k}$ is the set of rejected bids for bidder γ_k for the set of tasks T_j at time τ, such that $B^r_{\tau,j,k} \subseteq \bigcup_{\forall i \leq \tau} B_{i,j,k}$, and these bids are excluded from the ordered list of bids of $B_{\tau,j,k}$ for T_j, i.e. *order_bids*$(T_j, B_{\tau,j,k})$ is equivalent to *order_bids*$(T_j, B_{\tau,j,k} \backslash B^r_{\tau,j,k})$, but we will use the former notation. Similarly, $B^{wr}_{\tau,j}$ is the set of rejected or withdrawn bids by auctioneer α for the set of tasks T_j at time τ, such that $B^{wr}_{\tau,j} \subseteq \bigcup_{\forall i \leq \tau} B_{i,j,k}$, and these bids are excluded from the ordered list of bids of $B^s_{\tau,j}$ for T_j, i.e. *order_bids*$(T_j, B^s_{\tau,j})$ is equivalent to *order_bids*$(T_j, B^s_{\tau,j} \backslash B^{wr}_{\tau,j})$, but we will use the former notation. We assume the n bids in $B^c_{\tau,j}$, and thus in $B_{\tau,j,k}$ and $B^s_{\tau,j}$, all have unique preference values for any set of tasks T_j, i.e. $b_1 > b_2 > \ldots > b_n$, so there is always only one preferred bid for T_j at any time. The use of the function *order_bids* by the centralised auctioneer list of bids $B^c_{\tau,j}$ will return *only a suitable and available* ordered list of bids for T_j as the auctioneer has all information regarding bid suitability (e.g. selected bids and conflicts with other bids) and availability (e.g. if they are withdrawn). Therefore, the function *order_bids* used on a set of bidder's bids $B_{\tau,j,k}$ will return a list of bids in the correct order, but may include "unsuitable" bids as additions, as they may be missing bid suitability information that the auctioneer possesses [17]. Unsuitable bid b is a bid such that $b \notin order\_bids(T_j, B^c_{\tau,j})$ & $b \in order\_bids(T_j, B_{\tau,j,k})$. Additionally, the function *order_bids* used on a set of auctioneer's bids $B^s_{\tau,j}$ may include additional bids which are no longer available as the auctioneer does not have bidder information regarding the availability of the bids (due to the persistence policy). Unavailable bid b is a bid such that $b \notin order\_bids(T_j, B^c_{\tau,j})$ & $b \in order\_bids(T_j, B^s_{\tau,j})$.

Lemma 2: With PAP, if a bidder γ_k contains a set of bids at time τ for an auctioneer α's set of tasks T_j such that γ_k's preferred bids, i.e. bid(s) on top of the ordered list $order\_bids(T_j, B_{\tau,j,k})$, are unsuitable, then the bidder γ_k will eventually submit either its best suitable bid $\boldsymbol{b}$ to the auctioneer α for T_j or submit no suitable bid.

Proof:

Refer to the PAP specification in section 5.1 (Figure 21), and Definition 6 and Definition 7. In the context of the lemma, at time τ the bidder γ_k contains an ordered set of bids $order\_bids(T_j, B_{\tau,j,k}) = \{b_{1,k}, b_{2,k}, \ldots, b_{q,k}, \boldsymbol{b}, b_{q+2,k}, \ldots, b_{u,k}\}$ for an auctioneer α's set of tasks T_j, where bids $b_{1,k}, \ldots, b_{q,k}$, $q \geq 1$, are unsuitable, and $u \geq q$ where if $u = q$, no suitable bid $\boldsymbol{b}$ exists, and if $u > q$, then bid $\boldsymbol{b}$ is the best *suitable* bid for T_j. In the bidding phase for a set of tasks T_j for auctioneer α, each bidder γ_k must submit their best bid (if the task is initially announced) or next best bid (if arrived at step 2 after a bid was withdrawn or rejected the in following steps) at step 2 of PAP. According to the PAP specification, the bidder γ_k will submit its best bid for T_j, which is the bid at the top of the ordered list of

[17] Ideally, the auctioneer should provide this information with the bid evaluation function which is used to determine the list *order_bids*, but this may not always be practicable.

bids for T_j, and hence γ_k submits bid $b_{1,k}$ at step 2 of the protocol, and the protocol proceeds to step 3. Since the bid $b_{1,k}$ is unsuitable, according to the PAP specification the auctioneer α will provisionally reject the bid at step 3, resulting in the bidder proceeding back to step 2 to submit an updated bid to replace the rejected bid. The rejected bid is added to the list of rejected bids, $b_{1,k} \in B^r_{\tau,j,k}$, and thus from Definition 7, the ordered list of bids for T_j now becomes $order\_bids(T_j, B_{\tau',j,k}) = \{b_{2,k}, ..., b_{q,k}, \boldsymbol{b}, b_{q+2,k}, ..., b_{u,k}\}$, $\tau' > \tau$, which excludes the rejected bid $b_{1,k}$. The bidder submits its next best bid, $b_{2,k}$ at step 2, and again the auctioneer α will provisionally reject the bid at step 3, etc., until the bidder has bid $b_{q,k}$ provisionally rejected. At step 2, if no suitable bid exists, then the bidder will submit nothing (no communication), and thus no suitable bid would have been submitted for T_j by bidder γ_k. If a suitable bid $\boldsymbol{b}$ exists, then the bidder γ_k submits bid $\boldsymbol{b}$ at step 2 of the protocol. As the bid is now suitable, the auctioneer will not provisionally reject the bid at step 3, and will keep the bid as a possible option to achieve the set of tasks T_j. Therefore, bidder γ_k will submit the bid $\boldsymbol{b}$, or no suitable bid, to the auctioneer α.

Q.E.D.

Lemma 3: If an auctioneer α using PAP has a set of bids B^s_τ at time τ submitted for the set of tasks T_j, and its preferred bid(s) in this list, i.e. top most bid(s) in the list $order\_bids(T_j, B^s_{\tau,j})$, are withdrawn (no longer available), then the auctioneer will eventually arrive at either its most preferred and available bid $\boldsymbol{b^{pa}}$ or no available bids for T_j.

Proof:

Refer to the PAP specification in section 5.1 (Figure 21), Definition 6 and Definition 7. In the context of the lemma, the auctioneer α using PAP has a set of bids $B^s_{\tau,j}$ submitted from bidders $\Gamma_{\tau'}$, $\tau' \leq \tau$, for the set of tasks T_j, such that the order of submitted bids are $order\_bids(T_j, B^s_{\tau,j}) = \{b_1, ..., b_q, b^a, b_{q+2}, ..., b_u\}$, where bids $b_1, ..., b_q$ are withdrawn (no longer available), and for $u \geq q$, if $u = q$, then there are no available bids in the list, and if $u > q$, then b^a is the next available bid in the list. The most preferred and available bid for T_j is either $\boldsymbol{b^{pa}}$, i.e. $order\_bids(T_j, B^c_{\tau,j}) = \{ \boldsymbol{b^{pa}}, b_2, b_3, ...\}$, or there are no available bids for T_j, i.e. $order\_bids(T_j, B^c_{\tau,j}) = \varnothing$. There are three cases, **(i)** $b^a = \boldsymbol{b^{pa}}$ or **(ii)** $b^a \neq \boldsymbol{b^{pa}}$ or b^a

does not exists but $\boldsymbol{b^{pa}}$ does or **(iii)** $\boldsymbol{b^{pa}}$ does not exists, and hence b^a does not exist. With the set of bids $B^s{}_{\tau,j}$ at step 3 of PAP, in the context of the lemma, the auctioneer α will attempt to provisionally grant a bid for T_j (the bidding deadline has passed, and all bids in the bidding phase are received, and hence withdrawn and provisionally reject speech acts can be ignored, and in the context of the lemma, $B^s{}_{\tau,j} \neq \varnothing$, so backtracking event can be ignored). The auctioneer α will attempt to provisionally grant its preferred bid at step 3 of PAP, which is b_1, and proceeds to step 4. Since the bid b_1 is withdrawn, then the bidder associated with the bid (say γ_k) will send a withdrawn message, and the protocol proceeds to step 2 for the bidder γ_k to send an updated bid, which γ_k may not (no communication) or may submit its next best *suitable* bid b^{sb} (from Lemma 2), and the protocol proceeds to step 3.

Case (i): When $b^a = \boldsymbol{b^{pa}}$ is the most preferred bid, then there are no available bids that are better than b^a, and therefore $b^a > b^{sb}$. Also, the withdrawn bid is added to the list of rejected and withdrawn bids, $b_1 \in B^{wr}{}_{\tau,j}$, and thus from Definition 7, the ordered list of bids for T_j now becomes $order\_bids(T_j, B^s{}_{\tau',j}) = \{b_2, \ldots, b_q, b^a, b_{q+2}, \ldots, b^{sb}, \ldots, b_u\}$, if an updated bid b^{sb} is sent or $order\_bids(T_j, Bs_{\tau',j}) = \{b_2, \ldots, b_q, b^a, b_{q+2}, \ldots, b_u\}$ if no updated bid b^{sb} sent, for $\tau' > \tau$, which excludes the (provisionally) withdrawn bid b_1. The auctioneer α will attempt to provisionally grant its next preferred bid, which is b_2, and the same process as described above will occur until b_q is withdrawn, and the auctioneer will be at step 3 with the list of bids $order\_bids(T_j, Bs_{\tau,j}) = \{ b^a, b_{q+2}, \ldots, b_u\}$. Therefore, the auctioneer α has arrived at its most preferred bid and available bid $b^a = \boldsymbol{b^{pa}}$, which is at the top of the list of ordered bids for the set of tasks T_j, ready for the auctioneer to provisionally grant.

Case (ii): When $b^a \neq \boldsymbol{b^{pa}}$ or b^a does not exist and $\boldsymbol{b^{pa}}$ does, then the most preferred and available bid $\boldsymbol{b^{pa}}$ has not been submitted by a bidder γ_z because at time $ts < \tau$, there was another bid which was better than $\boldsymbol{b^{pa}}$, i.e. $order\_bids(T_j, B_{ts,j,z}) = \{b_{1,z}, \ldots, b_{q,z}, \boldsymbol{b^{pa}}, b_{q+2,z}, \ldots, b_{u,z}\}$, or $\boldsymbol{b^{pa}}$ did not exist then, so $order\_bids(T_j, B_{ts,j,z}) = \{b_{1,z}, \ldots, b_{u,z}\}$ (still, $b_{1,z} > \boldsymbol{b^{pa}}$, because if $\boldsymbol{b^{pa}} > b_{1,z}$, when $\boldsymbol{b^{pa}}$ arrived in the set of possible bids $B_{ta,j,z}$, $ts < ta \leq \tau$, from PAP's bidding policy, $\boldsymbol{b^{pa}}$ would be submitted to α as an updated better bid, and thus we would have case (i) above). The bid $b_{1,z}$ would have been submitted for T_j at time ts, and now the bid $b_{1,z}$ is withdrawn, so bidder γ_z now has $order\_bids(T_j, B_{\tau,j,z}) =\{ \boldsymbol{b^{pa}}, b_{2,z}, ..., b_{u,z}\}$. Since $b_{1,z} > \boldsymbol{b^{pa}}$ and $\boldsymbol{b^{pa}} > b^a$ (if b^a exists), then $\exists y[b_y \in order\_bids(T_j, B^s{}_{\tau,j})\ \&\ b_y > b^a\ \&\ b_y = b_{1,z}]$ if b^a exists, or $\exists y[b_y \in order\_bids(T_j, B^s{}_{\tau,j})\ \&\ b_y = b_{1,z}]$ if b^a does not exist. Therefore, one of the bids $\{b_1, ..., b_q\}$ in $order\_bids(T_j, B^s{}_{\tau,j})$ described above is from the bidder γ_z which submitted the withdrawn bid $b_{1,z}$ and has bid $\boldsymbol{b^{pa}}$. So, continuing from above (step 3 and receiving an updated bid b^{sb} or not), if the updated bid b^{sb} is submitted and is from a bidder $\gamma_j \neq \gamma_z$, then since $\boldsymbol{b^{pa}}$ is the most preferred bid, $\boldsymbol{b^{pa}} > b^{sb}$, and since $b_{1,z} > \boldsymbol{b^{pa}}$, then $b_{1,z} > b^{sb}$, and therefore we will arrive at the withdrawn bid $b_{1,z}$ before the available updated bid b^{sb}. If no updated bid b^{sb} is submitted, then we will still arrive at the withdrawn bid $b_{1,j}$ before an available bid (if one exists). Also, the withdrawn bid is added to the list of rejected and withdrawn bids, $b_1 \in B^{wr}{}_{\tau,j}$, and thus from Definition 7, the ordered list of bids for T_j now becomes $order\_bids(T_j, B^s{}_{\tau',j}) =\{b_2, ..., b_q, b^a, b_{q+2}, ..., b_u\}$, $\tau' > \tau$ and $u \geq q$, which excludes the withdrawn bid b_1, where $b_{1,z} \in \{b_2, ..., b_q\}$ and $b^{sb} \in \{b_{1,z}, ..., b_u\}\backslash\{b_{1,z}\}$ (if b^{sb} was sent). The auctioneer α will attempt to provisionally grant its next preferred bid, which is b_2, and the same process as described above will occur, until withdrawn bid $b_e = b_{1,z}$, $e \leq q$, is provisionally granted from bidder γ_z (this may occur with the first withdrawn bid b_1). Therefore, the updated bid b^{sb} by bidder γ_z will be the bid at the top of its preference order, and from above, $order\_bids(T_j, B_{\tau',j,z}) =\{ \boldsymbol{b^{pa}}, b_{2,z}, ..., b_{u,z}\}$, $\tau' > \tau$ (assume bid $\boldsymbol{b^{pa}}$ is still preferred at time τ''), and hence $b^{sb} = \boldsymbol{b^{pa}}$. The auctioneer will be at step 3, and the withdrawn bid b_e will be added to the list of rejected and withdrawn bids, $b_e \in B^{wr}{}_{\tau',j}$. There are two possibilities: (a) the ordered list of bids for T_j now becomes $order\_bids(T_j, B^s{}_{\tau'',j}) =\{b_{e+1}, ..., \boldsymbol{b^{pa}}, ..., b_u\}$, $\tau'' > \tau$, i.e. there are withdrawn bids preceding bid $\boldsymbol{b^{pa}}$, which in this situation, we have case (i) above, which states that eventually the auctioneer α will arrive at the available and preferred bid $\boldsymbol{b^{pa}}$; (b) the ordered list of bids for T_j now becomes $order\_bids(T_j, B^s{}_{\tau'',j}) =\{ \boldsymbol{b^{pa}}, ..., b_u\}$, $\tau'' > \tau$, and thus the auctioneer α has arrived at the available and preferred bid $\boldsymbol{b^{pa}}$.

Case (iii): When there are no available bids, bidder γ_k is unable to submit an updated bid, and therefore, no updated bid b^{sb} is sent at step 3. The withdrawn bid is added to the list of rejected and withdrawn bids, $b_1 \in B^{wr}{}_{\tau,j}$, and thus from Definition 7, the ordered list of bids for T_j now becomes *order_bids*$(T_j, B^s_{\tau,j}) = \{b_2, \ldots, b_q\}$, as $u = q$. At step 3, the auctioneer will go through the same process with withdrawn bid b_2, attempting to provisionally grant b_2 at step 3, finding that b_2 is withdrawn at step 4 and receiving no updated bid at step 2, etc., until it arrives at bid b_q and provisionally grants b_q. Auctioneer α will receive a withdrawn message for b_q and proceed to step 2 where no updated bid is submitted. The protocol proceeds to step 3 where the auctioneer has no remaining bids, and hence no available bids for its set of tasks T_j.

Therefore, during provisionally granting a bid for an auctioneer α set of tasks T_j, and α's most preferred bid(s) are withdrawn, the auctioneer will eventually arrive at either its most preferred and available bid $\boldsymbol{b}^{\boldsymbol{pa}}$, or no bids, when using PAP.

Q.E.D.

Lemma 4: For any announced set of tasks T_j at time τ, an auctioneer using PAP will receive its preferred bid $\boldsymbol{b}^{\boldsymbol{pb}}$ for T_j, if one exists, from one of the available bidders during initial bidding, or will receive no bids for T_j.

Proof:

Refer to the PAP specification in section 5.1 (Figure 21), Definition 6 and Definition 7. For auctioneer α's announced set of tasks T_j at time τ, α's most preferred bid of the compete set of bids $B^c_{\tau,j}$ is given by *order_bids*$(T_j, B^c_{\tau,j}) = \{\boldsymbol{b}^{\boldsymbol{pb}}, b_2 \ldots, b_w\}$, $w \leq n$ if an available bid exists, and by *order_bids*$(T_j, B^c_{\tau,j}) = \varnothing$ if no available bids exists. When a set of tasks T_j is received by a bidder $\gamma_k \in \Gamma_\tau$, the protocol specification states that it will submit its best (preferred) bid for T_j. Therefore, each bidder γ_k will sort their bids $B_{\tau,j,k}$, to obtain an ordered list of bids *order_bids*$(T_j, B_{\tau,j,k}) = \{b_{1,k}, b_{2,k}, \ldots, b_{u,k}\}$, $u \geq 0$, and each bidder γ_k will submit the bid at the top of the order, $b_{1,k}$ if $u > 0$ (bid exists) or will submit nothing if $u = 0$ (no available bid exists by bidder γ_k).

If no available bid exists for T_j, *i.e.* *order_bids*$(T_j, B^c_{\tau,j}) = \varnothing$, then each bidder γ_k either **(i)** *order_bids*$(T_j, B_{\tau,j,k}) = \varnothing$ or **(ii)** *order_bids*$(T_j, B_{\tau,j,k}) = \{b_{1,k}, b_{2,k}, \ldots, b_{u,k}\}$, where all $b_{1,k}, \ldots, b_{u,k}$ are unsuitable. Therefore when the set of tasks T_j is announced at step 1 of PAP, then for bidders $\Gamma^{(i)} \subseteq \Gamma_\tau$ with case (i) above, it is not able to submit any bids for T_j, and for the rest of the bidders $\Gamma^{(ii)} \subseteq \Gamma_\tau$ that have case (ii) above, from **Lemma 2**, each bidder

$\gamma_k \in \Gamma^{(ii)}$ will submit all their unsuitable bids (which will be provisionally rejected) until they have no suitable bids to submit, effectively resulting in the submission of no bids. Therefore, the auctioneer α will not have any available bids at step 3 for the set of tasks T_j.

If an available bid does exist $order\_bids(T_j, B^c_{\tau,j}) \neq \varnothing$, from Definition 6, $B_{\tau,j,1} \cup B_{\tau,j,2} \cup \ldots \cup B_{\tau,j,p} = B^c_{\tau,j}$, and since $\boldsymbol{b^{pb}} \in B^c_{\tau,j}$, then $\exists k[\boldsymbol{b^{pb}} \in B_{\tau,j,k}]$, i.e. one of the bidders γ_k must hold bid $\boldsymbol{b^{pb}}$. $\boldsymbol{b^{pb}}$ is preferred over all γ_k*'s* bids for T_j, i.e. $\forall y[\boldsymbol{b^{pb}} \in order\_bids(T_j, B_{\tau,j,k})\ \&\ b_y \in order\_bids(T_j, B^c_{\tau,j})\ \&\ b_y \in order\_bids(T_j, B_{\tau,j,k})\ \&\ \boldsymbol{b^{pb}} \neq b_y \Rightarrow \boldsymbol{b^{pb}} > b_y]$. There are two cases:

(a) There are no additional "unsuitable" bids which appear in $order\_bids(T_j, B_{\tau,j,k})$, $\boldsymbol{b^{pb}} \in B_{\tau,j,k}$, that have a greater preference than $\boldsymbol{b^{pb}}$, i.e. $\forall y[b_y \notin order\_bids(T_j, B^c_{\tau,j})\ \&\ b_y \in order\_bids(T_j, B_{\tau,j,k}) \Rightarrow \boldsymbol{b^{pb}} > b_y]$ where b_y are unsuitable bids. If this is the case, then the bid $\boldsymbol{b^{pb}}$ is the at the top of the ordered list of bids for the set of tasks T_j (the best bid), i.e. $\forall y[b_y \in order\_bids(T_j, B_{\tau,j,k})\ \&\ \boldsymbol{b^{pb}} \neq b_y \Rightarrow \boldsymbol{b^{pb}} > b_y]$, and $order\_bids(T_j, B_{\tau,j,k}) = \{\boldsymbol{b^{pb}}, b_{2,k}, \ldots, b_{u,k}\}$. Therefore, bidder γ_k will send its best bid $\boldsymbol{b^{pb}}$ to the auctioneer.

(b) There are additional "unsuitable" bids which appear in $order\_bids(T_j, B_{\tau,j,k})$, $\boldsymbol{b^{pb}} \in B_{\tau,j,k}$, that have a greater preference than $\boldsymbol{b^{pb}}$, i.e. $\exists y[b_y \notin order\_bids(T_j, B^c_{\tau,j})\ \&\ b_y \in order\_bids(T_j, B_{\tau,j,k})\ \&\ b_y > \boldsymbol{b^{pb}}]$ where b_y are unsuitable bids. Therefore, we have $order\_bids(T_j, B_{\tau,j,k}) = \{b_{1,k}, \ldots, b_{q,k}, \boldsymbol{b^{pb}}, b_{q+2,k}, \ldots, b_{u,k}\}$, $q \geq 1$, and q bids $b_{1,k}$ to $b_{q,k}$ are the unsuitable bids. According to Lemma 2, the bidder γ_k will eventually send the bid $\boldsymbol{b^{pb}}$ to the auctioneer α in the current bidding phase.

Therefore, in both cases (a) and (b) above, the auctioneer will receive its preferred bid $\boldsymbol{b^{pb}}$ for the initial announcement of the set of tasks T_j at time τ if a bid is available. Therefore, for the initial announcement of the set of tasks T_j at time τ, the auctioneer α will receive its most preferred bid $\boldsymbol{b^{pb}}$ or will receive no bids, at step 3 of PAP.

Q.E.D.

Lemma 5: When an auctioneer backtracks to the set of tasks T_j at time τ using PAP, requiring it to provisionally reject its previously provisionally granted bid $\boldsymbol{b^r}$ from bidder γ_k, where $\boldsymbol{b^r}$ was the auctioneer's most preferred bid for T_j at time $\xi < \tau$, then the auctioneer will obtain (receive or have held) its next preferred available bid $\boldsymbol{b^p}$ for the set of tasks T_j, or will have no further available bids for T_j. We assume the auctioneer allows enough time for bidders to process and submit their updated bids, if one exists, before proceeding with planning (i.e. bidding deadline is set conservatively).

Proof:

Refer to the PAP specification in section 5.1 (Figure 21), Definition 6 and Definition 7. At time τ bid $\boldsymbol{b^r}$ by bidder γ_k is provisionally rejected for the set of tasks T_j, therefore the rejected bid is added to the list of rejected bids for both the bidder γ_k and auctioneer α, i.e. $\boldsymbol{b^r} \in B^r_{\tau,j,k}$ and $\boldsymbol{b^r} \in B^{wr}_{\tau,j}$. In the centralised case, the auctioneer α's new preferred bid for set of tasks T_j at time τ of the compete set of bids $B^c_{\tau,j}$ if one is available is $order\_bids(T_j, B^c_{\tau,j}) = \{\boldsymbol{b^p}, b_2 \ldots, b_w\}$, $w \leq n\text{-}1$, or is $order\_bids(T_j, B^c_{\tau,j}) = \varnothing$ if no further bid for T_j is available, where the ordered lists now excludes bid $\boldsymbol{b^r}$.

If a bid is available, $order\_bids(T_j, B^c_{\tau,j}) \neq \varnothing$, from Definition 6, $B_{\tau,j,1} \cup B_{\tau,j,2} \cup \ldots \cup B_{\tau,j,p} = B^c_{\tau,j}$, and since $\boldsymbol{b^p} \in B^c_{\tau,j}$, $\exists y[\boldsymbol{b^p} \in B_{\tau,j,y}]$. After the provisional rejection, which occurs at step 5 of PAP, the protocol proceeds to step 2 where the bidder γ_k is able to submit an updated bid, which we denote b^u, that is suitable (from Lemma 2) or submit nothing (no communication). The protocol then proceeds to step 3. There are three cases:

Case (1): The bid $\boldsymbol{b^p}$ was available at time ξ, i.e. $\boldsymbol{b^p} \in B^c_{\xi,j}$, but is not one of the bids held by the provisionally rejected bidder γ_k, i.e. $\boldsymbol{b^p} \notin B_{\xi,j,k}$ & $\boldsymbol{b^r} \in B_{\xi,j,k}$, but $\boldsymbol{b^p}$ is held by another bidder γ_z, where $z \neq k$ and $\boldsymbol{b^p} \in B_{\xi,j,z}$. A few situations are possible: **(a)** At time ξ, the bid $\boldsymbol{b^p}$ was at the top of the list of ordered bids for T_j by bidder γ_z, i.e. $order\_bids(T_j, B_{\xi,j,z}) = \{\boldsymbol{b^p}, b_{2,z}, \ldots, b_{k,z}\}$, and therefore, $\boldsymbol{b^p}$ has been already sent by γ_z as bidders in PAP send their best bid for a task, and the auctioneer therefore already has bid $\boldsymbol{b^p}$. Note that at time ξ, there could have been unsuitable bids above $\boldsymbol{b^p}$, i.e. $order\_bids(T_j, B_{\xi,j,z}) = \{b_{1,z}, \ldots, b_{q,z}, \boldsymbol{b^p}, b_{q+2,z}, \ldots, b_{u,z}\}$, where $b_1, \ldots, b_q$ are unsuitable for T_j, but from Lemma 2, bid $\boldsymbol{b^p}$ will eventually be placed on top of the bid order and be submitted to the auctioneer α. If the bid $\boldsymbol{b^p}$ is not at the top of the auctioneer's bid preference order *order_bids*$(T_j, B^s_{\tau,j}) = \{b_1, \ldots, b_q, \boldsymbol{b^p}, b_{q+2}, \ldots, b_u\}$, then it is because the bids $b_1, \ldots, b_q$ that are above $\boldsymbol{b^p}$ are withdrawn (in the context of this lemma, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids held by the auctioneer, $B^s_{\tau,j}$, are withdrawn), and from Lemma 3, the auctioneer will eventually arrive at bid $\boldsymbol{b^p}$. **(b)** At time ξ, the bid $\boldsymbol{b^p}$ was not at the top of the list of ordered bids for T_j by bidder γ_z, i.e. *order_bids*$(T_j, B_{\xi,j,z}) = \{b_{1,z}, \ldots, b_{q,z}, \boldsymbol{b^p}, b_{2,z}, \ldots, b_{u,z}\}$, and bid $b_{1,z}$ was sent for T_j, but at time τ, $b_{1,z}$ is withdrawn (in the context of this theorem, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids by the auctioneer are withdrawn, and any better bids by the bidder are unsuitable), so $b_{1,z} \in$ *order_bids* $(T_j, B^s_{\tau,j})$, and $\forall y[(b_y \in B^s_{\tau,j} \;\&\; b_y) > \boldsymbol{b^p} \Rightarrow withdrawn(b_y)]$, $b_{1,z} > \boldsymbol{b^p}$, where *withdrawn*$(b_y)$ signifies that bid b_y is withdrawn. Therefore, the top one or more preferred bids by the auctioneer, *order_bids* $(T_j, B^s_{\tau,j})$, are withdrawn. From Lemma 3, the auctioneer will eventually arrive at bid $\boldsymbol{b^p}$, sent by bidder γ_z, for either possible γ_z's ordered list – *order_bids*$(T_j, B_{\tau,j,z}) = \{\boldsymbol{b^p}, b_{2,z}, \ldots, b_{u,z}\}$, $u \geq 1$, or *order_bids*$(T_j, B_{\tau,j,z}) = \{b_{1,z}, \ldots, b_{q,z}, \boldsymbol{b^p}, b_{q+2,z}, \ldots, b_{u,z}\}$, $u \geq q+1$, where bids $b_1, \ldots, b_q$ are unsuitable for T_j. In both cases (a) and (b), since $\boldsymbol{b^p}$ is the preferred bid, then $\boldsymbol{b^p} > b^u$ if an updated bid b^u is submitted by bidder γ_k.

Case (2): The bid $\boldsymbol{b^p}$ was available at time ξ, i.e. $\boldsymbol{b^p} \in B^c_{\xi,j}$, and is one of the bids held by the provisionally rejected bidder γ_k, i.e. $\boldsymbol{b^p} \in B_{\xi,j,k} \;\&\; \boldsymbol{b^r} \in B_{\xi,j,k}$. Once the bid $\boldsymbol{b^r}$ is rejected, the bid order for γ_k becomes either: **(a)** *order_bids*$(T_j, B_{\tau,j,k}) = \{\boldsymbol{b^p}, b_{2,k}, \ldots, b_{u,k}\}$, $u \geq 1$, in which case the updated (next best) bid sent by γ_k is $\boldsymbol{b^p}$, i.e. $b^u = \boldsymbol{b^p}$; **(b)** *order_bids*$(T_j, B_{\tau,j,k}) = \{b_{1,k}, \ldots, b_{q,k}, \boldsymbol{b^p}, b_{q+2,k}, \ldots, b_{u,k}\}$, $u \geq q+1$, where bids $b_1, \ldots, b_q$ are unsuitable for T_j (in the context of this theorem, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids by the bidder are unsuitable), therefore, the updated (next best) bid from bidder γ_k is $b_{1,k}$, i.e. $b^u = b_{1,k}$, which is not suitable for T_j, and from Lemma 2, the bidder will eventually send bid $\boldsymbol{b^p}$.

Case (3): The bid $\boldsymbol{b^p}$ was not available at time ξ, i.e. $\boldsymbol{b^p} \notin B^c_{\xi,j}$, but became available at a later time, $\boldsymbol{b^p} \in B^c_{\tau',j}$, $\xi < \tau' \leq \tau$. Therefore, a bidder γ_z acquired the bid $\boldsymbol{b^p}$ at time τ'. There are three possible situations: **(1)** Bidder $\gamma_z \neq \gamma_k$ did not send any bids previously as it did not have any bids to submit, therefore *order_bids*$(T_j, B_{t,j,z}) = \emptyset, t < \tau'$ and *order_bids*$(T_j, B_{t,j,z}) = \{ \boldsymbol{b^p} \}, t \geq \tau'$. According to PAP's bidding policy, if a bid becomes available that is better than the worst previously sent bid, then the bidder should send this *updated better bid.* As no bids were sent previously, any bid is better than no bid. Hence bidder γ_z sends the bid $\boldsymbol{b^p}$ at time $\tau' < \tau$, and thus the auctioneer should already have the bid $\boldsymbol{b^p}$. **(2)** Bidder γ_z acquired the bid $\boldsymbol{b^p}$ after it submitted a previous bid(s), which the worst submitted bid is b^w, where $\boldsymbol{b^p} > b^w$. According to PAP's bidding policy, if a bid becomes available that is better than the worst previously sent bid, then the bidder should send this *updated better bid.* Hence bidder γ_z sends the bid $\boldsymbol{b^p}$ at time $\tau' < \tau$, and thus the auctioneer should already have the bid $\boldsymbol{b^p}$; **(3)** Bidder $\gamma_z \neq \gamma_k$ acquired the bid $\boldsymbol{b^p}$ after it submitted a previous bid(s), which is the worst submitted bid is b^w, where $\boldsymbol{b^p} < b^w$, in which case b^w is currently withdrawn at time τ (in the context of this lemma, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids held by the auctioneer $B^s_{\tau,j}$, are withdrawn), so $b^w \in$ *order_bids* $(T_j, B^s_{\tau,j})$, and $\forall y[(b_y \in B^s_{\tau,j} \; \& \; b_y > \boldsymbol{b^p}) \Rightarrow withdrawn(b_y)]$, where $withdrawn(b_y)$ signifies that bid b_y is withdrawn. Therefore, the top one or more preferred bids by the auctioneer, *order_bids* $(T_j, B^s_{\tau,j})$, are withdrawn. From Lemma 3, the auctioneer will eventually arrive at bid $\boldsymbol{b^p}$, sent by bidder γ_z, for either possible γ_z's ordered list – *order_bids*$(T_j, B_{\tau,j,z}) = \{\boldsymbol{b^p}, b_{2,z}, \ldots, b_{u,z}\}, u \geq 1$, or *order_bids*$(T_j, B_{\tau,j,z}) = \{b_{1,z}, \ldots, b_{q,z}, \boldsymbol{b^p}, b_{q+2,z}, \ldots, b_{u,z}\}, u \geq q+1$, where bids $b_1, \ldots, b_q$ are unsuitable for T_j. **(4)** Bidder $\gamma_z = \gamma_k$ acquired the bid $\boldsymbol{b^p}$ after it submitted a previous bid(s), which the worst submitted bid is b^w, where $\boldsymbol{b^p} < b^w$, in which case b^w is either the bid that was provisionally rejected, i.e. $b^w = \boldsymbol{b^r}$, or b^w is another bid submitted in addition to $\boldsymbol{b^r}$ for T_j that must be currently withdrawn at time τ (in the context of this lemma, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids held by the auctioneer $B^s_{\tau,j}$, are withdrawn). If $b^w = \boldsymbol{b^r}$, we have case (2) above, ignoring the fact that bid $\boldsymbol{b^p}$ arrived later. If $b^w \neq \boldsymbol{b^r}$, then we have situation (3) above (in case (3)), ignoring the fact that $\gamma_z \neq \gamma_k$. Therefore, the auctioneer will receive bid $\boldsymbol{b^p}$ for T_j. For all situations, since $\boldsymbol{b^p}$ is the preferred bid, then $\boldsymbol{b^p} > b^u$ if an updated bid $b^u \neq \boldsymbol{b^p}$ is submitted by bidder γ_k. For situations (1) and (2), if the bid $\boldsymbol{b^p}$ is not at the top of the auctioneer's bid preference order *order_bids*$(T_j, B^s_{\tau,j}) = \{b_1, \ldots, b_q, \boldsymbol{b^p}, b_{q+2}, \ldots, b_u\}$, this is because the bids $b_1, \ldots, b_q$ that are above $\boldsymbol{b^p}$ are withdrawn (in the context of this lemma, $\boldsymbol{b^p}$ is the most preferred *available and suitable* bid, hence any better bids held by the auctioneer, $B^s_{\tau,j}$, are withdrawn), and from Lemma 3, the auctioneer will eventually arrive at bid $\boldsymbol{b^p}$.

Therefore, in all three cases, the auctioneer using PAP will obtain its most preferred bid $\boldsymbol{b^p}$ for the set of tasks T_j at time τ when it backtracks to T_j, if an available bid exists for T_j.

If no available bid exists for T_j, *i.e.* $order\_bids(T_j, B^c_{\tau,j}) = \emptyset$, then there are two cases: (i) $order\_bids(T_j, B^s_{\tau,j}) = \emptyset$, (ii) $order\_bids(T_j, B^s_{\tau,j}) = \{b_1, \ldots, b_q\}$ where bids $b_1 \ldots, b_q$ are withdrawn. After the provisional rejection of $\boldsymbol{b^r}$, which occurs at step 5 of PAP, the protocol proceeds to step 2 where the bidder γ_k is able to submit an updated bid b^u that is suitable (from Lemma 2) or submit nothing (no communication). Since there are no available bids for T_j, from Lemma 2, no suitable updated bid b^u will be submitted by γ_k. The protocol then proceeds to step 3. For case (i), the auctioneer α is left with no available bid for T_j. For case (ii), from Lemma 3, the auctioneer will eventually be left with no available bid for T_j.

Therefore, when the auctioneer α backtracks to the set of tasks T_j at time τ, it will either obtain its next preferred bid $\boldsymbol{b^p}$ for T_j, or have no further bids available for T_j, at step 3 of PAP.

Q.E.D.

Theorem 6: PAP allows an auctioneer to produce the same plan (set of bids to achieve an auctioneer's task) as a centralised depth-first search with a dynamic search tree. In the decentralised (PAP) case, the auctioneer acquires bids from a set of bidders to achieve its initial set of tasks T_0. In the centralised case, the auctioneer holds all the possible bids that are available in the decentralised case, and can perform a depth-first search on the bids in order to achieve its initial set of tasks T_0. In both cases, the bids, and hence the branches in the search tree that the auctioneer is performing the depth first search on, may change with time (except for selected/granted bids/branches).

Proof:

Refer to the PAP specification in section 5.1 (Figure 21), Definition 6 and Definition 7.

Centralised Depth First Search

Stages in depth first search:

(1) *Condition*:

a. If T_j is a new set of tasks (or node in search tree, initially $T_j \leftarrow T_0$), find the list of bids $order\_bids(T_j, B^c_{\tau,j}) = \{b^p, b_2, \ldots, b_n\}, n \geq 0$ (these bids are the branches of the search tree from the node T_j) to obtain the auctioneer α's most preferred (available and suitable) bid b^p for T_j at time τ.

b. Else (arrive from backtracking at stage 2.b or 2.a.ii.2, and thus bid b^r is deselected for T_j) find the new list of bids $order\_bids(T_j, B^c_{\tau,j}) = \{b^p, b_2, \ldots, b_n\}, n \geq 0$, excluding the backtracked bid b^r (b^r is discarded), i.e. $b^r \notin order\_bids(T_j, B^c_{t,j})$ for $t \geq \tau$, to obtain the auctioneer's α most preferred (available and suitable) bid b^p for T_j at time τ.

(2) *Condition:*

a. If bids are available for selection ($n \geq 1$) for T_j AND *plan_suitable*:

   i. Select the bid b^p.

   ii. Condition:

      1. If $task\_achieved(T_j, b^p)$ & *plan_suitable,* then exit (success) – the plan for the set of tasks T_0 consists of the selected bids.

      2. Else, if $task\_achieved(T_j, b^p)$ & $\neg$ *plan_suitable,* then backtrack to T_j by deselecting bid $b^r = b^p$ for T_j, i.e. backtracked back to T_j. Go to stage (1).

      3. Else, the new set of tasks that must be achieved in order to achieve T_0 is $T_j \leftarrow diff(T_j, b^p)$, i.e. the child node of bid/branch b^p. Go to step (1)

b. Else no bids are available for T_j ($n = 0$, an infeasible solution) or $\neg$*plan_suitable*:

   i. Backtracking required. C*ondition:*

      1. If $T_j = T_0$ (initial/root task/node), then backtracking from root node, so exit (failure, no solution found).

2. Else, backtrack to previous set of tasks T_c by deselecting bid b^r for T_c (with T_c as parent node of branch/bid b^r) and discard T_j (child node of branch/bid b^r). $T_j \leftarrow T_c$, go to stage (1).

PAP Performing the Depth-First Search

We will show that PAP performs equivalent stages that the centralised depth first search performs. Note that the dynamic bids can simulate bidders negotiating with other auctioneers at the same time as negotiating with auctioneer α – i.e. bids are withdrawn due to bids being provisionally/confirm granted by other auctioneers. Additionally, if bidders during PAP submit bids for a set of tasks T_j that was involved in backtracking, the auctioneer will send the bidder a withdrawn message to inform the bidder that the task is no longer available. In the following proof, we ignore any occurrences of this, as it does not affect the planning.

(1) *Condition*:

a. **[Start – achieve initial task T_0]** PAP commences at step 1 at time τ with the task announcement of the initial set of tasks T_0. From Lemma 4, the auctioneer will receive its most preferred (available and suitable) bid b^p from the set of bidders Γ_τ for T_0, or will receive no bids for T_0, and arrives at step 3 of PAP.

   [New set of tasks T_j to achieve, from stage 2.a.ii.3] For any following tasks T_j that need to be achieved, because a bid was selected for a previous task that did not achieve the complete task, then a new PAP protocol process is started to achieve T_j, which commences at step 1 at time τ with the task announcement of T_j. From Lemma 4, the auctioneer will obtain its most preferred (available and suitable) bid b^p from the set of bidders Γ_τ for T_j, or will have no bids for T_j, and arrives at step 3 of PAP.

b. **[Backtrack to T_j, from stage 2.b or 2.a.ii.2 below]** After backtracking at time τ (bid b^r is provisionally rejected, and discarded, for T_j at step 5), from Lemma 5, the auctioneer α will obtain its next best/preferred (available and suitable) bid b^p for task T_j, or will have no bids remaining for T_j, and arrives at step 3 of PAP.

(2) *Condition:*

a. **[Bid b^p available for T_j AND *plan_suitable]*:**

   i. At step 3 of PAP, bid b^p by bidder γ_k is provisionally granted, and the Protocol proceeds to step 4. Since the bid b^p is available, it is not withdrawn, and hence bidder γ_k should not send a withdrawn or provisionally withdrawn message. Therefore, at step 4, the bidder γ_k should accept the provisional grant for its bid b^p.

   ii. Condition:

      1. If *task_achieved*(T_j, b^p) *& plan_suitable,* then the auctioneer α has found a suitable plan for the set of tasks T_0 which consists of the provisionally granted bids. PAP will proceed to step 5 (control path (f)), where all the provisionally granted bids are confirm granted. The protocol exits (successfully).
      2. Else, if *task_achieved*(T_j, b^p) *&* $\neg$ *plan_suitable,* then the current plan that fully achieves T_0 is not suitable. The protocol proceeds to step 5 (control path (f)) and the bid b^r = b^p for T_j by bidder γ_k is provisionally rejected, i.e. backtracked back to T_j. The protocol proceeds to step 2 (for task T_j). Go to stage (1).
      3. Else, the new set of tasks that must be achieved in order to achieve T_0 is $T_j \leftarrow$ *diff*(T_j, b^p). Since the task is not completely achieved, then take control option (e) back to step 1 of a new protocol process for new task T_j, go to step (1).

b. **[Bid b^p <u>not</u> available for T_j OR $\neg$*plan_suitable]*:**

   i. Backtracking required at step 3 of PAP. *Condition:*

1. If T_j = T_0 (auctioneer α's initial set of tasks), then PAP specification states that the protocol takes control path (c) and exits the protocol (failure, no solution found).
2. Else, take control option (d) and proceed to step 5 of the previous protocol process, for the set of tasks T_c. Provisionally reject bid b^r for T_c, and discard b^r and T_j. Proceed to step 2 of T_c's protocol process, make $T_j \leftarrow T_c$, and go to step (1).

Therefore, PAP performs the equivalent stages as the centralised depth first search, and thus, PAP performs a decentralised (distributed) depth first search on a dynamic search tree (where the branches/bids change with time).

Q.E.D.

5.3.5 Time Complexity

PAP allows an auctioneer to perform a depth-first search (Theorem 6) in order to find a plan and task allocation to achieve its set of tasks. The worse case time complexity for a depth-first search is exponential. If *br* is the branching factor, which is the maximum number of bids per set of tasks, and *m* is the depth of the tree, which is the maximum number of bids required to achieve the auctioneer's set of tasks, the maximum number of bids that must be considered (e.g. searched) before a solution is found is given by (Russell and Norvig 1995)

Eq 4: Maximum number of bids to consider $= \sum_{i=1}^{m} br^i$

and therefore, the time complexity is of order (Russell and Norvig 1995)

Eq 5: Time complexity = $O\left(br^m\right)$

A depth first search is only likely to get close to the worst case time complexity if it is used in a domain where there are few possible solutions (Russell and Norvig 1995). Therefore, if PAP is used in a domain where there are many possible solutions that could be arrived at, and since a depth first search uses minimal backtracking (usually only if an infeasible solution is encountered, and therefore a fairly greedy planning approach), a solution should be found relatively quickly, but most likely at the expense of the quality of the solution.

$\Delta\tau = d_i + pgad_i$, is the sum of the bidding deadline d_i and the provisional grant acceptance deadline $pgad_i$, which is the maximum time duration that must pass for the planning to progress in PAP [18]. The auctioneer waits less than $\Delta\tau$ before either: selecting a bid for a task, initially or during backtracking; when backtracking, as the auctioneer creates a new task and waits less than $\Delta\tau$ before backtracking; or when bids are provisionally rejected [19] or (provisionally) withdrawn, to wait for an updated bid to be sent to replace the rejected or withdrawn bid. For each branch that requires backtracking, the auctioneer must wait $2\cdot\Delta\tau$. The actual time for the auctioneer to find a plan to achieve its set of tasks is therefore:

Eq 6: Planning time $\leq br_w \cdot \Delta\tau + br_s \cdot 2 \cdot \Delta\tau - br_c \cdot \Delta\tau = (br_w + 2 \cdot br_s - br_c) \cdot \Delta\tau$

where br_w is the total number of bids (or branches in the search tree) that were provisionally rejected or withdrawn, br_s is the total number of bids *searched* in the planning tree, and br_c ($\leq m$) is the total number of bids in the path of the final plan (hence are searched but did not require backtracking as they are confirm granted). Using *Eq 4* and *Eq 6*, the worst case planning time is when all the bids except the final plan (confirm granted bids) are backtracked, i.e. no withdrawn or rejected bids, as backtracked bids consume twice the time, therefore we have:

Eq 7: Worst case planning time $= 2 \cdot \Delta\tau \cdot \sum_{i=1}^{m} br^i - \Delta\tau \cdot m \approx 2 \cdot \Delta\tau \cdot \sum_{i=1}^{m} br^i$

[18] Assume that d_i commences at the beginning of the task announcement at step 1, and $pgad_i$ commences at the beginning of the provisional grant at step 3 of the PAP. Assume time to confirm grant or provisionally reject bids at step 5 is negligible compared to the overall planning time.

[19] This only includes bids that are provisionally rejected for the current set of tasks (i.e. the current protocol process) since bids can be sent anytime for another (previous) set of tasks and rejected.

where $br = br_s$ is the total number of bids that are searched, rejected or withdrawn in the planning tree.

The planning time is dependent on the size of the planning tree, and hence the amount of backtracking required, and the duration $\Delta\tau$. $\Delta\tau$ should consider: (i) communication time to send tasks, bids, the provisional grant and the provisional grant acceptance; (ii) the computation time for bidders to form, process and find its best bid for the auctioneer, and to decide whether it will, or can, accept the provisional grant and commit to the bid. Ideally, to minimise planning time, $\Delta\tau$ should be as small as possible such that there is just enough time for agents to communicate and complete their computation.

5.3.6 Communication Requirements

In order to calculate the worst case communication requirements (send and receive) to achieve an auctioneer's task using PAP, we consider the tasks, bids, and speech acts (grants, rejects and withdrawals) that are communicated among all the agents in the worst case planning scenario. *Eq 4* provides the worst case number of bids that must be considered to find a solution, which is also equivalent to the maximum number of set of tasks (or nodes) in the search tree that are considered (includes the initial/root set of tasks, but not the last set of tasks associated with the final bid that is searched/granted to produce the final plan, as the set of tasks is empty/achieved). Therefore,

Eq 8: Max number of set of tasks considered = $\sum_{i=1}^{m} br^i$

The set of tasks that need to be announced could contain separate independent sub-tasks which can be bid for separately, and therefore it may be more appropriate to announce the sub-tasks separately. This occurs in our transportation implementation that is discussed in the next chapter. Let λ be the maximum number of tasks for every set of tasks that need to be announced separately, κ is the maximum number of bidders that the auctioneer announces the set of tasks to, then from *Eq 8* the maximum number of task announcements *mta( )* sent by the auctioneer α will be:

Eq 9: $mta(\alpha) = \kappa \cdot \lambda \cdot \sum_{i=1}^{m} br^i$

and thus, the maximum number of set of tasks received each bidder γ is

Eq 10: $mta(\gamma) = \lambda \cdot \sum_{i=1}^{m} br^i$

For each announced set of tasks, bidders may send a maximum of β bids, where

Eq 11: $\beta \cdot \lambda \cdot \kappa = br$

but bidders only send bids for all set of tasks (nodes) in the search tree except for the leaf nodes, where the auctioneer backtracks due to an infeasible solution (no bids), so The number of announced set of tasks that bidders submit bids for (*ω*) is (add one for the initial/root set of tasks):

Eq 12: $\omega = \lambda \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right)$

Therefore, the maximum number of bids *mb( )* that bidders γ may submit during the planning process is (using *Eq 12*) is

Eq 13: $mb(\gamma) = \beta \cdot \omega = \beta \cdot \lambda \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right)$

The maximum number of bids that the auctioneer *α* may receive is given by *Eq 4*,

Eq 14: $mb(\alpha) = \sum_{i=1}^{m} br^i$

During planning with PAP, in the worst case, every bid searched may require a *maximum* of three speech acts – provisional grant, provisional grant accepted, and either provisional reject (for all backtracked bids) or confirm grant (for bids in the final plan). The auctioneer must send these negotiating speech acts for each branch in the planning tree, therefore, the maximum number of negotiating speech acts *mns( )* that the auctioneer *α* must send or receive is

Eq 15: $mns(\alpha) = 3 \cdot \sum_{i=1}^{m} br^i$

where *(2/3)·mns(α)* is the maximum number of speech acts that the auctioneer sends, and *(1/3)·mns(α)* is the maximum number of speech acts that the auctioneer receives. The bidder also communicates, in the worst case, a maximum of three speech acts for every bid that it submits to the auctioneer. The maximum number of speech acts that the bidders γ must send or receive is (using *Eq 13*):

Eq 16: $mns(\gamma) = 3 \cdot mb(\gamma) = 3 \cdot \beta \cdot \lambda \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right)$

where *(2/3)·mns(γ)* is the number of speech acts that the bidder receives, and *(1/3)·mns(γ)* is the number of speech acts that the bidder sends.

The total maximum communication requirements *tc( )* for the auctioneer α is (using *Eq 9*, *Eq 14* and *Eq 15*)

Eq 17: $tc(ac) = mta(\alpha) + mb(\alpha) + mns(\alpha)$

$$= \kappa \cdot \lambda \cdot \sum_{i=1}^{m} br^i + \sum_{i=1}^{m} br^i + 3 \cdot \sum_{i=1}^{m} br^i$$
$$= (\kappa \cdot \lambda + 4) \cdot \sum_{i=1}^{m} br^i$$

and the total maximum communication requirements for the bidder γ is (using *Eq 10*, *Eq 13* and Eq 16)

Eq 18: $tc(\gamma) = mta(\gamma) + mb(\gamma) + mns(\gamma)$

$$= \lambda \cdot \sum_{i=1}^{m} br^i + \beta \cdot \lambda \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right) + 3 \cdot \beta \cdot \lambda \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right)$$
$$= \lambda \cdot \left(\sum_{i=1}^{m} br^i + 4 \cdot \beta \cdot \left(\sum_{i=1}^{m-1} br^i + 1 \right) \right)$$

In many application domains, λ = *1*, i.e. the auctioneer announces its complete set of tasks once rather than announce components (sub-tasks) of the set of tasks separately. This is the case with our combinatorial auction application discussed in the next section. If this is the case, then *Eq 17* and *Eq 18* become

Eq 19: $tc(\alpha) = (\kappa + 4) \cdot \sum_{i=1}^{m} br^{i}$

Eq 20: $tc(\gamma) = \sum_{i=1}^{m} br^{i} + 4 \cdot \beta \cdot \left(\sum_{i=1}^{m-1} br^{i} + 1 \right)$

It is unlikely that agents will reach the worst case communication using PAP for many applications domains. As mentioned in the previous section, PAP is not likely to reach the worst case communication if applied to a domain with many possible solutions. Not all bids in the search tree with PAP may be searched (granted), and therefore, due to the persistence policy, auctioneers do not require a (provisional) reject message to be communicated in order to inform the bidders that their bids are not being used. Due to this, we show below that PAP requires less communication that CNP, CNP-ext and ECNP under certain circumstances. Bidders in PAP only send one bid at a time, of its maximum of β bids, for each sub-task in a set of tasks. Only if a bid is used and then provisionally rejected or withdrawn, does the bidder send another bid. Therefore, a bidder's complete β number of bids will only be communicated if all of its first $(\beta - 1)$ bids are provisionally rejected or withdrawn. The auctioneer need not send its task announcements to all bidders. Using domain knowledge, it may send its sets of tasks to the most suitable candidates, minimising communication. Not all the bidders that receive the task announcement may submit bids, as they may not have suitable bids for the set of tasks.

5.3.7 Communication Compared with CNP, ECNP and CNP-ext

Below we show that PAP requires less communication than CNP, CNP-ext and ECNP, *in the worst case*[20] and when used in the same manner as these protocols (i.e. solving simpler planning and task allocation problems than PAP is able to). The reduction in communication is predominantly due to PAP's persistence policy and the removal of the requirement to reject unused bids (see section 5.2.7). The saving in communication

[20] PAP's worst case communication versus CNP, CNP-ext and ECNP's best case communication. In the normal case, PAP is likely to save more communication.

compared with CNP, CNP-ext and ECNP occurs when bids submitted for a set of tasks is greater than 3, 5 and 2, respectively. We believe that there are many real world applications where this would be the case. With the combinatorial auction application in the next chapter, up to 100 bids were submitted for each set of tasks, and with the transportation scheduling application in chapter 7, up to 10 bids were submitted.

Note that since CNP and CNP-ext require bids that fully achieve the auctioneer's set of tasks, and each sub-task in the set of tasks require one bid to be granted for it, then the number of sub-tasks $\lambda = 1$. Since ECNP does not allow for multiple auctioneers, and thus multiple tasks, simultaneously, then $\lambda = 1$ also.

<u>Communication Requirements of PAP compared to CNP</u>

The best case communication requirements for CNP were presented in section 4.4.2, *Eq 1*. We now formulate the communication requirements for PAP, when PAP is used in the same manner as CNP, i.e. a single auctioneer, only allows bids that can fully achieve the auctioneer's announced set of tasks, and precludes withdrawals, the updating of bids, rejection of bids and backtracking.

Refer to the PAP specification in section 5.1 (Figure 21). Assume a maximum of κ bidders and a maximum of *br* bids received per task announcement. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, therefore communicates κ messages. The protocol proceeds to step 2, where bidders submit (communicate) *br* **bids** that fully achieved the announced task. The protocol proceeds to step 3. Since we do not consider withdrawals and backtracking in the context of the analysis, these speech acts or events (e.g. withdrawn, provisionally reject and backtracking) can be ignored at steps 3. The auctioneer will grant its most preferred bid, and therefore communicates *1* **provisional grant** message. PAP proceeds to step 4 where *1* **provisional grant accepted** message is communicated since the bidder is unable to (provisionally) withdraw the bid in the context of the analysis. As the bid fully achieves the announced task, the protocol proceed to step 5 (take control option (f)). In the context of the analysis, rejection of bids are precluded, and therefore *1* **confirm grant** message is communicated. Therefore, the total communication required by PAP for a task allocation is the sum of the number of messages communicated:

Eq 21: Total communication for PAP $= \kappa + br + 1 + 1 + 1 = \kappa + br + 3$

Conditions for reduced communication compared with CNP

Using *Eq 1* and *Eq 21*, PAP requires less communication than CNP if $br > 3$. This follows because:

Total communication for PAP < Total communication for CNP

if $\kappa + br + 3 < \kappa + 2 \cdot br$

$\therefore br > 3$

Saving in communication compared with CNP

Using *Eq 1* and *Eq 21*, the saving in communication by using PAP over CNP is:

Saving in comms = Total comms for CNP - Total comms for PAP

$\therefore$ *Saving in comms* $= \kappa + 2 \cdot br - (\kappa + br + 3)$

$\therefore$ *Saving in comms* $= br - 3$

Communication Requirements of PAP compared to ECNP

The best case communication requirements for ECNP were presented in section 4.8.2, *Eq 3*. We now formulate the communication requirements for PAP, when PAP is used in the same manner as ECNP, i.e. a single auctioneer that precludes withdrawals, the updating of bids, the rejection of bids on submission, and backtracking only when a plan is found and deemed unsuitable.

Refer to the PAP specification in section 5.1 (Figure 21). Assume a maximum of κ bidders, a maximum of br bids received per task announcement, and m is the depth of the search (number of bids to achieve the initial set of tasks). At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, and therefore communicates κ messages. The protocol proceeds to step 2, where bidders submit (communicate) br **bids**. PAP proceeds to step 3. Since we do not consider withdrawals, rejection on bid submission and backtracking until the end, in the context of the analysis, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. The auctioneer will grant its most preferred bid, and therefore communicates *1* **provisional**

grant message. PAP proceeds to step 4 where *1* **provisional grant accepted** message is communicated since the bidder is unable to (provisionally) withdraw the bid in the context of the analysis. If the bid does not fully achieve the set of tasks announced by the auctioneer, then PAP proceeds to step 1 (control option (e)) to announce the remaining set of tasks that the bid did not achieve, and the same process is repeated until the m^{th} bid's provisional grant is accepted at step 4 which does fully achieve the set of tasks announced by the auctioneer. Therefore, PAP has performed step 1 to step 4 *m* times, and thus the total communication at this point is $(\kappa + br + 1 + 1)\cdot m$. The protocol proceeds to step 5 (take control option (f)) where the auctioneer will either send all *m* provisionally granted bids a **confirm grant** (if the plan is suitable) or a **provisional reject** (if the plan is unsuitable – refer to Theorem 2 proof for detailed discussion on how this occurs), and thus communicates *m* messages. The total communication required by PAP to plan and allocate tasks is the sum of the number of messages communicated:

Eq 22: Total communication for PAP $= (\kappa + br + 2)\cdot m + m$

<u>Conditions for reduced communication compared with ECNP</u>

Using *Eq 3* and *Eq 22*, PAP requires less communication than ECNP if $br > 2$. This follows because:

Total communication for PAP < Total communication for ECNP

if $(\kappa + br + 2)\cdot m + m < (\kappa + 2\cdot br)\cdot m + m$

$\therefore br > 2$

<u>Saving in communication compared with ECNP</u>

Using *Eq 3* and *Eq 22*, the saving in communication by using PAP over ECNP is:

Saving in comms = Total comms for ECNP - Total comms for PAP

$\therefore$ *Saving in comms* $= (\kappa + 2\cdot br)\cdot m + m - ((\kappa + br + 2)\cdot m + m)$

$\therefore$ *Saving in comms* $= (br - 2)\cdot m$

<u>Communication Requirements of PAP compared to CNP-ext</u>

The best case communication requirements for CNP-ext were presented in section 4.6.2, *Eq 2*. We now formulate the communication requirements for PAP, when PAP is used in the same manner as CNP-ext, i.e. PAP only allows bids that can fully achieve the auctioneer's announced set of tasks, precludes withdrawal of tasks, rejection of bids on submission, provisionally withdrawn bids, and backtracking, and the condition used for provisionally rejecting a provisionally granted bid in PAP is that the provisionally granted bid is no longer preferred over a newly submitted updated bid, and if a bidder's updated bid submitted to replace its withdrawn bid is provisionally granted before any other submitted bids, then the bidder in PAP is not able to withdraw the bid for the second time.

Refer to the PAP specification in section 5.1 (Figure 21). Assume a maximum of κ bidders, a maximum of *br* bids received per task announcement, and η is the number of repeated negotiations (a provisionally granted bid is rejected to select another bid) before a bid is confirm granted. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders, and therefore communicates κ messages. The protocol proceeds to step 2, where bidders submit (communicate) *br* **bids**. PAP proceeds to step 3. Since we do not consider task withdrawals, rejection on bid submission and backtracking, in the context of the analysis, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. The auctioneer **provisionally grants** its most preferred bid, and thus communicates *1* message, and the protocol proceeds to step 4, which is the point where the negotiation repeating commences (see below). In order to be consistent with CNP-ext having a different definitive bid submitted for the provisionally granted bid, the bid that is first provisionally granted in PAP is withdrawn and a new bid submitted and committed to by the same bidder. Therefore, in order to do this, the bidder **withdraws** (or provisionally withdraws) the provisional grant for its bid, then proceeds to step 2 and submits an updated **bid**, proceeds to step 3 and the auctioneer **provisionally grants** the updated bid, and then the bidder **accepts the provisional grant** of its bid (in the context of the analysis, a bidder cannot withdraw a bid consecutively more than once) – requiring *4* messages to be communicated. The protocol proceeds to step 5 because in the context of the analysis, the bid fully achieves the task, and therefore, take control option (f). To be consistent with CNP-ext (and for the worst case PAP communication), all other bidders submit updated (better) **bids** [21] (due to PAP's bidding policy, bidders may submit updated better bids at any time), and thus they will submit $br - 1$ bids at step 2, and proceed to step 3. In the context of the analysis, there are η negotiations before the provisionally granted bid is confirm granted. Therefore, one of the updated bids is better than the provisionally granted bid (as with CNP-ext), and therefore, the auctioneer will **provisionally reject** the provisionally granted bid at step 5, PAP will proceed to step 2 allowing the bidder to submit an updated **bid**, and the new preferred bid is **provisionally granted** – requiring 3 messages to be communicated. This takes PAP back to the commencement of the negotiation repeating process. If this process repeats η times, the number of messages communicated is $(4 + br - 1 + 3)\cdot\eta$. At the η^{th} repeated negotiation (at step 4 after the provisional grant), the bidder will **accept the provisional grant** (the process does not repeat) and since in the context of the analysis, the bid fully achieves the task, then it takes control option (f) to step 5. The auctioneer will then **confirm grant** the bid. Therefore, 2 further messages are communicated to finally confirm grant the bid. The total communication required by PAP to allocate a task is the sum of the number of messages communicated:

Eq 23: Total communication for PAP $= \kappa + br + 1 + (4 + br - 1 + 3)\cdot\eta + 2$

<u>Conditions for reduced communication compared with CNP-ext</u>

Using *Eq 2* and *Eq 23*, PAP requires less communication than CNP-ext if (i) $br \geq 6$; or (ii) $br = 5\ \&\ \eta \leq 1$; or (iii) $br = 4\ \&\ \eta = 0$. This follows because:

Total communication for PAP < Total communication for CNP-ext

$$\therefore \kappa + br + 3 + (6 + br)\cdot\eta < \kappa + 2\cdot br + 2\cdot br\cdot\eta$$

$$\therefore (br-6)\cdot\eta + br > 3$$

Therefore, the equation above is satisfied if (i) $br \geq 6$; or (ii) $br = 5\ \&\ \eta \leq 1$; or (iii) $br = 4\ \&\ \eta = 0$.

[21] Note that the PAP does not usually operate like this. With the PAP, bidders submit their best bid first, so only need to send an update if their bids are rejected or withdrawn. With CNP-ext, bidders may not send their best bid, so when they their bid is provisionally rejected, they may submit a better bid.

Saving in communication compared with CNP-ext

Using *Eq 2* and *Eq 23*, the saving in communication by using PAP over CNP-ext is:

Saving in comms = Total comms for CNP-ext - Total comms for PAP

$\therefore$ *Saving in comms* $= \kappa + 2\cdot br + 2\cdot br\cdot\eta - (\kappa + br + 3 + (6 + br)\cdot\eta\ )$

$\therefore$ *Saving in comms* $= br + br\cdot\eta - 6\cdot\eta - 3$

5.3.8 Memory Requirements

PAP allows an auctioneer to perform a depth-first search (Theorem 6), and as with a depth-first search (Russell and Norvig 1995), at any time, the auctioneer must store a single path in the search tree from the initial set of tasks (root node) to the current set of tasks that it is trying to achieve (leaf node), along with the remaining unexpanded bids (branches). If *br* is the maximum number of bids for a set of tasks and *m* is the depth of the search (number of bids required to achieve the initial set of tasks), then

Eq 24: Max number of bids stored by the auctioneer $= br\cdot m$

Eq 25: Max number of set of tasks stored by the auctioneer $= m + 1$

where *Eq 25* includes the initial set of tasks (root node) and the final set of tasks, which may not be empty (not fully achieved) and hence must be stored. As mentioned in section 5.3.6, each set of tasks may contain a collection of independent sub-tasks. Let λ be the maximum number of independent sub-tasks for every set of tasks, then the total number of subtasks that need to be stored is:

Eq 26: Max number of sub-tasks stored by the auctioneer $= (m+1)\cdot\lambda$

The number of bids submitted with PAP is restricted. The auctioneer receives only one bid from each bidder for its set of tasks (or each of its sub-task if $\lambda > 1$), unless an updated better bid becomes available, which for many domains may not occur often. If a bidder's bid is withdrawn or rejected, and thus deleted by the auctioneer, the bidder *might* submit an updated (replacement) bid. Hence, if κ is the number of bidders (that submit bids), then

Eq 27: Number of bids stored by the auctioneer $\approx \kappa\cdot m\cdot\lambda$

Eq 27 will approach that of *Eq 24* if $\kappa \cdot \lambda$ approaches *br*, i.e. each of the κ bidders contain only one bid per sub-task (or set of tasks if $\lambda = 1$), and thus all possible *br* bids are submitted for the set of tasks after the task announcement.

Therefore, the memory requirements to store the auctioneer's complete search tree to achieve its initial set of tasks increases with: the number of bidders (that submit bids); the number of bids required to achieve the initial set of tasks; and the number of independent sub-tasks for each set of tasks.

Bidders using PAP store a set of tasks (or its sub-tasks if $\lambda > 1$) and bids submitted for the sub-tasks. This enables bidders, for example, to: keep track of bids that are submitted for a sub-task so that they do not submit the same bid again; refer to bids and its associated sub-task if a bid is granted or rejected; check if it is possible to submit a bid for a sub-task that is better than previously submitted bids. If β is the maximum number of bids that a bidder may submit for each sub-task received, and ε is the maximum number of subtasks considered (and thus stored) by bidders at any one time, then the maximum memory requirement to store bids for bidders is

Eq 28: Maximum number of bids stored by bidders $= \beta \cdot \varepsilon$

ε considers sub-tasks received by a bidder from all auctioneers that it is negotiating with at the time. For a single auctioneer, *Eq 26* provides the number of sub-tasks sent by the single auctioneer, and thus stored by bidders. Therefore, memory requirements for a bidder to store bids for a single auctioneer is

Eq 29: Max number of bids stored by bidders for a single auctioneer $= \beta \cdot (m + 1) \cdot \lambda$

Bidder memory requirements grow with the number of sets of tasks (or sub-tasks) received and bids sent. As sub-tasks are withdrawn by the auctioneer, expired (become very old), or a bid for a sub-task is confirm granted (ending negotiations for the sub-task), the sub-task and its associated bids can be deleted, restoring memory. If there is a constant flow of sub-tasks over time, replacing deleted sub-tasks, the memory requirements should remain relatively static. If not, bidders may set aside a fixed memory limit, deleting the oldest sub-tasks and bids when the limit is reached.

5.3.9 Memory Compared with CNP, ECNP and CNP-ext

As shown below, the memory requirements for agents using PAP is greater than CNP-ext and ECNP. Memory requirements for PAP are the same as CNP, but bidders may need to hold on to the sub-task and its bid for a longer duration. The increased or prolonged memory requirements with PAP are due to: (i) bidders not receiving a rejection message informing them that the bid is not required for the sub-task; and (ii) the auctioneer storing unused bids in case they are required later during backtracking, where CNP, CNP-ext and ECNP do not allow backtracking of individual bids in order to grant a replacement bid. If agents using PAP are aware that they are performing planning and task allocation in the same manner as CNP, CNP-ext or ECNP, then auctioneers may delete bids if not required and bidders can remove tasks and bids submitted that are not provisionally granted soon after the deadline, indicating the bid was not required.

Memory Requirements of PAP compared to CNP

The best case memory requirements for CNP were presented in section 4.4.3. We now formulate the memory requirements for PAP, when PAP is used in the same manner as CNP, i.e. a single auctioneer that only allows bids that can fully achieve the auctioneer's announced set of tasks and precludes withdrawals, the updating of bids, rejection of bids and backtracking.

Refer to the PAP specification in section 5.1 (Figure 21). Assume a maximum of κ bidders and a maximum of br bids received per task announcement. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders. Therefore, the auctioneer and each bidder must store *1* set of tasks. The protocol proceeds to step 2, where bidders submit br **bids** that fully achieve the announced task, and therefore, the auctioneer must store br bids and the bidders must store their *1* submitted bid. The protocol proceeds to step 3. Since we do not consider withdrawals and backtracking in the context of the analysis, these speech acts or events (e.g. withdrawn, provisionally reject and backtracking) can be ignored at step 3. The auctioneer will **provisionally grant** its most preferred bid, and PAP proceeds to step 4. Unlike CNP, no rejection messages are sent to the bidders of the unselected bids, and hence these bidders keep the set of tasks and its bid stored in memory (until some prolonged duration of time that the bidder believes the set of tasks is no longer available, i.e. expired). The **provisional grant is accepted** because the bidder is unable to (provisionally) withdraw the bid in the context of the analysis. As the bid fully achieves the announced task, the protocol proceed to step 5 (take control option (f)). In the context of the analysis, rejection of bids are precluded, and therefore provisionally granted bid is **confirm granted**. The auctioneer may delete all the *br - 1* unused bids. The total memory requirement for the auctioneer is *1* set of tasks and *br* bids, and for the bidders are *1* set of tasks and 1 bid, which will remain in memory (until expired) if the bid is not used.

The total memory requirement with CNP for the auctioneer is *1* set of tasks and *br* bids, and for the bidders are *1* set of tasks and 1 bid, which is deleted if/when their bid is rejected. Therefore, both CNP and PAP require the same amount of memory if applied in the same way, except that the bidders that do not have their bid granted in PAP and will have to keep the set of tasks and its bid stored in memory for a longer period of time.

Memory Requirements of PAP compared to ECNP

The best case memory requirements for ECNP were presented in section 4.8.3. We now formulate the memory requirements for PAP, when PAP is used in the same manner as ECNP, i.e. a single auctioneer that precludes withdrawals, the updating of bids, the rejection of bids on submission, and backtracking only when a plan is found and deemed unsuitable.

Refer to the PAP specification in section 5.1 (Figure 21). Assume *br* is the maximum number of bids submitted for each announced set of tasks, κ is the number of bidders that submit bids and *m* is the depth of the search (number of bids required to achieve the auctioneer's initial set of tasks). At step 1, the auctioneer **announces** its set of tasks to each of the κ bidders. Therefore, the auctioneer and the κ bidders store the *1* set of tasks. The protocol proceeds to step 2, where bidders submit *br* **bids**, and hence the auctioneer stores *br* received bids and the bidders store their *1* submitted bid. PAP proceeds to step 3. Since we do not consider withdrawals, rejection on bid submission and backtracking until the end, in the context of the analysis, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. The auctioneer will **provisionally grant** its most preferred bid. Since there are no rejection messages in PAP, and due to the persistence policy, bidders and the auctioneer keep unused bids and set of tasks in memory. Therefore, at this stage, the κ bidders have a total maximum of κ set of tasks and κ bids stored in memory. PAP proceeds to step 4 where the **provisional grant is accepted** because the bidder is unable to (provisionally) withdraw the bid in the context of the analysis. If the bid does not fully achieve the set of tasks announced by the auctioneer, then PAP proceeds to step 1 (control option (e)) to announce the remaining set of tasks that the bid did not achieve, and the same process is repeated. For the next announced set of tasks, again the auctioneer will send its set of tasks, receive *br* bids, and therefore the auctioneer will have *2* sets of tasks and $2 \cdot br$ bids in memory (includes previous unused bids). The bidders will have stored the new sets of tasks and its bids, in addition to their previous set of tasks and its bid (which are not deleted). Therefore, the bidders have stored together $2 \cdot \kappa$ set of tasks and $2 \cdot \kappa$ bids. Again, if the bid does not fully achieve the set of tasks, the process repeats, until the m^{th} bid's provisional grant is accepted at step 4 which does fully achieve the set of tasks announced by the auctioneer. The auctioneer would have a maximum of *m* sets of tasks and $br \cdot m$ bids stored in memory. The total maximum memory required for all the κ bidders together is $\kappa \cdot m$ sets of tasks and $\kappa \cdot m$ bids. The protocol proceeds to step 5 (take control option (f)) where the auctioneer will either send all *m* provisionally granted bids a **confirm grant** (if the plan is suitable) or a **provisional reject** (if the plan is unsuitable – refer to Theorem 2 proof for detailed discussion on how this occurs). Therefore, the total maximum memory required for the auctioneer is *m* sets of tasks and $br \cdot m$ bids, and the total maximum memory requirements for all the κ bidders together is $\kappa \cdot m$ sets of tasks and $\kappa \cdot m$ bids.

Memory differences

The auctioneer using PAP stores the same number of set of tasks as an auctioneer using ECNP, which is *m* set of tasks. An auctioneer using PAP must store $br \cdot m$ bids, which is greater than the $br - m + 1$ bids that the auctioneer using ECNP must store. The extra bids that must be stored is:

Eq 30: Extra bids auctioneers must store $= br \cdot m - (br - m + 1) = (br + 1) \cdot (m - 1)$

In total, bidders using PAP must store, *all together*, $\kappa \cdot m$ set of tasks and $\kappa \cdot m$ bids, which is greater than the $\kappa + m - 1$ set of tasks and $\kappa + m - 1$ bids stored by bidders using ECNP (as $\kappa > 0$ *and* $m > 0$). The extra set of tasks and bids that must be stored is:

Eq 31: Extra set of tasks and bids stored by bidders $= \kappa \cdot m - (\kappa + m - 1)$

$$= (\kappa + 1)\cdot(m - 1)$$

The auctioneer and bidders using PAP require more memory than the auctioneer and bidders using ECNP. Therefore, PAP requires more memory than ECNP.

Memory Requirements of PAP compared to CNP -ext

The best case memory requirements for CNP-ext were presented in section 4.6.3. We now formulate the memory requirements for PAP, when PAP is used in the same manner as CNP-ext, i.e. PAP only allows bids that can fully achieve the auctioneer's announced set of tasks and precludes withdrawal of tasks, rejection of bids on submission, provisionally withdrawn bids, and backtracking, and the condition used for provisionally rejecting a provisionally granted bid in PAP is that the provisionally granted bid is no longer preferred over a newly submitted updated bid, and if a bidder's updated bid submitted to replace its withdrawn bid is provisionally granted before any other submitted bids, then the bidder in PAP is not able to withdraw the bid for the second time. We define *c1* as the number of bids that are withdrawn and its updated bid provisionally granted in PAP and *c2* as the number of times a provisionally granted bid is provisionally rejected and another bid selected.

Refer to PAP specification in section 5.1 (Figure 21). Assume a maximum of κ bidders and a maximum of *br* bids received per task announcement. At step 1, the auctioneer sends out a **task announcement** to each of the κ bidders. Therefore, the auctioneer and the κ bidders store *1* set of tasks. The protocol proceeds to step 2, where bidders submit *br* **bids**, and hence the auctioneer stores *br* received bids and the bidders store their *1* submitted bid. PAP proceeds to step 3. Since we do not consider task withdrawals, rejection on bid submission and backtracking, in the context of the analysis, these speech acts or events (e.g. withdrawn, reject and backtracking) can be ignored at step 3. The auctioneer **provisionally grants** its most preferred bid, and the protocol proceeds to step

4. To be consistent with CNP-ext, all other bidders associated with the *br – 1* bids that were not provisionally granted, send *br - 1* updated (better) bids [22] (due to PAP's bidding policy, bidders may submit updated better bids at any time). At this stage, the auctioneer has *1* set of tasks and *2·br – 1* bids, and the bidders together, have a total of *1* set of tasks and *2·br – 1* bids. There are three options: (i) the provisionally granted bid is **withdrawn** at step 4; (ii) the **provisional grant is accepted** at step 4, but there exists an updated bid which is better than the provisionally granted bid; and (iii) the **provisional grant is accepted** at step 4 and the provisionally granted bid is better than all updated bids.

Case (i): The protocol will proceed to step 2 for the bidder that withdrew its bid to send an updated bid. The auctioneer and bidder store the extra *1* bid. The auctioneer then **provisionally grants** its new preferred (updated) bid at step 3, and then the bidder **accepts the provisional grant** (in the context of the analysis, a bidder cannot withdrawn a bid consecutively more than once). Now we have case (ii) and (iii) above, and the number of bids increased for the auctioneer and all the bidders by *1*.

Case (ii): After the provisional grant is accepted, because in the context of the analysis the bid fully achieves the task, the protocol will proceed to step 5. The auctioneer will provisionally reject the provisionally granted bid, and proceed to step 2 for the rejected bidder to send *1* updated **bid**. The auctioneer will then **provisionally grant** its new preferred updated bid at step 3, and proceed to step 4. Again, the bidders of the *br – 1* bids that did not get a provisional grant will send *br – 1* updated bids. We now have case (i), (ii) or (iii) above, and the number of bids increased for the auctioneer and all the bidders by *br*.

Case (iii): After the provisional grant is accepted, because in the context of the analysis the bid fully achieves the task, the protocol will proceed to step 5. The auctioneer will **confirm grant** the provisionally granted bid, which is still its preferred bid. The protocol completes. No extra bids of set of tasks are stored.

[22] Note that the PAP does not usually operate like this. With the PAP, bidders submit their best bid first, so only need to send an update if their bids are rejected or withdrawn. With CNP-ext, bidders may not send their best bid, so when they their bid is provisionally rejected, they may submit a better bid.

If c1 represents the number of times case (i) occurs and c2 represents the number of times that case (ii) occurs, then the auctioneer stores *1* set of tasks and $2 \cdot br - 1 + c1 + c2 \cdot br$, and the bidders store together, in total, *1* set of tasks and $2 \cdot br - 1 + c1 + c2 \cdot br$.

Memory differences

The auctioneer using PAP stores the same number of set of tasks as an auctioneer using CNP-ext, which is *1* set of tasks. An auctioneer using PAP must store $2 \cdot br - 1 + c1 + c2 \cdot br$ bids, which is greater than the $br + 1$ bids that the auctioneer using CNP-ext must store. The extra bids that must be stored is:

Eq 32: Extra bids auctioneers must store $= 2 \cdot br - 1 + c1 + c2 \cdot br - (br + 1)$

$$= br + c1 + c2 \cdot br - 2$$

In total, bidders using PAP store, *all together*, the same number of set of tasks as CNP-ext, which is *1* set of tasks, but the number of bids for PAP is $2 \cdot br - 1 + c1 + c2 \cdot br$, which is greater than the $br + 1$ bids stored by bidders using CNP-ext. The extra bids that must be stored is:

Eq 33: Extra bids bidders must store $= 2 \cdot br - 1 + c1 + c2 \cdot br - (br + 1)$

$$= br + c1 + c2 \cdot br - 2$$

The auctioneer and bidders using PAP require more memory than the auctioneer and bidders using CNP-ext. Therefore, PAP requires more memory than CNP-ext.

5.3.10 Convergence and Livelock

Theorem 7 below presents the two conditions required for PAP to terminate (assuming no agent failures or lost messages), and thus PAP will converge to either a solution or no solution, and prevent livelock. The first condition is that the auctioneer should grant each bid a finite number of times. As discussed previously, in our current PAP implementation, we only allow bids to be provisionally granted once, and therefore once a bid is provisionally rejected or (provisionally) withdrawn, they are discarded and not used again. This allows PAP to perform the same planning as a depth-first search. As will be discussed in following sections, there may be benefits in allowing the auctioneer to provisionally grant a bid more than once. This is the subject of future work.

The second condition requires the number of bids sent or received to be finite. Our two application domains that we will discuss contain finite bids, and therefore this condition is satisfied. This may not be the case for all domains, for example, a bidder could supply infinite bids for services of duration *ds* in an empty schedule from time zero to infinity. If this occurs, then we could place two restrictions on receiving bids. We could limit the number of bids received for each individual set of tasks announced, and limit the overall number of bids received for all the auctioneer's set of tasks used to achieve its initial set of tasks. If the limit is reached, the auctioneer is not able to accept any more bids and a solution must be found with the bids already submitted. Both limits effectively places a restriction on the depth of the (depth-first) search that PAP can perform, which is a commonly known termination problem with the depth-first search (Russell and Norvig 1995).

Theorem 7: PAP will guarantee to terminate, and thus converge to either a solution or no solution and prevent livelock, if (a) there are a finite number of bids that may be sent or received and (b) the auctioneer provisionally grants each bid for a set of tasks only a finite number of times. We assume no agent failures or lost messages.

Proof:

Refer to the PAP specification in section 5.1 (Figure 21). The three control flows in PAP that are of concern for convergence are: **(1)** the *rejecting or withdrawing of bids* in steps 3, 4 and 5, moving the protocol process back to step 2; **(2)** the *provisional grant* in step 4, where the bid does not achieve a complete task, starting a new protocol process (e); **(3)** *backtracking* in step 3 that moves the protocol execution back to the previous protocol process. Other speech acts either exit the protocol process, or move the protocol process to a subsequent step, and thus will eventually cause the protocol to complete (confirm grant in step 5).

Convergence for conditions (1), (2) and (3) above is as follows:

Case (1): The two conditions that ensure an infinite loop does not occur between step 2 and steps 3, 4, and 5 is:

(a) Only a finite number of bids should be sent **or** received at step 2, thus only a finite number of new bids can be rejected (particularly at step 3) or withdrawn.

(b) The auctioneer should grant each of the finite number of bids a finite number of times, otherwise, the auctioneer may indefinitely persist in granting the same bid(s), which will be consequently rejected or withdrawn.

Case (2): It may be possible that bids are granted indefinitely at step 4, never completely achieving the task (a search tree of infinite depth), resulting in an infinite sequence of protocol processes. This is more likely if bids are selected that move the auctioneer further away from its set of tasks [23]. Convergence can be ensured if either:

(a) Only provisionally granted bids that move the auctioneer closer (not infinitesimally closer) to achieving its set of tasks. Therefore, eventually, the auctioneers set of tasks will be achieved.

(b) There are a finite number of bids (or resources) that could be sent **or** received for the set of tasks. Once bids are depleted, then PAP will need to either accept the current plan or backtrack.

(c) Limit the number of bids that can be provisionally granted for a set of tasks, i.e. limit the depth of the search tree.

Case (3): PAP may alternate between protocol processes indefinitely – if the auctioneer continually backtracks, and then selects the same bid in the previous protocol process that caused backtracking. This can be prevented if the same bid is only granted a finite number of times.

If *sfb* is true if there is a finite number of bids submitted, *gfb* is true if the auctioneer grants each bid a finite number of times, *mc* is true if only bids are selected that move the

[23] For example, in our transportation domain, bids may be selected that move resources further away from its required destination (which may be required for a solution). If this occurs indefinitely, then the resources will move further and further away from its destination (the achievement of the task), and never reach it.

auctioneer closer to its set of tasks, and *ld* is true of the depth if the search is limited, then from cases (i) to (iii) above, which all must be satisfied, we have:

Eq 34: $(sfb \wedge gfb) \wedge (mc \vee sfb \vee ld) \wedge gfb$

$= (sfb \wedge gfb \wedge mc) \vee (sfb \wedge gfb) \vee (sfb \wedge gfb \wedge ld)$

$= \boldsymbol{sfb \wedge gfb}$

Therefore, from *Eq 34*, the conditions for convergence, and hence termination and livelock prevention, is to have a finite number of bids sent or received and provisionally grant each bid only a finite number of times.

Q.E.D.

5.3.11 Deadlock

Theorem 8 below shows that PAP will not get caught in deadlock. If any agent fails or messages between agents are lost during PAP, the protocol will either proceed or exit or not commence at all. No agent will be bound to PAP negotiation/process indefinitely such that they need to wait indefinitely for a message to be submitted in order to continue or be released from negotiations.

Although PAP does not deadlock, a bidder may fail just before a final confirm grant message is sent for its provisionally granted bid, or the confirm grant message may be lost. The auctioneer will exit the protocol assuming that the bidder will perform its bid, which it may not. CNP-ext is able to detect such cases, but requires extra communication overhead as each bidder must communicate with all other bidders regarding the auctioneer's decisions on their tasks. This may not be possible as bidders may not want to release this information, the auctioneer may not want to release information regarding other bidders it is negotiating with or the system may be very large requiring many messages to be communicated. The bidder could have failed after the confirm grant was received, resulting in the bidder also not performing its task. Therefore, in order to ensure a bidder receives the message, a suitable network level communication protocol (e.g. TCP/IP) should be used to send the confirm grant message and acknowledge that the message was received by the bidder. To ensure an agent is able to respond if the other agent fails to perform or achieve its bid, another protocol (or extension to PAP) may be required to inform the auctioneer of the execution, progress and/or completion of a granted bid, and to perform replanning when failure does occur. PAP is currently only concerned with the planning and task allocation problem.

Theorem 8: PAP does not fall into deadlock if an agent fails or a message is lost during the negotiation.

Proof:

In order to prove that deadlock does not occur, we show that for the auctioneer and bidder involved in negotiations that if the other fails or a message is lost, then they will proceed to another step in PAP or exit or not commence PAP at all, and will not be bound to PAP negotiations indefinitely (i.e. not waiting indefinitely for a message from another agent in order to be released from negotiations).

Refer to the PAP specification in section 5.1 (Figure 21).

Step 1:

- *Bidders:* If the auctioneer fails, then there is no **task announcement**, and therefore, no negotiation takes place and PAP does not commence. If the **task announcement** message is lost to one or more bidders, then these bidders will not be involved in the negotiation and the auctioneer will only receive bids (enter into negotiations) from those that receive the task announcement and submit bids (bidders using PAP do not need to submit bids or require any communication with the auctioneer after a task announcement – **no communication** at step 2). PAP proceeds to step 2 with bidders that did receive the task announcement. If all the auctioneer's task announcement messages are lost, then no bidder will receive the message and no bids will be received by the auctioneer at step 2 (see step 2).
- *Auctioneer:* If messages are lost to one or more bidders, or they fail, then they are not able to accept the **task announcement**. As a result, negotiations (PAP) do not commence for these bidders.

Step 2:

- *Bidders:* If messages to the auctioneer are lost or the auctioneer fails, then the all the bidders' **bids** will not be received by the auctioneer, but will still be sent by the bidders. The bidders will proceed to step 3. Due to PAP's commitment policy, bidders are not committed to their bids, and therefore are not bound in any way to the negotiation at this step. At step 3 all bidders will exit the protocol with the **no communication** event, assuming that the auctioneer is not interested in their bids.
- *Auctioneer:* If messages from the bidders are lost or one or more bidders fails, then the auctioneer will not receive **bids** (or updated bids if arrived at this step from a following step) from these bidders. The auctioneer still proceeds to step 3 after the bidding deadline. If all messages are lost or all bidders fail in the initial task announcement, then the auctioneer will receive no bids and proceed to step 3 (**no communication** event and control option (b) at step 2) after the bidding deadline and **backtrack** at step 3, resulting in the auctioneer assuming the task is unachievable. The auctioneer therefore exits if the set of tasks it is trying to backtrack from is the initial set of tasks, or proceeds to step 5 of the previous protocol process.

Step 3:

- *Bidders:* If the auctioneer fails or messages (provisional grant, provisional reject, or withdrawn messages) to the bidders are lost, then we have a similar case to step 2. Bidders have a bid submitted but since bidders are not committed to their submitted bids, they are not bound in any way to the negotiation at this step. Hence they may exit the protocol using the **no communication** event at step 3 if no further messages are received regarding their bid.
- *Auctioneer:* If bidders other than those that are having their bid **provisionally granted**, **withdrawn** or **provisionally rejected** fails, then the auctioneer will proceed with the protocol as it does not send messages for (or use) these bids. A lost message or failure of a bidder which is sent a **provisional reject** message results in PAP for the auctioneer to proceed to step 2 and not receive an updated bid from the bidder, in which case the auctioneer will proceed to step 3 (**no communication**, control option (b) at step 2) and continue with its current

submitted bids. A lost message or failure of a bidder which is sent a **withdrawn** message has no effect on the auctioneer because a withdrawn message is sent when the negotiation (protocol process) has already exited for the auctioneer. A lost message or failure of a bidder which is sent a **provisional grant** message results in the protocol proceeding to step 4 where the auctioneer waits for a response from the bidder before the provisional grant acceptance deadline (*pgad*). When no messages are received from the bidder by *pgad*, then the auctioneer assumes that the bid is withdrawn and proceeds back to step 2, where again, no update is sent by the bidder and the auctioneer proceeds to step 3 with its current bids. If all bidders fail or all **provisional grant** messages are lost, then the auctioneer will try to provisionally grant all its submitted bids, and receive no response by *pgad*, and hence assume they are withdrawn. After provisionally granting its last bid, the auctioneer will proceed to step 2 and receive no updates before proceeding to step 3. The auctioneer will assume no bids are available for its set of tasks and **backtrack** at step 3.

Step 4:

- *Bidders:* If the auctioneer fails or does not receive messages, then the auctioneer would not receive a *provisional grant acceptance* message or **(provisional) withdrawn** message from the bidder that had its bid provisionally granted. If a **(provisional) withdrawn** message is sent, then the bidder is informing the auctioneer that its bid is not available. Thus, the bidder is not committed to the bid or bound to the negotiation in any way. The bidder will proceed to step 2 where it may send an updated bid (see step 2). If a **provisional grant accepted** message is sent, then according to PAP's commitment policy, the bidder is committed to the bid, and proceeds to step 5 of the protocol. The bidder waits for a **confirm grant** or **provisional reject** message, but will only wait as long as the confirm grant deadline (*cgd*). Therefore, when the auctioneer does not send one of the required messages to the bidder by *cgd*, the bidder assumes the auctioneer provisionally rejects the bid and proceeds to step 2 where it may send an updated bid (see step 2).

- *Auctioneer:* If the bidder that has its bid **provisionally granted** fails or has its messages lost, then as with the previous step, the auctioneer waits for a response from the bidder before the provisional grant acceptance deadline (*pgad*). When no messages are received from the bidder by *pgad*, then the auctioneer assumes that the bid is withdrawn and proceeds back to step 2 (see step 2).

Step 5:

- *Bidders:* If the auctioneer fails or has its messages lost, then as with the previous step, the bidder waits for a **confirm grant** or **provisional reject** message, but will only wait as long as the confirm grant deadline (*cgd*). Therefore, when the bidder does not receive one of the required messages by *cgd*, the bidder assumes the auctioneer provisionally rejects the bid and proceeds to step 2, where it may send an updated bid (see step 2).
- *Auctioneer:* If the bidder with the provisionally granted bid fails while a **provisional reject** message is sent, or the message is lost, then because the auctioneer did not require the bid from the bidder anyway, it proceeds to step 2 (see step 2) and continues negotiations. If the bidder with the provisionally granted bid fails while a **confirm grant** message is sent, or the message is lost, then there is no effect on PAP for the auctioneer because after a confirm grant is sent, the negotiation (protocol process) for this set of tasks exits for the auctioneer.

We have shown that at each step in PAP, if an auctioneer or bidder fails or loses messages, they will proceed to another step in PAP or exit or not commence PAP at all, and will not be bound to PAP negotiations indefinitely (i.e. not waiting indefinitely for a message from another agent in order to be released from negotiations). Therefore, there is no deadlock with PAP.

Q.E.D.

5.4 Summary

This chapter presented PAP, its features and formal analysis of the protocol[24]. PAP is able to overcome shortfalls with ECNP by allowing backtracking, multiple auctioneers (facilitating planning and task allocation in a many-to-many agent setting) and planning and task allocation a dynamic environment where bids and tasks may come and go during the planning process. Therefore, PAP facilitates planning and task allocation in decentralised, open and dynamic environment with a many-to-many setting (many auctioneers and many bidders interacting simultaneously).

PAP has greater flexibility with planning and task allocation than CNP, CNP-ext and ECNP, as it is able to perform the planning and task allocation that they are able to, but the converse is not true. PAP is able to facilitate a decentralised depth-first search (with a dynamic search tree), which from our knowledge, has not been previously done. PAP has reduced communication than CNP, CNP-ext and ECNP if bids submitted for each announced set of tasks are greater than 3, 2, and 6, respectively. We believe this condition is satisfied in many real world applications, such as the two applications we present in the next two chapters. Although PAP requires less communication than these protocols, PAP requires more memory. Therefore, PAP would be the protocol of choice if communication costs are high and memory costs are low. PAP is able to take advantage of distributed processing, is predominantly consistent with (legal) contracting, and reduces broken contracts. We have also presented the conditions for PAP to converge and thus prevent livelock, and showed that PAP does fall into deadlock.

In the next two chapters, we have applied PAP to the combinatorial auction domain and to the transportation scheduling domain. Applying PAP to two different domains shows the protocol generality.

Other domains for which PAP is suited are coalition formation and virtual enterprises (see chapter 2). Each task t_i in the set of tasks $T_j = \{t_1, \ldots, t_n\}$ announced by an auctioneer

[24] The proofs of protocol properties could also have been carried out using formal logical tools, which is the subject of future work.

can refer to services required to achieve some business goal T_j. PAP enables a reverse auction to find a set (coalition) of agents that can provide suitable services in order to achieve the goal, within the complexity of an open market, which is decentralised, dynamic, open, and has a many-to-many setting. In the transportation application discussed in chapter 7 and (Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004), the services t_i to achieve the goal T_j are not known, and thus PAP can be used to solve the planning problem of what services are required to achieve the business goal (which is dependent on the services available at the time), in addition to the allocation problem of who should perform the services.

Note that Simulation Trading (ST) (see section 4.7.3) could be used in addition to PAP to further optimise the solution found by PAP (similar to how ST is used in addition to ECNP to further optimise the solution found by ECNP). Although ST was developed for cooperative agents where TA would accept any proposal for the redistribution of tasks from the MA, ST can be extended to the non-cooperative setting (a characteristic of our domain) when side payments are introduced.

Chapter 6

6 Combinatorial Auctions

The development of PAP initially evolved from addressing the (MALT's) transportation scheduling problem. Due to the transportation scheduling application's complexity, it was difficult to separate the protocol from the application in order to experimentally evaluate the protocol on its own. In this chapter, we use combinatorial auctions as an application to evaluate PAP (Perugini, Lambert et al. 2005), before discussing the transportation scheduling application in the next chapter. In addition to experimentally evaluating the behaviour of the protocol, we present benefits that PAP has over current one-shot (centralised single auctioneer) combinatorial auction approaches, and PAP's ability to facilitate the novel multiple simultaneous combinatorial auction problem.

6.1 Combinatorial Auctions Domain

6.1.1 Problem Description

In the typical combinatorial auctions domain (Nisan 2000; Sandholm 2002; Cramton, Shoham et al. 2006) an auctioneer α must allocate a set of non-identical goods $G = \{g_1, \ldots, g_n\}$ to bidders $\gamma_j \in \Gamma$, which contain a set of bids $bids(\gamma_j) = \{b_1, \ldots, b_m\}$, and bidders may submit bids b_j for a portion of the goods $(b_j \subseteq G)$ for price p_j. Typically there is free disposal, so not all goods need to be allocated, and each good can only be allocated once. The aim is for the auctioneer to find an allocation of bids that maximises its price. This problem is equivalent to the set packing problem, and this optimisation problem is known to be NP-complete (Garey and Johnson 1979).

The problem description above we refer to as the (well-known) single auction with static bids problem. Only one auction occurs at a time and the bidder's bids that are available for the auctioneer does not change throughout the auction (planning) process. There are two other problems which we implemented solutions to using PAP, which from the literature on combinatorial auctions using agents that we have are aware of, has not been investigated (see related work at the end of the chapter). The first is a single auction with dynamic bids. Using the problem description given above, we have a set of bidders (and thus their bids) which may come and go throughout the auction process, so $\gamma_j \in \Gamma_\tau$, where τ represents time, and we also have bidders bids (at times τ when they are available) changing with time, $bids(\gamma_j, \tau) = \{b_1, \ldots, b_m\}$. Therefore, from the auctions (or auctioneers) perspective, bids are dynamic – new bids arrive and old bids are withdrawn while the auction is performed.

The second unique problem we investigate is multiple simultaneous combinatorial auctions (many-to-many auction setting, Figure 30), which is inherently dynamic because during any one particular auction, bids may be accepted by other auctioneers. Therefore, the problem description above changes to: $\alpha_i \in \Lambda_\tau$, where Λ_τ represents the set of auctioneers that perform a combinatorial auction simultaneously at time τ, in which they are auctioning goods $G_{\tau,i} = \{g_1, \ldots, g_n\}$, and bidders $\gamma_j \in \Gamma_\tau$, are used to allocate the goods using bids $bids(\gamma_j, \tau) = \{b_1, \ldots, b_m\}$, where bidders may submit bid $b_{\tau,j}$ at time τ for a portion of the goods $(b_{\tau,j} \subseteq G_{\tau,i})$ for price $p_{\tau,j}$.

In auctions, a bidder may have a large quantity of bids to communicate, or it may not want to release all its bids as this may release private information about its intentions. Bids may have complex dependencies that may not be easily described with the OR-of-XOR language (Nisan 2000). With multiple auctions, it is not clear how to deal with dependencies between bids in different auctions, e.g. if a bid in one auction is allocated, then a bid in another auction cannot be. Therefore, it may not be practicable for bidders to submit all their bids, and the dependencies between them, to auctions for processing, which is assumed in much of the current (one-shot auction) literature (Andersson, Tenhunen et al. 2000; Hunsberger and Grosz 2000; Nisan 2000; Walsh, Wellman et al. 2000; Collins, Bilot et al. 2001; Collins, Ketter et al. 2002; Sandholm 2002; Sandholm, Suri et al. 2005). We discuss in the next section how PAP addresses these issues.

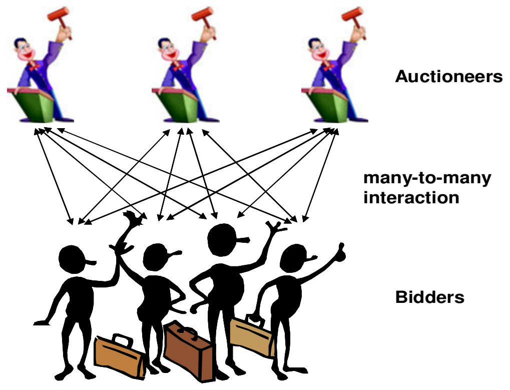


Figure 30. Many-to-many interaction with Multiple Simultaneous Combinatorial Auctions.

6.1.2 PAP with Combinatorial Auctions

In applying PAP to the combinatorial auction problems discussed in the previous section, the auctioneer's task announcement is the set of goods that the auctioneer is trying to achieve. For example, a task announcement could be $G = \{1, 2, 3, 4, 5, 6\}$, where the auctioneer is trying to allocate (or sell) non-identical goods labelled *1* to *6*. The auctioneer also submits its bid evaluation function *f*, informing the bidders of how it will evaluate the bids, and thus allows bidders to submit their single best bid. Bidders in PAP may respond with bids to fully achieve or partially achieve the task. For example, a bidder γ_j may submit bid $b_j = \{1, 2, 3, 4, 5, 6\}$ for price $p_j = \$100$ to fully achieve the task, or $b_j = \{1, 4, 6\}$ for price $p_j = \$35$ to partially achieve the task. If the auctioneer provisionally grants a partial bid for the allocation of its task, then the remaining task (set of goods) that is not achieved is re-announced in a new protocol process. For example, if bid $b_j = \{1, 4, 6\}$ was selected for G, then the remaining task that will be announced is $\{2, 3, 5\}$, and the PAP process continues.

Allowing the auctioneer to provide details to the bidder regarding how the bids will be evaluated has a few benefits over single-shot combinatorial auctions. Since bidders are able to determine which bid will be preferred by the auctioneer, and understand the (potentially complex) dependencies between its own bids, they can process the bids themselves, and only need to submit the one best bid. Therefore, this eliminates the need for the bidder to send all of its bids and their dependencies, as required in single-shot combinatorial auctions, where the dependencies can be complex, difficult to define and involve private information. PAP also pushes the bid processing onto the many distributed bidders, rather then one centralised agent (auctioneer), improving scalability with the number of bidders (see section 6.3.1).

6.2 PAP Implementation

The auctioneer and bidder agents were implemented in the ATTITUDE multi-agent architecture. Up to 10 auctioneers and 100 bidder agents were executed in a scenario. Due to limited resources, each scenario was executed on a single computer (Pentium 3.6GHz), and over 1000 scenarios were executed. To minimise the time to execute each scenario, rather than fix a bidding deadline *d*, the auctioneer continued with planning once all bids and updates were received by bidders. We assume perfect communication, and communication time (latency) in our experiments is negligible. Since there is free disposal, for the scenarios without backtracking, the first (greedy) solution found in which no more bids can be allocated is taken as the final solution, whether all the goods in the task are allocated or not.

The data we used was extracted from the Combinatorial Auction Test Suite (CATS) software (Leyton-Brown, Pearson et al. 2000). CATS produced various types of data. *Paths* and *scheduling* data were arbitrarily selected. Paths data aims to simulate bidding on paths in space, such as truck routes and network bandwidth allocation. Scheduling data aims to simulate bidding in job-shop scheduling domain. The data generated had between 10 to 1000 goods, and 10 to 1000 bids. A bidder may have more than one bid, and can have allocated at most only one of each good. Bids contain dummy goods, which are imaginary goods primarily used to define XOR relationships between generated bids. If a bidder has a bid allocated by an auctioneer which contains a dummy good, then the bidder is unable to use (or submit) any other bids it has with that dummy good as it has already been allocated.

The auctioneers used a simple heuristic for *f*, which is similar to the heuristic that Dang and Jennings used for combinatorial auctions (Dang and Jennings 2002).

$$f = \frac{number \quad of \quad goods \quad in \quad bid}{price \quad of \quad bid}$$

It was not our intention to focus on the heuristics and how well they performed (at the moment). We instead focused on the behaviour of the protocol and its benefits, by altering certain variables and performing comparisons.

6.3 Single Auction, Static Bids

Single auction with static bids comprised one auctioneer and bids distributed between a number of bidders. Updated better bids are not sent with static bids.

6.3.1 Scalability with Time

Formally, the time for an auctioneer to find an allocation with PAP, *without backtracking*, depends on the number of iterations of PAP, or the ***depth*** of the search, which is at most equal to the number of goods g in the initial goal G (g bids of one good are selected to achieve the complete goal), and the bidding deadline d for each iteration. The worst case time is $twc = g \cdot d$. d depends on the time taken for a bidder to process its β candidate bids (is $O(\beta)$ in our implementation to calculate f for each bid) plus communication time tc to send/receive the bids. κ bidders send a bid of length l bits to an auctioneer with bandwidth w bits/sec, thus $tc = l \cdot \kappa / w$ sec (assume communication travel time, or latency, is constant), hence tc is dependent on κ. The time complexity is therefore $O(g \cdot \beta \cdot \kappa)$, which is not exponential.

Results from Figure 31 and Figure 32 show that time scales linearly with the number of bids and goods in our experiments, as expected. Figure 31 assumes that deadline d increases with the number of bids to process, which in our experiments with one processor and a deadline equal to the time for all bids to be processed, this is the case. In an ideal distributed setting, d should remain constant, as more bids may be the result of more bidders with the same number of bids, so it will take the same time to process the bids, and thus the graph in Figure 31 would be flat.

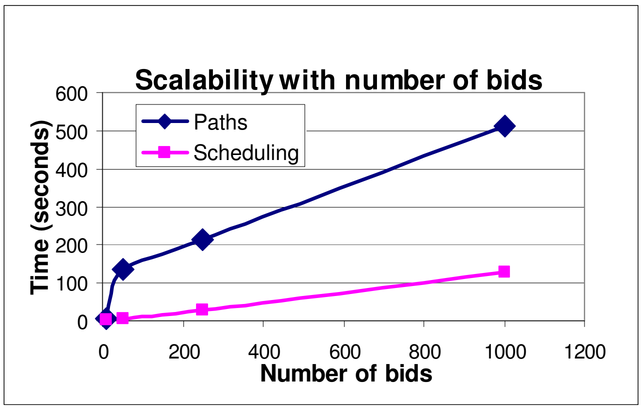


Figure 31. Scalability of time with PAP implementation with the number of bids (20 goods).

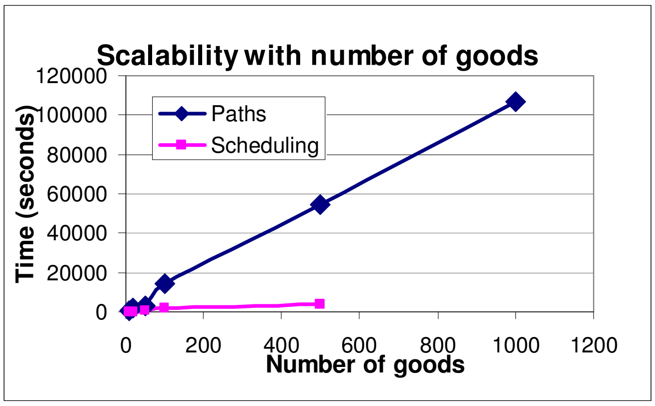


Figure 32. Scalability of time with PAP implementation with the number of goods (1000 bids). Could not obtain scheduling data with 1000 goods.

We experimented with different numbers of bidders. Since communication costs in our experiments are negligible, PAP was not affected by the bandwidth or delay. Since the bidders were all running on one processor, there was extra computational overhead as we increased the number of agents, as shown in Figure 33. Ideally, the time should decrease, because if the same number of bids are distributed among more agents, then it is likely that each agent will have less bids to process, and therefore the auctioneer can reduce *d*.

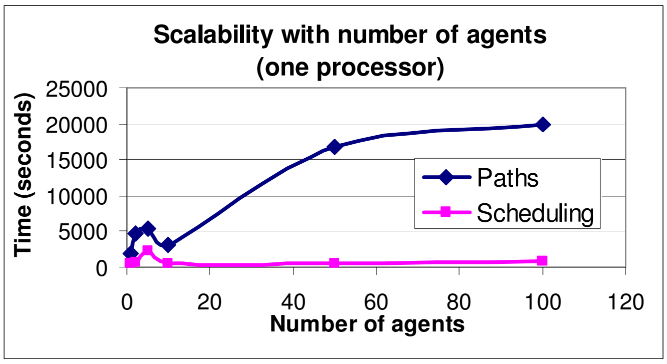


Figure 33. Scalability of time with PAP implementation with number of agents (50 goods and 1000 bids).

6.3.2 Scalability with Memory

The worst case number of bids that the auctioneer must hold in memory is $mb = g{\cdot}\kappa$, where κ bidders submit one bid for each of the g iterations of the protocol, which is linear. Figure 34 and Figure 35 show our experimental results. Figure 35 flattens out as the number of goods increase because the depth of the search remained constant, due to bids with greater number of goods and more goods in the goal unallocated in the final solution (or plan). If $\mathrm{B} = \sum_{p=1}^{\kappa} \beta_p$, is the total number of bids held by all κ bidders, in current one-shot (centralised) approaches to combinatorial auctions, all bids are submitted to the auctioneer. Therefore PAP guarantees to use less memory if $g{\cdot}\kappa < B$.

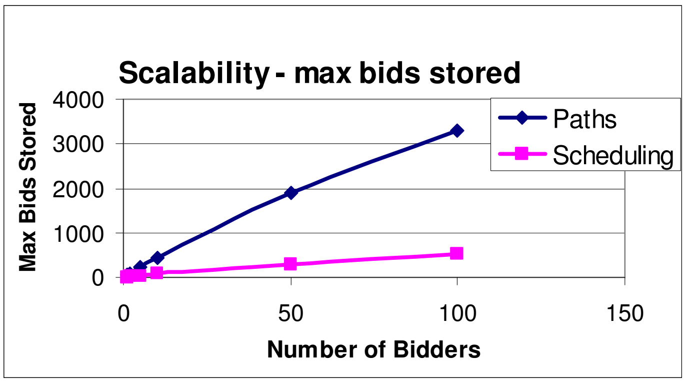


Figure 34. Scalability of memory with PAP with the number of bidders (50 goods, 1000 bids).

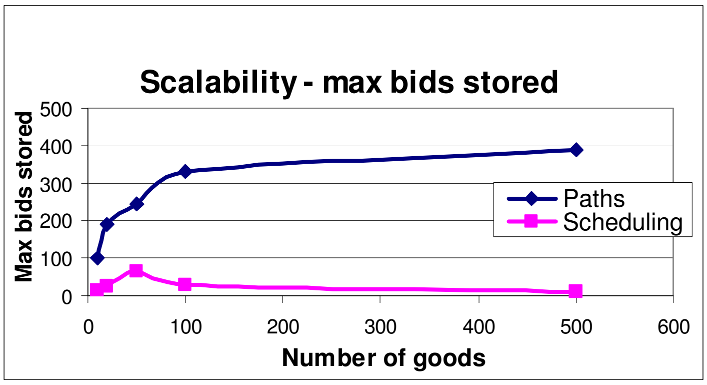


Figure 35. Scalability of memory with PAP with the number of goods (250 bids, 10 bidders).

6.3.3 Communication

The amount of communication required in PAP, without backtracking and assuming all bids received are suitable (not rejected), is $g{\cdot}\kappa + g{\cdot}\kappa + g + g + g$ for the task announcement, bids received, provisional grant, provisional grant accepted, and confirm grant, respectively, which is $g{\cdot}(2{\cdot}\kappa{+}3)$. Current one-shot combinatorial auction approaches require all B bids (see previous section) from all κ bidders to be received and assume that *all* bids need to be either granted or rejected after the auction completes. PAP requires less communication if $g{\cdot}(2{\cdot}\kappa{+}3) < 2{\cdot}B$. Therefore, in particular, PAP requires less communication if each bidder possesses a large number of bids. In our transportation application discussed in the next chapter, bidders bid for transportation along routes. There are many routes and each route has many start times. Therefore, the number of potential bids for each bidder is *extremely* large. PAP in such a case would be beneficial over one-shot approaches.

Figure 36, Figure 37 and Figure 38 show our experimental results for paths data, which is consistent with the theory. PAP communication improves over centralised when: Figure 36, the number of bidders decrease for the same number of bids; Figure 37, as goods decrease (once again, the graph flattens as goods increase because the search depth remains constant); Figure 38, the number of bids increases for the same number of bidders.

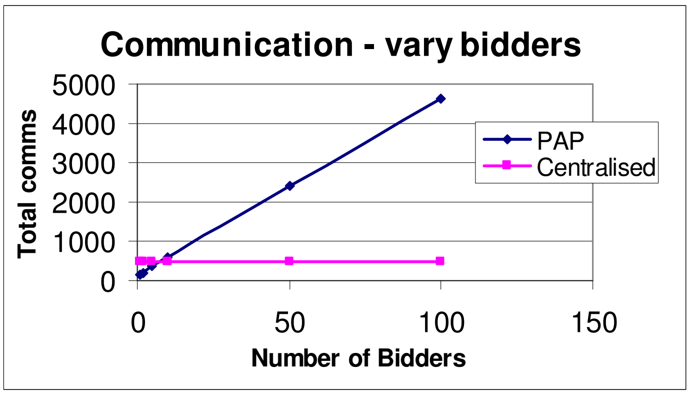


Figure 36. Communication – PAP versus one-shot (centralised) approaches to combinatorial auction – varying the number of bidders (250 bids, 50 goods).

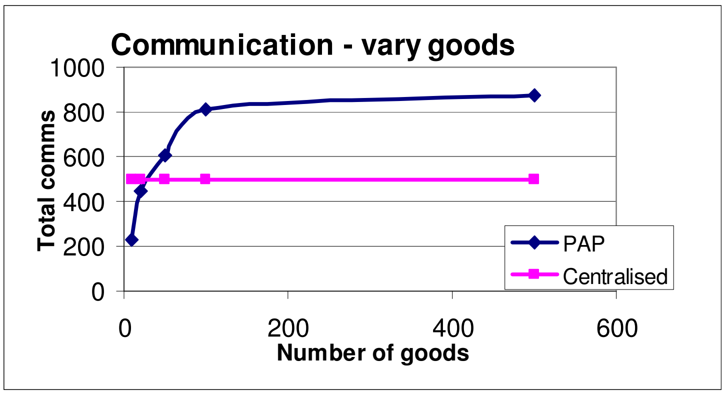


Figure 37. Communication – PAP versus one-shot (centralised) approaches to combinatorial auction – varying the number of goods (250 bids, 10 bidders).

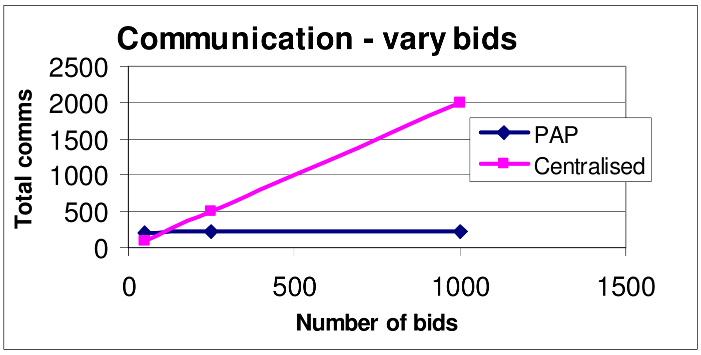


Figure 38. Communication – PAP versus one-shot (centralised) approaches to combinatorial auction – varying the number of bids (10 goods, 10 bidders).

Figure 39, Figure 40 and Figure 41 show results of comparing PAP with ECNP when applied to the combinatorial auction domain with a single auctioneer, static bids and no backtracking, as required by ECNP. We also allowed ECNP to submit the bid evaluation function so that one bid for each bidder is submitted per task. As shown in the previous chapter, PAP requires less communication than ECNP if the number of bids *br* for each task is greater than 2, and the saving in communication is *(br – 2)·m*, where is the depth of the search (number of bids required in the final solution). Therefore, savings in communication increases with the depth of the search or the number of bids submitted.

Figure 39 and Figure 40 illustrates the total communication required for paths and scheduling data, respectively, for varying goods and number of bidders. As the number of goods and bidders increases, so too does the saving in communication by using PAP, where the saving in communication is shown by the gap between each of the two lines with the same number of goods. The reason is that the number of goods is proportional to the depth of the search (*m*) as a greater set of goods to achieve required more bids to achieve it in our experiments. Additionally, the number of bidders is proportional to the number of bids submitted because each bidder will usually submit one bid each (if possible) for the task. Therefore, a greater number of bidders results in more bids being submitted for each task. Note that the 100 goods results in Figure 40 could not be distributed among 50 bidders. Bids with the same dummy goods were distributed to the same bidders, and there were not enough dummy goods in the data to be distributed among the 50 bidders.

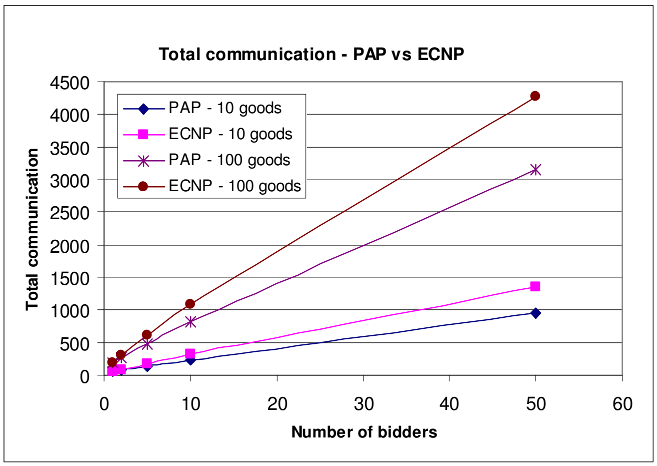


Figure 39. Communication – PAP versus ECNP (paths data, 250 bids).

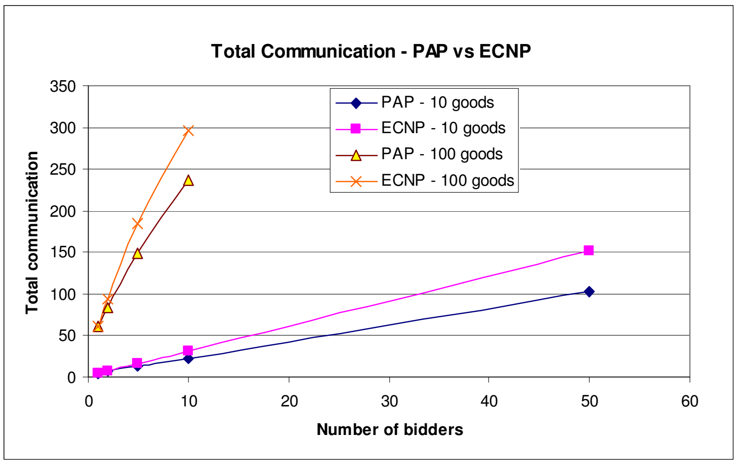


Figure 40. Communication – PAP versus ECNP (scheduling data, 1000 bids)

In Figure 41, we show the total communication for PAP and ECNP as the number of bids and bidders varied, keeping the depth *m* relatively constant. We only used paths data because the scheduling data had varying depths for the different experiments. The saving in communication remained *relatively* constant for the same number of bidders. Increasing the number of bids did not usually result in an increase in the number of bids submitted for each task because each bidder only submits a maximum of one bid per task, regardless of how many bids they have. The saving in communication did increase slightly with a greater number of bids because there was a slight increase in the number of bids submitted. When each bidder had less bids, some ran out of bids to submit when they had other bids allocated by the auctioneer (as bidders cannot submit other bids containing goods that are allocated). Therefore, not all bidders submitted a bid for each announced task when they had less bids.

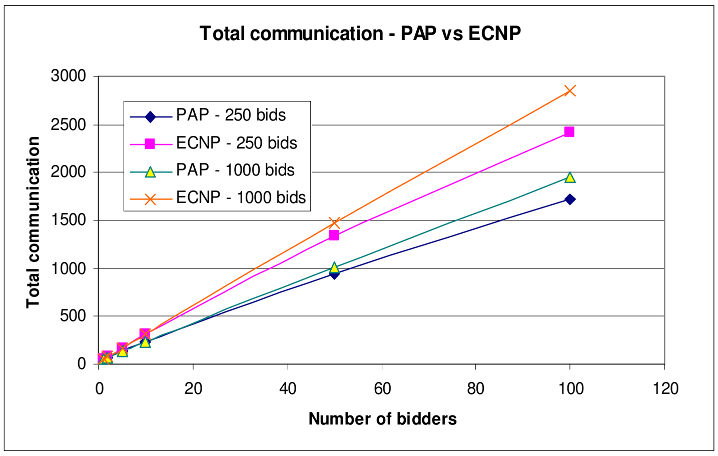


Figure 41. Communication – PAP versus ECNP (paths data, 10 goods).

6.3.4 Performing Decentralised Depth-First Search

We ran 8 data sets using both a centralised depth first search and PAP with backtracking. The data was modified to allow the bids to have unique bid evaluations (which was not the case for two datasets). Therefore, for every announced task, the auctioneer would always prefer one bid over all others, rather than having two bids with the same bid evaluation and selecting one at random. This made it easy to compare the two algorithms. The first 100 tasks (or nodes) and selected bids (or branches) of the search tree produced by both PAP and the centralised depth-first search were used in the comparison. In both cases, the search tree was the same, which is consistent with the theoretical results that PAP will run a decentralised depth first search. Note that the two datasets with bids that did not have unique bid evaluations before the first 100 tasks and bids were checked up to that point.

6.3.5 Backtracking

Backtracking was used on the datasets, using the heuristic that backtrack until a solution is found that is a greater allocation than the initial solution: if < 70%, 70% - 79%, 80% - 89% or 90% - 99% allocation is found initially, then backtrack until a > 70%, > 80%, > 90% or 100% allocation is found, respectively, or some time limit is reached.

In our results, out of 41 scenarios which successfully completed, 37 (90.2%) produced a better solution by backtracking, which on average, was 9.4% better. Of the 4 (9.8%) which produced a worse solution, it was only 2% worse. From our results, backtracking was useful, was likely to produce a better solution, and if not, the solution was not significantly worse.

Rather than focus on the suitability of the heuristic employed, we are examining the case of a heuristic that utilises backtracking and how PAP supports this. In the case of no free disposal, backtracking can be used until a solution is found such that all goods are allocated.

Note that there is a communication and time overhead with backtracking, which depends on the amount of backtracking required to find a suitable solution. We also found that in some cases, there were no solutions which provided a greater allocation. Therefore, in trying to find a better solution, the auctioneer was left with the worst solution, no allocation of goods. In such a situation, it may be beneficial for the auctioneer to revert back to provisionally rejected bids, which are currently discarded. This issue is under investigation.

6.4 Single Auction, Dynamic Bids

In order to simulate dynamic bids, we ran 16 scenarios where up to 50% of the bidders' bids were *delayed,* and hence became available after depth 1, 2, 5 and 10 in the auction using PAP with no backtracking (see Table 2 – values are percentage of the optimal solution with all bids available). We ran centralised auctions without delayed bids, to simulate current one-shot approaches that collect bids once and process them. Greedy centralised runs the same heuristic, and hence search, as PAP with no backtracking.

	Centralised		**PAP – delayed bids at depth:**			
	Optimal	***Greedy***	***1***	***2***	***5***	***10***
Mean	83.2	78.0	83.9	82.9	79.6	79.5
Std Dev	13.0	15.4	12.2	14.6	14.3	15.6

Table 2. PAP versus One-shot (centralised) approaches with delayed bids.

On average, solutions for PAP were better (greater) than centralised greedy. This indicates that PAP's ability to interact with the environment and take advantage of new (potentially better) bids introduced *during* the auction (planning) can improve the final solution. As expected, the later in planning (increasing depth) the bids become available, the smaller the improvement. The average solution for PAP with delayed bids available at depth 1 was better than the centralised optimal, indicating that even a greedy search that takes advantage of a changing environment can produce a better solution than the optimal solution which does not.

It was not necessary to perform experiments with bids being retracted because if the centralised approach found a solution that contained a retracted bid, then the solution would be infeasible. PAP allows bids to be retracted. Bidders are sent a (provisionally) withdrawn message if the auctioneer tries to provisionally grant a retracted bid. Therefore, PAP's ability for auctions to interact with the changing environment, taking advantage of new bids and acknowledging retracted bids, can improve the quality of solutions over current one-shot approaches.

6.5 Multiple Simultaneous Combinatorial Auctions

6.5.1 Without Backtracking

Multiple auction scenarios comprise up to 10 auctioneers with the *same* goal and up to 50 bidders. Note that PAP aims to optimise each auction's *local* solution, and *not* the *global* solution. We examine the behaviour of PAP as bidders and auctioneers change. Figure 42 shows results comparing global price, or the sum of prices obtained from all auctions, versus number of bidders for datasets using paths data, 100 goods and 250 bids. Increasing the number of bidders in this case is equivalent to increasing the number of bids, and hence *resources* available to auctioneers. Even if a bidder has many bids, it can only be allocated at most one of each good, and hence any of its other bids comprising that good are no longer available.

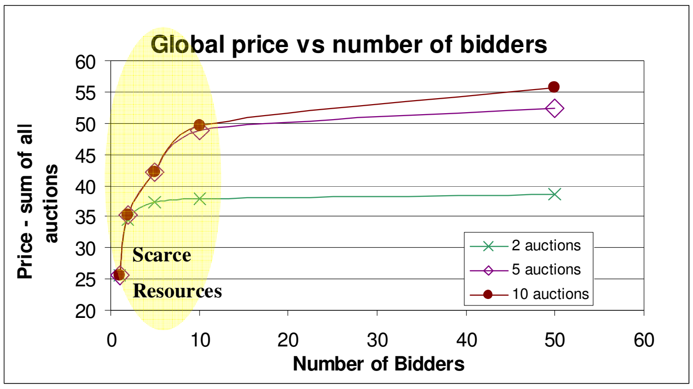


Figure 42. Global price versus number of bidders (paths data, 100 goods).

From Figure 42, as resources increase, auctioneers approach their globally, and thus locally, optimal price, but as resources become scarce, the global price decreases [25]. This occurs for two reasons. Firstly, due to the lack of bids, auctioneers may only get a small, or no, allocation of bids. Secondly, competition increases with decreasing resources as auctioneers must fight for the same bids. Competition increases the chances that an auctioneer's locally optimal allocation of bids *conflicts* with others, reducing its chances of obtaining the optimal. Additionally, a major problem with increasing competition is that it may result in a globally inefficient allocation of resources as one auctioneer may obtain a bid that another requires, and vice versa. This phenomena is known as the tragedy of the commons, whereby each agent in trying to maximise their local utility results collective behaviour that minimises each agent's utility, and thus minimises the global utility (Hardin 1968). This phenomenon in our situation decreases with a large amount of resources because resources required to achieve individual agents' goals do

[25] The global price does not necessarily increase proportionally with the number of auctioneers because once high value bids are allocated, lower value bids are used, leaving many auctioneers with little price improvement.

not conflict. Therefore, PAP is likely to perform better locally and globally when resources are plentiful, reducing the chances of conflicts between required bids.

It is often said that "auctions are used to allocate scarce <u>resources</u>". Note that we underline <u>resources</u> in this context to avoid confusion because its meaning differs to what we mean by resources. <u>Resources</u> in the traditional auction context are the good(s) that the auctioneer is auctioning. Therefore, scarce <u>resources</u> imply that there are many bids for the auctioneer's good(s), and so auctions are used to find the most appropriate bidder to have the <u>resource</u> allocated to. Hence, PAP is likely to perform well when applied to auctions in this context as there are likely to be many bids.

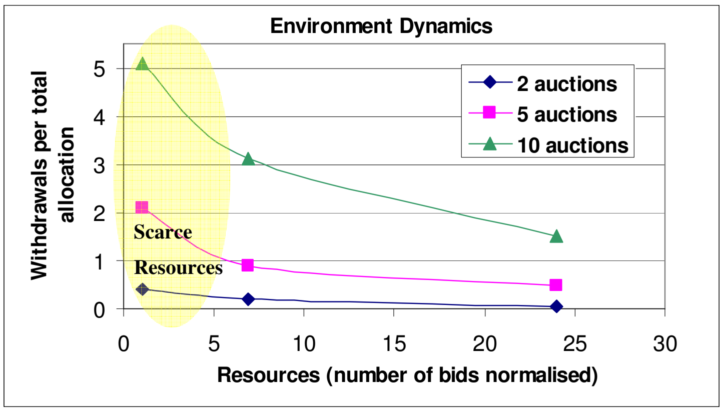


Figure 43. Environment dynamics versus resources (paths data, 100 goods).

In Figure 43, as resources become scarce, the number of (provisionally) withdrawn messages per total allocation increases, indicating that the environment is more dynamic (bids are being retracted). We divide withdrawn messages by the total allocation (sum of bids allocated by all auctioneers) because the number of withdrawn messages should increase as planning time and depth increases. Increasing auctions with the same number of bids increases withdrawals, and thus dynamics, because increasing the number of auctions reduces resources as there are less bids available per auctioneer.

6.5.2 With Backtracking

In section 6.3.5, we showed that backtracking to obtain a greater allocation in a single auction case (with our datasets) was likely to improve the solution. Our aim is to investigate if the same applies to the multiple auction case. We ran experiments, varying the number of auctioneers and bids. Auctioneers backtracked to find an allocation that has one or more goods than the initial solution. The heuristic differs to that in section 6.2 to increase the chances of finding a solution with minimal backtracking.

The results are shown in Figure 44 (paths data, 100 goods), displaying the number of auctioneers who fail to find a solution versus the quantity of resources. As resources become scarce, a greater number of auctioneers did not find an allocation. There are four reasons. Firstly, there may not be a solution with a better allocation, and less resources implies less possible solutions. Due to partial observability (do not have all bids), the auctioneer does not know if a better solution exists. Secondly, there may not be enough bids for all the auctioneers, and therefore it is inevitable that some will not obtain a solution. Thirdly, due to increased dynamism as a result of scarce resources, ungranted bids that could potentially provide a better solution may no longer be available when backtracking. Fourthly, auctioneers hold bids by provisionally granting them, and thus preventing bidders from sending the bid to other auctioneers, or allow other auctioneers to grant the bid, which is later released. Therefore, by the time the bidder sends the bid to other auctioneers for their tasks, the auctioneers may have already given up on achieving the task believing no solution was possible (as no bids were submitted before the bidding deadline). Additionally, if the held bid was already sent, and an auctioneer tried to provisionally grant it, the bidder will provisionally withdraw the bid. This causes the auctioneers to discard that option, which they could have used soon after. This is also detrimental to the bidder as it prevents the bidder from having its bid accepted. Allowing auctioneers to reconsider provisionally withdrawn bids, and the vulnerability of bidders having resources held, is the subject of future work.

Figure 45 shows the result of scenarios where only half of the auctioneers backtracked (paths data, 100 goods, 10 auctioneers). Auctioneers that did not backtrack when resources were scarce were better off than those that did backtrack, since they all found a solution. Once (or if) they obtained (provisionally granted) scarce bids, they held on to them. Auctioneers that backtracked released the resources and were usually unable to find a better, or any, replacement (for reasons explained above). Thus, backtracking with scarce resources can be detrimental, and a greedy approach is more suited. A priori, this seems counter intuitive as one would generally expect backtracking to provide a better solution.

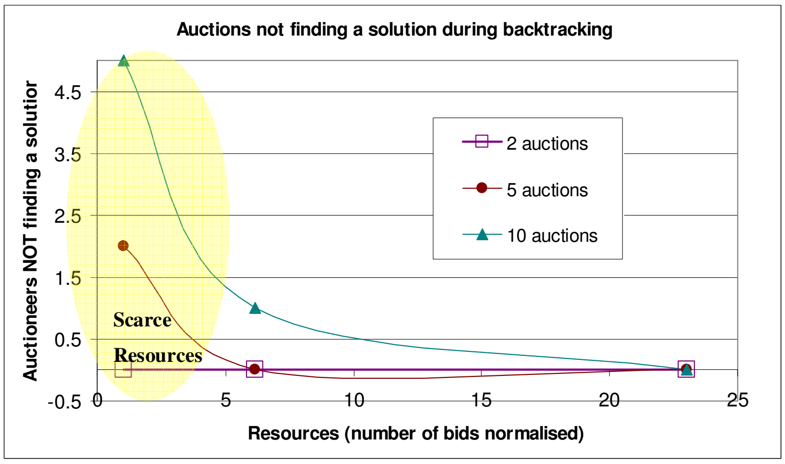


Figure 44. Multiple auctions with backtracking (paths data, 100 goods)

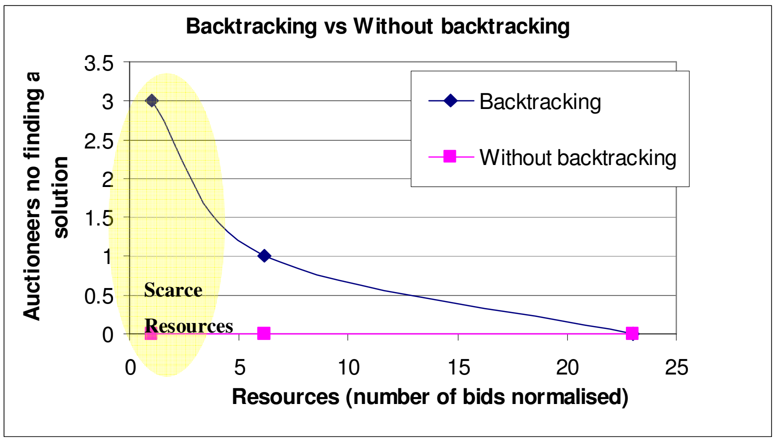


Figure 45. Backtracking versus without backtracking (paths data, 100 goods, 10 auctioneers)

6.6 Summary

In this chapter we applied PAP to the combinatorial auction problem in order to empirically evaluate PAP, demonstrate the benefits of PAP over current one-shot combinatorial auction approaches and show that PAP can facilitate the novel multiple simultaneous combinatorial auction.

One-shot combinatorial auction approaches, such as (Andersson, Tenhunen et al. 2000; Hunsberger and Grosz 2000; Nisan 2000; Walsh, Wellman et al. 2000; Collins, Bilot et al. 2001; Collins, Ketter et al. 2002; Sandholm 2002; Sandholm, Suri et al. 2005) require all bids and their dependencies to be submitted to the auctioneer for processing. This may not always be possible as there may be many bids with complex dependencies. The PAP approach is decentralised so bidders process their bids themselves and submit their single best bid. Therefore, each bidder does not need to submit all bids and their dependencies. PAP has communication benefits over one-shot combinatorial auctions if bidders possess many bids. This is the case in many real world applications, such as our transportation domain presented in the next chapter. Additionally, we have shown that PAP may produce a better solution in a dynamic environment than one-shot auctions approaches that find an optimal solution due to PAP's ability for auctions (or the auctioneer) to interact with the changing environment during the auction process. Dang and Jennings (Dang and Jennings 2002) uses a greedy approach to combinatorial auctions, like PAP, but is also centralised and does not consider backtracking. Similar to Dang and Jennings approach, PAP, when applied in a greedy manner (no backtracking), scales well. We showed that PAP's backtracking facility can be used in a static environment, with a suitable heuristic, to obtain a better solution than the first (greedy) solution found.

Experiments were performed that showed that the saving in communication for PAP compared with ECNP increased with the number of bidders or the depth of the search (number of bids to achieve the auctioneer's task), but remains relatively constant as the number of bids increase. Empirical results are consistent with the fact that PAP performs a decentralised depth-first search.

PAP was able to facilitate the multiple simultaneous combinatorial auctions. We found that as resources (bids per auctioneer) became scarce, dynamism and competition (and thus the tragedy of the commons phenomenon) increased. Therefore PAP is likely to perform better locally and globally when resources are plentiful, which are when typical auctions are useful anyway. Although PAP's backtracking facility allowed a better solution in a static environment, it can be detrimental with multiple auctions when resources are scarce. This is primarily due to the increase in dynamism and ability for auctioneers to hold on to resources that they later release, preventing others from using them. Auctioneers in our scenarios are better off holding on to bids they have obtained rather than releasing them and attempting to obtain other bids that appear to be better. The potentially better bid that the auctioneer is trying to replace its current bid with may no longer be available or may be held by other auctioneers, which PAP currently discards this potential future option. Backtracking being detrimental seems counter intuitive as a better solution is always expected with backtracking. In our domain, due to partial observability and dynamism, this is not always the case. Due to partial observability (auctioneer does not have all bids – *as with single auctions with static and dynamic bids, and multiple auctions*), it may not be possible to determine whether a better solution is available, and how to get there. This makes it difficult to know when to backtrack and which bids to select. Even if the auctioneer is aware of a better solution, it may no longer be available by the time the auctioneer backtracks and attempts to secure it. The solution that was given up as a result of backtracking may also become unavailable when an auctioneer tries to regain it.

The issue of auctioneers holding on to, and later releasing, bids is also detrimental to bidders. Bidders that are committed to a provisionally granted bid are (currently) unable to accept provisional grants for any other conflicting bids they have submitted. Therefore, if the provisionally granted bid is later withdrawn, it prevents potential contracts with the previously granted conflicting bid. Sen and Durfee (Sen and Durfee 1994; Sen and Durfee 1998) investigate the issue of commitment strategies – whether to commit to a bid and block other conflicting bids, or to not commit to a bid until a full agreement is reached. The latter prevents the problem of blocked resources that become available again blocking potential bids from being made. In our domain, strict contractual obligations force commitment when a provisional grant is accepted for a bid. Any decommitment requires a penalty to be paid. Allowing auctioneers to reconsider provisionally withdrawn bids in case they become available again, and the vulnerability of bidders having resources held, is the subject of future work.

From our knowledge of the current literature, the multiple simultaneous combinatorial auction problem that we describe have not been addressed. Other combinatorial approaches, such as iBundle, use ascending auctions (Parkes and Ungar 2000; Wurman and Wellman 2000; Walsh and Wellman 2003; Ausubel and Cramton 2004). These are suited to domains where bids for auctioned items are dynamically priced. In our domain, bidders have fixed (true) valuations (or prices) for their bids – as is the case in many reverse auctions (e.g. our transportation domain) – and therefore is a problem of allocation rather than *price determination*. The auctioneer or bidders may not want others to see bid prices, as it may be private information. Additionally, they do not consider the multiple auction case, where *each auctioneer* allocates *multiple goods*. Double auctions (Wurman, Walsh et al. 1998; Park, Durfee et al. 2000; Tesauro and Das 2001; He, Leung et al. 2003; Babaioff and Nisan 2004; Vytelingum, Dash et al. 2004), which have a many-to-many setting, require both auctioneers and bidders to submit goals and bids to a mediator that matches them (e.g. stock market). This may not always be practicable as a bidder may have many bids, they may not know what they want until a goal is presented, or particularly in reverse auctions, bids (services offered) are tailored to suit the goal at hand. This is the case in the transportation scheduling application discussed in the next section. Additionally, with a mediator, agents may not have control over who receives their sensitive information. Therefore, we focus on single sided auctions in a many-to-many setting. (Priest 2000; Byde, Priest et al. 2002; Shehory 2002; Airiau and Sen 2003; Anthony and Jennings 2003; He, Leung et al. 2003; Cheng, Leung et al. 2005; Greenwald and Boyan 2005; Reeves, Wellman et al. 2005) investigates the problem of which auctions bidders should bid in, and at what price, in order to obtain a good at the best price. Again, they assume dynamically priced bids. In PAP, bidders are allowed to bid in all auctions at a fixed price until the bid is allocated.

Mechanism design aims at finding protocols for agents to optimise the global welfare (Mas-Colell, Whinston et al. 1995; Varian 1995; Nisan 1999; Parsons and Wooldridge 2002; Cramton, Shoham et al. 2006). PAP does not aim at maximising the global price (sum of individual agent's prices) for the multiple auction case. Our aim is to provide a protocol to facilitate interaction that is present in many real world situations – self-interested agents (organisations) finding themselves a suitable plan and allocating tasks, using contracting, in a complex and dynamic environment, in the presence of other agents that it must compete with for bids. As in many real world situations, our protocol is also susceptible to the tragedy of the commons (Hardin 1968), where individuals trying to greedily maximise their own local utility may worsen their local and the global solution. It may be socially desirable if organisations do act towards increasing the global welfare – which organisations will most likely be reluctant to do so. Research in COllective INtelligence (COIN) attempts to address the tragedy of the commons by setting agents' local goals/behaviours such that maximising their own utility results in agents maximising the global utility (Tumer and Wolpert 2000; Wolpert, Wheeler et al. 2000). In our domain, we cannot directly control agents' goals in our domain. Incentives can be placed so that agents redirect their goals in order to exhibit more global behaviour. Current incentives include social law (e.g. anti-trust laws), where breaking them results in penalties. It may not be practical to devise incentives that require the creation of new laws. In mechanism design, for example, payments may be made to agents in order to provide incentives for them to act towards a social goal. Devising protocols such that agents can exhibit global behaviour, in our particular multiple simultaneous combinatorial auction scenario, is the subject of future work. In devising protocols, one must consider that organisations may be reluctant to use a protocol that attempts to maximise a utility other than their own and one which is not consistent with organisational interactions (e.g. must involve contracting and be decentralised).

There may be situations in which multiple auctions using PAP obtains an inefficient allocation, due to tragedy of the commons, and there exists an allocation which is both globally and individually superior – i.e. the solution is not Pareto optimal. This brings to a light a dilemma in the multiple auction case. On the one hand, using a protocol which acts selfishly (such as PAP) ensures that decisions made are in the agent's self-interest, but acting selfishly may result in a worse allocation than if they acted cooperatively towards a global utility. On the other hand, using a protocol that acts toward a global utility may cause the agents to not act in their own interests and obtain a worse solution than if they were selfish.

This dilemma is analogous to our own variation of the famous prisoner's dilemma game from game theory, shown in Table 3 – the example is taken from (Sandholm and Lesser

2002) [26]. In the original prisoner's dilemma problem, there are two players A and B in separate rooms. Each one can press one of two buttons: (1) cooperate or (2) defect. Based on which buttons the agents press, they receive payoffs according to Table 3. In relation to our multi-auction problem, there are two auctioneers (or auctions) A and B. Defect implies that the player greedily tries to acquire resources in order to achieve its own local goal. Cooperate implies that the player attempts to cooperate and share resources in order to maximise their shared goal. The situations in Table 3 where one player defects (and receives a payoff of 5) and the other cooperates (and receives a payoff of 0) are interpreted differently in our domain. There are two cases: (i) the same as in the traditional prisoner's dilemma game, where one player decides to cooperate and the other defects, somehow resulting in a greater payoff for the defector [27] and/or (ii) the defecting player greedily achieves its own goal and successfully obtains a greater number of resources than the other player, regardless of whether the other player decides to cooperate or defect [28].

		Player A	
		Cooperate	*Defect*
Player B	*Cooperate*	3, 3	0, 5
	Defect	5, 0	1, 1

Table 3. Prisoner's Dilemma Game – in each square, player B's payoff is listed first.

[26] Relating our tradegy of the commons dilemma observed in multiple combinatorial auctions to the well known Prisoner's dilemma may provide insight to addressing/understanding our problem.

[27] How this could occur in our domain (in the real world), and how likely is it occur, is an open question.

[28] This is a realistic scenario in our domain where deciding to act selfishly *may* produce a better result. This case differs from the traditional prisoner's dilemma game as the output for the defecting player is not dependent on the choice of the other player, but on other factors (e.g. chance – the *cooperative player* here should be labeled the *unlucky player*). As in the traditional prisoner's dilemma game, the defector player does not have full control of the output – it's dependent on other factors (e.g. the other player and chance).

In the traditional prisoner's dilemma game, the dominant strategy for each player is to defect, having a global utility of 1 + 1 = 2 (Sandholm and Lesser 2002), even though if they cooperated they could increase both the global and their local utility. Is the dominant strategy in our variation of the game to also defect? If so, then is PAP (and many real world organisational interactions) performing the dominant strategy – act selfishly and hope that the maximum payoff is obtained? What mechanisms, information and situations are required so that agents act cooperatively? In the prisoner's dilemma problem, each player knows the existence of the other player and the payoffs as a result of both their decisions. In a real world situation, can agents identify this information? How do agents identify other agents whose actions (acquiring common resources) will conflict with its own actions, particularly in a decentralised domain which is typically partially observable? If agents do identify other agents that they may conflict with, how can they determine the payoffs of cooperating and defecting in a partially observable environment? Therefore, is cooperation possible among auctioneers?

In future work, we intend to investigate protocols (or PAP extensions) which can address issues of producing a Pareto optimal solution, that is, when to act selfishly or cooperative to obtain individually better solutions, or how to recover from a non-Pareto optimal solution. In our domain, a suitable protocol that allows a pareto optimal solution is necessary. A suitable protocol that enables a globally optimal solution is the Holy Grail.

Chapter 7

7 Global Transportation Scheduling

Having presented PAP in previous chapters, we now apply PAP to our global transportation scheduling domain (Perugini, Lambert et al. 2003; Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004), as required for our Multi-Agent Logistics Tool (MALT) (Perugini, Lambert et al. 2002; Perugini, Wark et al. 2003). The global transportation scheduling framework we present extends Fischer's transportation scheduling framework, which uses ECNP. Fischer's framework is not intended to allow partial route, in addition to partial quantity bids, and inherits the limitations of the protocol (ECNP) used to facilitate the scheduling. The flexibility our transportation scheduling framework allows us to address a wide range of transportation problems, such as Fischer's transportation problem (Fischer and Kuhn 1993; Fischer, Muller et al. 1996), multi-commodity, multi-modal network flow problem with time windows (Haghani and Oh 1996), the dial-a-ride problem and Pickup and Delivery Problem with Time Windows (PDPTW) (Savelsbergh and Sol 1995), and the greyhound scheduling problem (Dean and Greenwald 1992; Dean and Greenwald 1992). To follow, we present details of the global transportation scheduling framework, its implementation and experimental results in addressing real world scenarios taken from military logistics exercises. Our implementation was able to automatically form transportation schedules for scenarios which are currently performed manually through a complex, tedious and time consuming process. Therefore, logistics planners could benefit from the automation and potential speed of our implementation. We demonstrate the flexibility of our implementation by applying it to the multi-commodity, multi-modal network flow problem (disaster relief) and the PDPTW. Finally, we discuss some issues regarding our transportation implementation, and thus the general type of planning and task allocation problems that PAP is aimed at addressing.

7.1 Global Transportation Scheduling Specification

7.1.1 Overview

In our global transportation problem, there are a set of Manager Agents (MA, *or auctioneers*) that must transport a large quantity of resources over a large distance (potentially on a global scale). They transport the resources using the services of Transport Agents (TA, *or bidders*). Due the to large quantity and distance required in the transportation, each TA may only be able to transport a partial quantity of the resources only a part of the distance (or *route*). Each MA must obtain and assemble these services from various TA in order to achieve its transportation goal. The MA and TA are operating in a complex environment that is characteristic of an open market, and thus is decentralised, dynamic and open, and all are interacting simultaneously in a many-to-many setting (many MA interacting with many TA at the same time). Figure 46 illustrates the agent interaction and Figure 47 provides an illustrative example of transportation.

Although this transportation problem evolved from the (Australian) military, it may not be specific to the military. Globalisation and deregulation are increasing in the commercial sector. Therefore, the commercial sector may also have an increasing requirement for the global transportation scheduling problem described above.

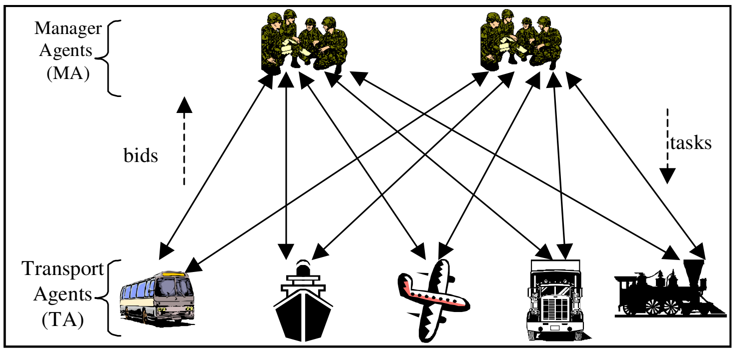


Figure 46. Agent interaction in global transportation scheduling.

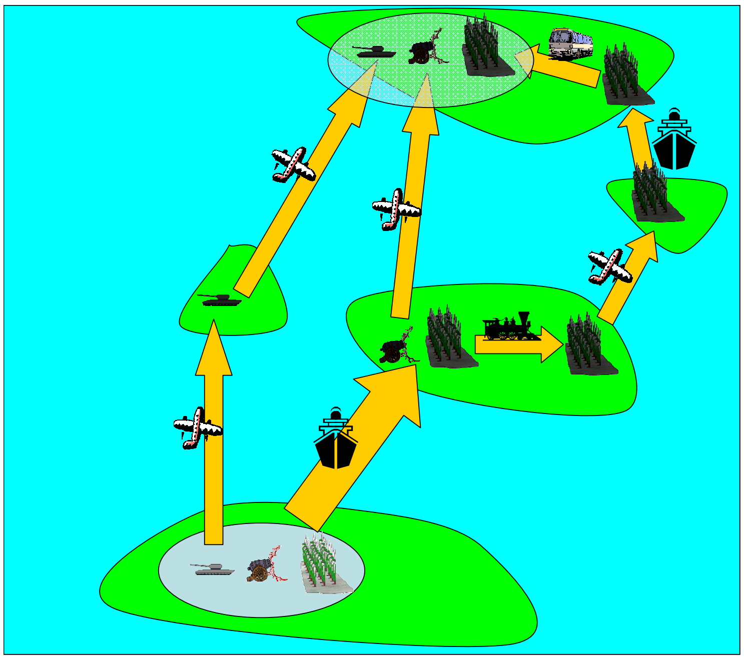


Figure 47. Example of transportation in global transportation scheduling, showing the path and mode of transport of the three military resources at the bottom of the figure to get to their destination at the top of the figure.

7.1.2 Problem Definition

There are a set of MA at time τ, denoted Λ_τ, and each MA $\alpha \in \Lambda_\tau$ has set of (root) transportation tasks T^α_τ that must be achieved. Each task $t \in T^\alpha_\tau$ is defined as $t = <n, q, est, lft, src, dst>$, where quantity q of package n is to be transported from source location *src* to destination *dst*, with earliest start time *est* and latest delivery time *lft*. There are a set of TA (or transportation assets) at time τ, denoted Γ_τ which can be used to achieve T^α_τ. Each TA $\gamma \in \Gamma_\tau$ is defined as $\gamma = <cap, R, lp>$ has capacity *cap*, set of routes R that it can service, and a local plan that contains a time ordered sequence of actions, *lp*. Each action $a \in lp$ is defined as $a = <tsa, tfa, lsa, lfa, inv>$, where *tsa* and *tfa* are the action start and finish times (dependent on the γ'*s* speed, loading time, etc.), respectively, *lsa* and *lfa* are the action start and finish locations (end points of the route $r \in R$), respectively, and *inv* is the set of inventory items. Each inventory item $i \in inv$ is defined as $i = <m, c, p>$, where the inventory item carries quantity c of package m at price p. We define *capacity*(a) as the sum of all the quantities c in all the inventory items i within the action inventory $inv \in a$. The capacity transported in each action must be smaller than the capacity that γ can carry, i.e. $capacity(a) \leq cap$. There must be enough time for the TA to get from the end location *lfa* of one action to start location *lsa* of the next action. An action may transport partial quantity and route of a task *t*, therefore, if $m = n$ for $m \in i$ and $n \in t$, then $0 < c \leq q$, and *lsa* and *lfa* may or may not be equal to *src* and *dst*, respectively. Packages may be picked up and delivered at any location, and all locations are assumed to have infinite storage capacity.

A distributed transportation plan $plan(T^{\alpha}_{\tau}) = \{<\gamma, a, inv> | \gamma \in \Gamma \ \& \ a \in lp \ \& \ inv \in a\}$ to achieve T^{α}_{τ} consists of the set of actions and inventories by TA γ that are associated with T^{α}_{τ}. If *tdel* is the delivery time of package *m* in $plan(T^{\alpha}_{\tau})$, then $tdel \leq lft$. The price to deliver package *m* is *pr*, which is the sum of all inventory prices *p* in $plan(T^{\alpha}_{\tau})$ associated with *m*. The plan $plan(T^{\alpha}_{\tau})$ has a cost value *cv* associated with it, which is a function of *tdel* and *pr*. We assume that the earlier the delivery time *tdel*, and the lower the price *pr*, for all tasks in T^{α}_{τ}, then the better the plan $plan(T^{\alpha}_{\tau})$, and hence the lower the *cv*. The aim of transportation planning is to find a $plan(T^{\alpha}_{\tau})$ such that *cv* is minimised.

7.1.3 Complexity & Depth-First Search Approach

Dean and Greenwald (Dean and Greenwald 1992; Dean and Greenwald 1992) have investigated a similar problem, called the grey-hound package scheduling problem. It allows TA to perform part of the route of transporting a package. They make the restriction that the TA schedules are fixed, simplifying the problem (Dean and Greenwald 1992; Dean and Greenwald 1992). They show that the optimisation problem is still NP-complete (Dean and Greenwald 1992). In our transportation domain, this restriction is not made, allowing TA to perform any transportation action at any time. Additionally, the global transportation problem is equivalent to the (multiple vehicle) Pickup and Delivery Problem (PDP) with time constraints, without the restrictions that transport assets must depart and return to a central depot and no drop-and-swap between transport assets. The PDP is known to be NP-hard (Savelsbergh and Sol 1995), without considering the additional complexity of drop-and-swap, i.e. cannot just assign a transportation task to a single transportation asset. Therefore, an optimal solution to most global transportation planning problems (scenarios) will be intractable.

In order to deal with the computational complexity, we use a depth-first search to address this problem – a greedy approach with backtracking if an infeasible solution encountered. Hence, we use PAP to facilitate the required depth-first search planning and task allocation, allowing a solution to be found in a reasonable amount of time. Additionally, PAP is also able to deal with other domain complexities, such as planning in a decentralised, dynamic and open environment, in a many-to-many setting.

7.2 Applying PAP to Global Transportation Scheduling

We provide a simple example of some of the operational details of our implementation using PAP, and illustrate the complex search tree that is created during planning. Refer to PAP specification in section 5.1, and the problem definition in section 7.1.2.

Figure 48 illustrates the search tree, and thus the planning process, from a single MA's perspective. In Figure 48 (a), the MA **announces** two root transportation tasks t_1 and t_2 (announced separately, as they are achieved independently), which are considered the root node, i.e. $T_{root} = \{t_1, t_2\}$. $t = <n, q, est, lft, src, dst, dl, f>$[29] where *dl* is the bidding deadline and *f* is the bid evaluation function used by the TA to determine the cost value *cv* that the MA will use to evaluate the bids. Each TA sends their **bid** with the smallest *cv*, and which each believes can fully or partially achieve the quantity and route of the transportation task. We discuss *f* and *cv* in detail in section 7.3.5. Say t_2 = <fuel, 110 Kg, 20 hrs, 50 hrs, Adelaide, Cairns, now+5, *f* >. TA submit bids b_1, b_2, b_3, b_4, which are the branches in the search tree from the root node. Each b_i is of the form $b = <n, c, tsa, tfa, lsa, lfa, p>$[30]. Say b_4 = <fuel, 50 Kg, 30 hrs, 38 hrs, Melbourne, Sydney, $1000>.

[29] *dl* and *f* do not appear in the problem description in section 7.1.2 because they are specific to our solution to address the problem (the use of PAP for transportation scheduling) rather than a component of the problem itself.

[30] Bids are conveniently respresented as a conglomeration of actions and inventories presented in section 7.1.2 because a bid corresponds to a single inventory item $i \in inv$ in an action *a* in the TA's local plan. Again, the representation of bids here is specific to our solution to address the problem rather than the problem itself (e.g. bids are not specified in the problem definition in section 7.1.2).

MA decides to **provisionally grant** b_1, which has the lowest *cv*, but the bid was **provisionally withdrawn** (?×). An updated **bid** for t_1 is not sent. MA then tries to **provisionally grant** b_3, but the bid is **withdrawn** (×).

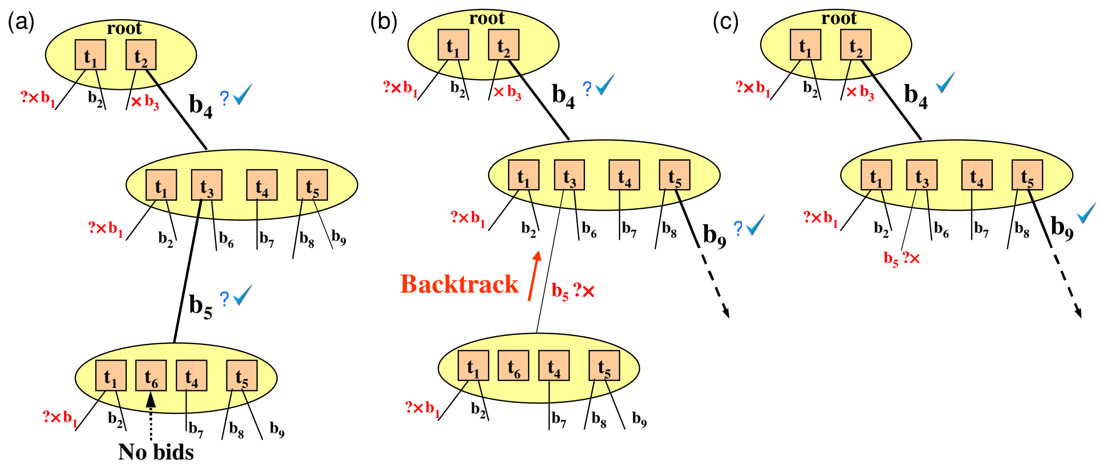


Figure 48. Transportation scheduling planning process.

b_4 is **provisionally granted** (?✓) and the **provisional grant is accepted**. The new task (node) T_{new} to achieve is the difference between the root node and b_4, which is essentially $T_{new} = \{t_1\} \cup diff(t_2, b_4)$. Up to three new tasks can be created (see Figure 49), to transport the package the remainder of the route (if b_4 is a partial route bid): at the beginning <*n, c, est, tsa, src, lsa, dlnew,* f_1>; end <*n, c, tfa, lft, lfa, dst, dlnew,* f_2>; and to transport the left over quantity the complete route (if b_4 is a partial quantity bid) <*n, (q-c), est, lft, src, dst, dlnew,* f_3>. In our example, the left over tasks are (beginning route) t_3 = <fuel, 50 Kg, 20 hrs, 30 hrs, Adelaide, Melbourne, now+5, f_1>, (end route) t_4 = <fuel, 50 Kg, 38 hrs, 50 hrs, Sydney, Cairns, now+5, f_2>, and (left over quantity) t_5 = <fuel, 60 Kg, 20 hrs, 50 hrs, Adelaide, Cairns, now+5, f_3>. The new tasks, in addition to all tasks that are not associated with the granted bid (b_4), become the new node. Only new tasks (t_3, t_4 and t_5) are **announced**, since bids for tasks in the previous node are carried over. This is only allowed if the cost value has certain properties, which will be discussed in section 7.3.5. The MA in this example then receives **bids** b_5, b_6, b_7, b_8, b_9.

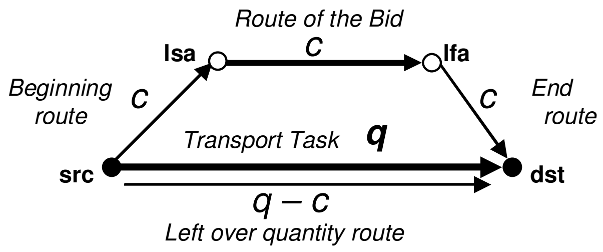


Figure 49. Spacial representation of the transportation task, bid and remaining transportation tasks (beginning, end and left over quantity route), and their corresponding quantities that must be transported (c, q or q – c).

b_5 is then **provisionally granted**. The newly created task is t_6, which is **announced**. The other tasks and their bids are carried over to the node, and therefore the new node contains tasks t_1, t_4, t_5 and t_6. After the bidding deadline *d*, no bids are received. The MA assumes there is no solution to achieve t_6, and thus has encountered an infeasible solution. In our implementation, **backtracking** (only) occurs when an infeasible solution is encountered, and thus b_5 is **provisionally rejected**, and b_9 is **provisionally granted**, as shown in Figure 48 (b). The PAP process continues until all the tasks in the current node are achieved. The transportation plan, which consists of the provisionally granted bids (b_4, b_9, …) are given a **confirm grant** (✓) to secure the plan (individual bids) into the TAs' local plans, as illustrated in Figure 48 (c). If a feasible solution does not exist, then the protocol will exit in failure after backtracking at the root node.

The behaviour of the MA is reasonably straight forward, as described above. The behaviour of the TA is more complicated, which is discussed in the next section.

7.3 Transport Agent Bidding

Allowing partial bids and the provisional granting of bids can complicate TA bidding. Partial bids are problematic as *any* bid, that meets the task constraints, can potentially be part of the solution (plan) to achieve a transportation task. For example, for a task to move resources between Melbourne and Sydney, a partial bid to transport the resources from New York to London could potentially be part of a feasible solution. A rational individual is unlikely to consider such an option, unless it was the only or best option.

Therefore, the TA have the dilemma of deciding which bids (route, time and capacity) is likely to be a suitable bid to achieve the task. It takes computational time to form and check each bid for its suitability (e.g. calculating the bid price and cost value), and there is likely to be an extremely large number of possible bids, e.g. for each of the many possible routes, the TA can transport various quantities, at many different times. Hence, the TA must constrain the search for bids so that it limits the vast possibilities, allowing them to form a suitable bid within a reasonable timeframe.

Provisional granting in PAP complicates bid pricing. If a bid is provisionally granted, the TA enters the bid (the action) into its local plan, as it must commit to the bid. The price of new bids is dependent on actions currently in TA's local plan. The TA may need to travel (or deviate) from locations of transport actions currently in its initial plan to the (pick up and delivery) locations of the new transport bid (or action). These deviations must be included in the price of the new bid. Therefore, where the TA is expected to be located before and after the new bid influences the price of the bid. If actions in the TA's local plan are provisionally granted, and can be rejected, then the TA does not know with certainty exactly where it will be before and after the new bid, making pricing difficult.

To follow, we discuss how we address these issues. Additionally, we discuss the TA updating and withdrawal of bids, and the bid evaluation function f (i.e. cost value cv).

7.3.1 Partial Bids

Figure 50 illustrates the TA's local plan, containing transport actions (granted bids) along a timeline.

Definition 8: Similar to section 7.1.2, *in our implementation* each action is represented by $a_m = <tsa_m, tfa_m, lsa_m, lfa_m, inv_m>$, where tsa_m and tfa_m are the action start and finish times, respectively, lsa_m and lfa_m are the action start and finish locations, respectively, and inv_m is the inventory, $inv_m = \{i_1, i_2, ..., i_w\}$, where $i_k = <n_k, c_k, st_k, pi_k>$. There is one i_k for each bid that attempts to achieve a transportation task, and thus an action achieves w transportation tasks in the single transport, where n_k provides details about the transportation task (the type of packages to be transported, the MA that requires the transport, and details regarding the price of the transport), c_k is the quantity that is being transported, st_k specifies whether the inventory item is provisionally granted or confirm granted, and pi_k is the price only of the transport (not including any deviation price to get to and from adjacent actions to this action – see below). If the maximum capacity that the TA can carry is *cap*, and $capacity(a_n) = c_1 + c_2 + \ldots + c_w$, then $capacity(a) \leq cap$. The finish location of one action (lfa_m) may not be the same as the start location of the next action (lsa_{m+1}), and thus there must be enough time between actions (gap) for the TA to travel between the locations of the two actions.

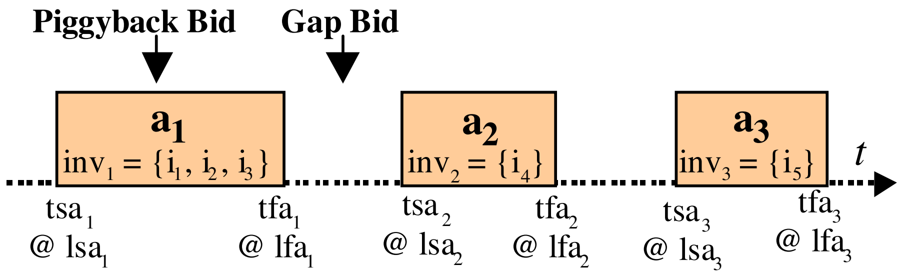


Figure 50. TA's local plan – boxes are transport actions/bids. Bids can be made "piggybacking" with other actions that have free capacity, or in the gaps.

Bids can be made either in with conjunction with existing actions, if there is free capacity available in that transport journey, which we called a *piggyback bid*, or by creating a new action between gaps of existing actions, which we call a *gap bid*. Piggyback bids are easier to check for as the route (lsa_m to lfa_m) and time interval (tsa_m to tfa_m) are fixed. The capacity of the bid must be determined, which is dependent on the left over capacity of the piggybacked action, and the task time constraints and the suitability of the route for the task must be checked. Gap bids are computationally harder to check for since the TA can consider potentially many suitable routes, range of times within the gap time interval, and a range of capacities, all for which the price and a cost value needs to be calculated.

To simplify the bidding process in our implementation, the TA always bids for the largest capacity that it can carry. Thus, the TA are not travelling under full capacity and wasting resources, and achieves as much of the transportation task (in terms of quantity to transport) as possible. Additionally, the TA schedule gap bids as early as possible because the cost value (quality of the bid) is dependent on time, and therefore the earlier the delivery, the better the resulting transportation plan. The time that the bid can be performed not only depends on the time constraints of the transportation task and gap, but if the bid is to perform part of the task's route, then the TA must ensure that there is sufficient time before and after the bid for other TA to perform the rest of the route within the task time constraints. If too much time is left before the bid, then the package may be delivered later than is possible, resulting in a worse plan. If insufficient time is provided before and after the bid, a solution may not be found and the bid will likely be rejected (if granted). Knowledge about the type of TA that can service routes could be used assist in determining the amount of time required before and after the bid.

The final issue with gap bids are selecting the routes that the TA should consider for bids, assuming the TA has a finite set of routes that it can service. Consider a TA that can transport a package from Melbourne to Sydney and Melbourne to London. Say a transportation task was received by the TA to transport a package from Melbourne to Brisbane. A *rational* choice for the TA would be to select the route Melbourne to Sydney when checking for bids for the task since it moves the MA closer to its goal, i.e. reduces the remaining distance that the package must travel, after the bid is performed, to achieve the task. The route Melbourne to London actually moves the MA further away from its destination goal, i.e. the remaining distance that must be travelled to achieve the task, after the bid is performed, is significantly greater than the original distance required by the task. The Melbourne to London route is less likely to produce a better plan than the Melbourne to Sydney route. Therefore, a rational approach is for TA to select the routes that move the MA as close to its goal as the TA possibly can, i.e. select routes that minimise the remaining distance that must be travelled to achieve the task after the bid (route) is performed. The TA have a route threshold *rth* where if the route does not achieve a certain portion of the task (or moves away from the task by a certain amount), then the route will not be considered for a bid, saving computational time.

The route that achieves the greatest amount of MA's task is not necessarily the best route to check for bids in a particular gap. The route may require a large deviation, which is the movement for the TA to get to and from the route (in the bid), from locations in the previous and following actions, respectively. Selecting routes that minimise the deviation in the gap will likely save time and price, and thus resulting in a better bid. Therefore, extra priority is given to routes that require less deviation in the specific gap.

Definition 9: The quantity *rq* that a route achieves of a transportation task is given by

$$rq = \frac{\text{remaining distance to travel if route selected}}{\text{task distance}}$$

and $rq < 1$ if the route moves the MA closer to its task.

The TA's algorithm for bidding for an announced transportation task is:

1. Sort routes in list *ls*, in order of quantity *rq* that the route achieves of the task, removing routes that are above the route threshold *rth*.
2. Check all actions in TA's local plan, that are within the task's time window, for piggyback bids. Only check actions with routes that are in *ls*, and those that have spare capacity. Form any bids and store them with their associated cost value.
3. For each gap in the plan, that is within the task's time window:

   3.1. Sort routes from list *ls* into a new list *lg*, in order of the quantity of deviation that is required by the route in the gap. Routes will a smaller deviation are at the top of the list.

   3.2. Combine the two lists (see below) to form sorted list *lc*, where routes at the top of the list achieve a large portion of the task and requires minimal deviation in the gap.

   3.3. Determine bids for the top *N* routes in *lc*, and store any bids with their associated cost values.
4. Sort the bids in a list *lb* based on their cost values, where bids with lower cost values are on top of the list.
5. If the bid at the top of the list *lb* has not been previously sent, then send it to the MA, and go to step 6. Otherwise, remove the bid from *lb* and go to step 5. If the list is empty, exit, as no suitable bids can be made for the task – keep the task stored as an opportunity may arise later to submit a bid.
6. If a **provisional reject** message is received from the MA, remove the bid from *lb*, and go to 5. If a **withdrawn** message is received from the MA, then delete the task and all its dependencies, and then exit. Otherwise, the bid submission was successful, so exit.

In step 3.2 of the algorithm, for our particular implementation the TA combine the two lists *ls* and *lg* by giving the route at the top and bottom of the lists (*ls* and *lg*) a *scaling value* of 1 and 0, respectively. Other routes are given scaling values between 0 and 1 depending their relative separation from the values (*rq* and deviation distance) of the top and bottom routes. For each route in lists *ls* and *lg*, the TA takes the sum of the two scaling values, and adds the route to the new list *lc*, where the greater the scaling value, the better the route.

Making *rth* too high, or *N* too low, may result in too few routes being checked for bids, and thus a possible solution that might be available may be overlooked. Making these values too low will result in the TA spending too much time computing bids with little benefit. In our prototype, these values were set manually for the specific experiments. Ideally, both *rth* and *N* should automatically adjust, depending on the available time for bidding (bidding deadline), and the number of other received tasks that the TA may bid for.

7.3.2 Bid Pricing

The price *p* of a new *gap bid*, which becomes an action in the TA's local plan with one inventory item, is calculated using

Eq 35: $p = pi + p^t + p^f - p^e$

where (refer to Figure 51 and Definition 8) *pi* is the price of only the transport for a newly entered bid *b* in the local plan, and p^t and p^f are *deviation* prices for the TA to get *to* the new bid from a preceding action in the local plan and *from* the new bid to the succeeding action respectively. This is shown as $p(a_{p,1},b)$ (the subscript *p* refers to *preceding*) and $p(b,a_{s,1})$ (the subscript *s* refers to succeeding) respectively, where $p(x,y)$ is the deviation price to get from *x* to *y*, where *x* and *y* are bids or actions. p^e is the price for the *existing* deviation from the preceding bid action to the succeeding bid action, shown as $p(a_{p,1},a_{s,1})$. p^e has been paid when the actions were previously entered in the TA's local plan and is no longer required, and can therefore be removed from the price of the new bid.

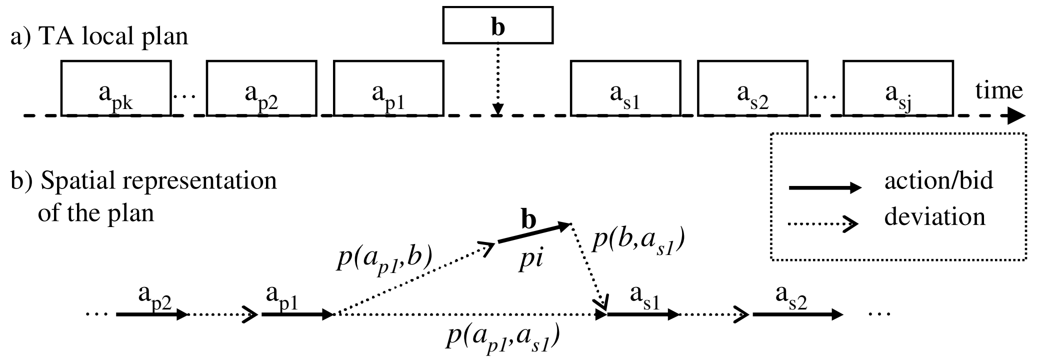


Figure 51. Deviations in the TA's local plan used to calculate a new bid's price. (a) b is a new gap bid that a price must be calculated, which is dependent on current actions a_j (their routes) in TA's local plan. (b) Illustrates the path (and hence deviations) taken by the TA with and without the bid.

Calculations for p^t, p^f and p^e are complicated because actions, which are actually bid(s), in TA's local plan may be provisional, and hence may be rejected. Therefore, if $a_{p,1}$ or $a_{s,1}$ are rejected, the actual deviation prices p^t, p^f and p^e may be different than that calculated and charged for the bid *b*. Thus, the price the MA was charged for the bid *b* may not be equal to the actual price to execute the plan, and hence the TA may have over or under charged.

Sandholm refers to a similar problem in his TRAnsportation COoperation NET (TRACONET) system (Sandholm 1993). He defines a maximum and minimum price that could be charged for the bid by considering the price of the bid with all the possible combinations of actions being rejected or accepted. Setting the price to the minimum is opportunistic and setting the price to the maximum is a safe approach. The prices of the bid if all actions remain (p^{remain}) and all actions are rejected (p^{reject}) are between this maximum and minimum price. Sandholm uses the price p^{reject} such that all of the bids are rejected, because it is computationally easier, providing a semi-opportunistic approach.

In our pricing approach, when accepting a new bid, each TA needs to charge a price that is likely to cover its cost, given that some provisionally granted bids may be rejected. The price that needs to be charged will therefore vary with the price associated with each action and the probability of that action remaining. The effect on the deviation prices p^t, p^f and p^e can therefore be calculated as the mathematical expectation, with price as the value function. Our approach aims at making a better informed choice for bid prices than the approach used by Sandholm, and thus is likely to result in a more accurate bid pricing.

$Pr(a)$ is the probability that action a will remain, and hence will *not* be rejected. We assume that each action is independent, and hence the probability of one action being rejected does not affect the probability of another action being rejected[31]. This simplifies the pricing calculation as conditional probabilities do not need to be calculated. The price, for example, for $p^t = Pr(a_{p,1}).p(a_{p,1}, b) + Pr(a_{p,2}).(1 - Pr(a_{p,1})).p(a_{p,2}, b) + \ldots + Pr(a_{p,k}).(1 - Pr(a_{p,(k-1)})\ldots(1 - Pr(a_{p,1})).p(a_{p,k}, b)$, where $Pr(a_{p,k}).(1 - Pr(a_{p,(k-1)})\ldots(1 - Pr(a_{p,1}))$ is the probability that all actions $a_{p,1}$ to $a_{p,(k-1)}$ will be rejected and $a_{p,k}$ remains, and this probability is multiplied by the deviation price $p(a_{p,k}, b)$ from the action $a_{p,k}$ to the bid b. Even if the (or more than one immediate) preceding and/or succeeding bid action(s) are rejected, resulting in new deviation prices that should have been used in the bid b quoted overall price, the TA has taken this into account in its calculation of the deviations prices, contributing to that deviation price by the amount of its likelihood of occurring.

There are two issues with the calculation of p^t that require further elaboration. Firstly, we do not consider the case where all the actions ($a_{p,1}$ to $a_{p,k}$) are rejected. If there is no action that the TA must travel from, we assume it commences at the location specified at start of the bid. In most situations, the TA will have originated from somewhere (e.g. from a depot). This can be specified by an action $a_{p,k}$ with the destination set to its originating location, and the finish time set to the earliest time that the TA may commence transport. In this case, the probability of action $a_{p,k}$ being rejected is zero.

Secondly, we do not need to calculate the probabilities of all combinations of actions remaining and being rejected preceding the first action before the bid that we consider remaining. For example, consider two actions $a_{p,1}$ and $a_{p,2}$. We define $price(x \cap y, b)$ to be the deviation price between bid b and actions x, y, where x/y implies the action

[31] We believe this is valid in many situations, particulary in a large open market with many independent (and decoupled) agents aquiring transport. For example, scheduling then cancelling a parcel to be delivered by truck may not affect the cancellation of other (potentially many) parcels being delivered by the truck by indeterminate agents in the open market.

remains and *–x*/*-y* implies that the action is rejected. The expectation value can be written as $p^t = Pr(a_{p,1} \cap a_{p,2}).price(a_{p,1} \cap a_{p,2}, b) + Pr(a_{p,1} \cap -a_{p,2}).price(a_{p,1} \cap -a_{p,2}, b) + Pr(-a_{p,1} \cap a_{p,2}).price(-a_{p,1} \cap a_{p,2}, b) + Pr(-a_{p,1} \cap -a_{p,2}).price(-a_{p,1} \cap -a_{p,2}, b)$. As discussed above, we do not consider the situation where all actions are rejected, and therefore the last term can be eliminated. With the first two terms, it is clear that $price(a_{p,1} \cap a_{p,2}, b) = price(a_{p,1} \cap -a_{p,2}, b) = p(a_{p,1}, b)$, i.e. regardless of whether action $a_{p,2}$ remains or is rejected, the deviation price is dependent only on the first remaining preceding action $a_{p,1}$. Therefore, the first two terms become $Pr(a_{p,1} \cap a_{p,2}).\ p(a_{p,1}, b) + Pr(a_{p,1} \cap -a_{p,2}).\ p(a_{p,1}, b) = Pr(a_{p,1}).\ p(a_{p,1}, b)$. The result is the same even if three or more actions preceed the remaining action $a_{p,1}$. As a result, the computation to calculate the expectation value is reduced by 2^j *for each action* $a_{p,k}$ that we are calculating the expectation value for, where j is the number of actions preceding the first remaining action $a_{p,k}$ before the bid. The saving in computation can be extensive if there are many actions in the local plan. A similar argument of the two points above applies for p^f, and thus p^e.

If we have

Eq 36: $\xi_{n+1} = a_{p,(n+1)} \cap \bigcap_{m=1}^{n} (-a_{p,m})$

Eq 37: $\eta_{n+1} = a_{s,(n+1)} \cap \bigcap_{m=1}^{n} (-a_{s,m})$

Eq 38: $\psi = \{\xi_1, \xi_2, ..., \xi_k\} \times \{\eta_1, \eta_2, ..., \eta_j)$

therefore, more formally, the prices for p_t, p_f and p_e are given by

Eq 39: $p^t = \sum_{n=1}^{k} p(a_{p,n}, b) \Pr(\xi_n)$

Eq 40: $p^f = \sum_{n=1}^{j} p(b, a_{s,n}) \Pr(\eta_n)$

Eq 41: $p^e = \sum_{<\xi_u, \eta_v> \in \psi} \left(p(a_{p,u}, a_{s,v}) \Pr(\xi_u \cap \eta_v) \right)$

If there are no previous or following bid actions, then $p^t = 0$ or $p^f = 0$ respectively, and in both cases $p^e = 0$.

Remember that each action in TA's local plan may have many inventories. After the first initial gap bid, and thus inventory item, is created, additional inventory items may be created from piggyback bids. In order for an action to be rejected, every inventory item contained in the action must be rejected. If Ω is the probability of an inventory item remaining, and assume each inventory item, and hence each bid, have the same probability of being rejected, then the probability *Pr(a)* of an action *a* remaining, which contains *m* inventory items, is one minus the probability that all inventory items (or their associated bids) are rejected, thus:

Eq 42: $\Pr(a) = 1 - (1 - \Omega)^m$

The price of the actual transport of a quantity *c* of resources along the specific route for a gap bid is: $pi \geq p^{\min}$

Eq 43: $$pi = \begin{cases} p^{full}\dfrac{c}{cap} & \text{if } p^b \geq p^{\min} \\ p^{\min} & \text{otherwise} \end{cases}$$

where p^{min} is the minimum price the TA requires to transport resources along the particular route, and p^{full} is the price of transporting resources, at full capacity (transporting a quantity of *cap*), along the particular route.

For a *piggyback bid*, the price of deviations and minimum price of the trip have already been paid when the gap bid that it is piggybacking with was first was entered into the local plan. If the bid is only charged the price for the transport (*pi* from *Eq 45*) and the initial inventory item (associated with the gap bid) in the action that it is piggybacking with (the piggybacked action) is rejected, then the TA will need to execute the bid without paying any of the deviation or minimum bid price. Therefore, a piggyback bid must also contribute to the deviation price and minimum price of the transport by the amount of likelihood that all the current inventory items in the piggybacked action are rejected. Using Eq 42, the probability that all m inventory items in piggybacked action a are rejected is $(1 - Pr(a)) = (1 - \Omega)^m$, and thus the price p for the piggyback bid is:

$$\textit{Eq 44:}\quad p = \begin{cases} pi + (1-\Omega)^m (p^t + p^f - p^e + (p^{\min} - p^b)) & \text{if } p^b < p^{\min} \\ pi + (1-\Omega)^m (p^t + p^f - p^e) & \text{otherwise} \end{cases}$$

where the price pi for the piggyback bid alone (without deviation prices) is

$$\textit{Eq 45}:\ pi = \left(p^{full} - \sum_{z=1}^{m} pi_z \right) \left(\frac{c}{cap - \sum_{z=1}^{m} c_z} \right)$$

where pi_z and c_z refer to the cost and quantity of the z^{th} inventory item in the piggybacked action (does not include pi and c of the piggyback bid).

The price p for gap and piggyback bids can be computationally intensive if there are many actions in the local plan that need to be considered for deviations prices. The calculation is simplified with confirm granted bids (inventory items) and using a probability threshold *pth*. If an action contains an inventory item that is confirm granted, then the probability that it will be rejected is assumed to be zero. Therefore, for deviations prices p^t, p^f and p^e, any preceding/succeeding actions after the confirmed action do not need to be considered, and for piggyback bids, only pi needs to be calculated. The probability threshold *pth* is used such that if the probability falls below *pth*, then the TA assumes the bid is confirmed and will not be rejected. *pth* should be set small enough such the price p is not significantly different to the price if *pth* was not used (this depends on what is considered as an acceptable margin of error, and the range of prices that are possible/expected). Therefore, the bid pricing computation can be constrained using *pth*, but still allow informed bid pricing.

With our current experiments, we did not want to focus on the issue of determining an accurate value for Ω. This may not be a trivial task or possible, particularly with the scenarios used. The scenarios used did not allow the TA to receive a large sample of bids and bid rejections over a reasonable time period to obtain a suitable Ω. Based on the rejection rate of test scenarios, we manually set Ω to a value we believed was a reasonable estimate, and used this value for all TA for all simulations that were executed.

7.3.3 Withdrawn Bids

As bids are entered into the TA's local plan, then these bids may conflict with other bids that the TA has previously sent. When the MA tries to provisionally grant a bid, the TA will check if the bid is still available. If the bid has a conflict with other actions (or bids) in the TA's local plan, then the TA sends a withdrawn or provisionally withdrawn message, and may send an updated bid to replace the withdrawn bid. If the bid conflicts with another provisionally granted bid, but not a confirm granted bid, then the bid is provisionally withdrawn. If the bid conflicts with a confirm granted bid, then the bid is withdrawn.

A bid conflicts with another if the times that the bids occur overlap or if the finish location and time of one bid and the start location and time of the other bid are spaced such that the TA cannot move between the two locations in the time required. If we have two bids, $<n_1, c_1, tsa_1, tfa_1, lsa_1, lfa_1, p_1>$ and $< n_2, c_2, tsa_2, tfa_2, lsa_2, lfa_2, p_2>$, then they are in conflict if either of the following conditions are true:

Condition 1: $(tfa_1 > tsa_2) \wedge (tfa_2 > tsa_1)$

Condition 2: $(tfa_1 \leq tsa_2) \wedge (time(lfa_1 \rightarrow lsa_2) > (tsa_2 - tfa_1))$

Condition 3: $(tfa_2 \leq tsa_1) \wedge (time(lfa_2 \rightarrow lsa_1) > (tsa_1 - tfa_2))$

where $time(l_1 \rightarrow l_2)$ is the time required for the TA to get from location l_1 to location l_2.

7.3.4 Updating Bids

If the TA's worst submitted bid (based on cost value) for a task is rejected or withdrawn, the TA will search for an *updated bid*, which is its next best bid, to replace the bid. This is done using the same methods discussed in the previous sections.

An *updated better bid* is sent if an opportunity arises to send a bid that is better than the worst bid currently sent. This may occur, for example, if a bid in the local plan is provisionally rejected, making available resources for the TA to bid for other tasks.

Updated better bids are checked for whenever an “event” occurs in TA’s local plan, which is when a bid is provisional granted, provisional rejected or confirm granted. A provisional grant enters an action in to the local plan, creating up to two new gaps to allow two gap bids, and allowing a piggyback bid with the newly entered action. Although the same bids could have been made before the new action was entered into the TA’s local plan, the price of deviations with the action in place could be greatly reduced, producing a better bid. Additionally, with a piggyback bid, the newly entered action would have paid the minimum transportation price (see

Eq *43*). Thus a piggyback bid to transport a small quantity along that route can now be made cheaper, charging only (or predominantly – see Eq 44) for the small portion of the quantity transported, without the additional minimum transport price. A provisional reject either removes an action from the local plan, thus creating a new gap for gap bids, or it removes an inventory item from the action, and thus creates an opportunity to make a piggyback bid with a larger capacity.

A confirm grant does not change the TA’s local plan, but it secures an action into the local plan. Since the probability of the bid, or action, remaining (not being rejected) is now 100% (we assume they are not rejected after a confirm grant, and if they are, any losses are recovered), the price of existing bids in proximity to the action may have changed, and hence, may have been reduced to allow a better bid. Similarly, when actions or inventory items are added or removed from the local plan, due to grants or rejects, respectively, the price of bids in proximity to the event may change. How close to an event a bid must be in order for its price to change depends on the probability of a bid remaining Ω (see section 7.3.2). If Ω is small, then a bid’s price may change even if the event is a considerable distance in the plan from the bid (i.e. many actions between the bid and the event). This increases the computation because rather than check for updated better bids in a narrow time window in the local plan directly where the event occurs, the TA may need to check for updated better bids in a wider time window in case the event had an impact on bid prices a number of actions away from the event. The TA can restrict the width of the time window either side of the event (to search for updated better bids) such that bids past the edge of the time window have a probability in Eq 39, Eq 40 and Eq 41 for the action at/adjacent-to the event that is below some threshold *pcth*, and therefore, the event will have minimal effect on the pricing of bids outside of the time window.

Ideally, the TA should check all the tasks that it has stored for updated better bids. If the TA has many tasks stored, then checking for updated better bids becomes computationally intensive. This becomes progressively worse as the TA receives more tasks over time. To overcome this problem, the latest tasks that are *used* (announced, provisionally granted or rejected) are stored at the top of a stack. The tasks on the stack are checked, one by one, until some time limit is reached. Tasks that have been used most recently are checked first since they are more likely to still be considered by the MA than older tasks that have not been used, which may because they have already been solved or are no longer available.

7.3.5 Bid Evaluation Function and Cost Value

The MA and TA evaluate the quality of bids based on their cost value *cv*, which is dependent on the price and package delivery time. The lower the *cv*, the better the bid for the particular task. As will be shown later, *cv* for the MA and TA are calculated differently. The MA must calculate the *cv* based on the particular branch in the search tree and the path leading to it, and since there may be more than one root transportation task (task in the root node), there is a *cv* associated with each root task. The TA only considers the *cv* as a function of the task it receives, and the bid that it is considering.

The *cv* should be determined by MA's bid evaluation function *f*. The function to calculate *cv* can be quite complex, but most of it (other than a few variables) remains the same for each task, and it may be considered a common bid evaluation function suitable for many MA for their transportation tasks. Therefore, it may be a waste of communication to submit the complete function for every task announced to every TA. Rather, the MA assumes that the TA have knowledge of the function, and for those that don't, the MA only needs to communicate the complete function to the TA once. Therefore, the MA is only required to provide a small amount of detail (or variables) that is needed within the function to calculate the *cv* for the specific task that is announced. Later, we show which details are required to be announced with each task. The *f* that is submitted with each task refers to the bid evaluation function *details* used to calculate *cv* rather than the complete function. Hence, we refer to the bid evaluation function to calculate *cv* as *cv( )*.

Definition 10: br_i is a branch in MA's search tree, associated with bid $bid(br_i) = b$ for the task $task(br_i) = t$ that is associated with the root task $root(br_i) = tr_v$. ρ is the number of root tasks that must be achieved, so $1 \le v \le \rho$.

Definition 11: *s(tr)* is a function that returns a weighting factor that the MA places on the particular root task *tr*. If $s(tr) > s(tr')$, then the root task *tr* has greater importance than the root task *tr'*.

Using Definition 10 and Definition 11, the bid evaluation function *cv( )* for the MA is

Eq 46: $$cv_{MA}(br_i) = \sum_{k=1}^{\rho} cv(tr_k, br_i) \cdot s(tr_k)$$

Therefore, the *cv* for each branch br_i is the sum of the *cv*'s for each root task at that branch. The *cv* for each root task takes into consideration the current *cv*, taken from the granted bids (path in the search tree) for that root task, and the *cv* that is *expected* to be incurred to achieve the remaining set of tasks that the granted bids did not achieve of the root task. The *cv* for a root task tr_k at branch br_i is given by

Eq 47: $$cv(tr_k, br_i) = w(tr_k) \cdot pd(tr_k, br_i) + (1 - w(tr_k)) \cdot td(tr_k, br_i)$$

where *pd( )* and *td( )* are functions that returns a value that is proportional to the dissatisfaction (how unhappy the MA is) in price and time, respectively, and *w( )* is a weighting factor, allowing the MA to *place priority on either price or time* for the plan for tr_k.

Definition 12: $abip(tr_k, br_i)$ is a function that returns all the bids that are selected for root task tr_k, which are on the path from the root node up to, and including, the branch br_i, in MA's search tree. Using Definition 10, $bid(br_i) \in abip(tr_k, br_i)$ if $root(br_i) = tr_k$, and $bid(br_i) \notin abip(tr_k, br_i)$ otherwise.

Definition 13: *diff(t, b)* is a function which returns the set of (one or more) remaining tasks that still need to be achieved if *b* is selected for the task *t*. $diff(t, b) = \varnothing$ if the bid *b* fully achieves *t*.

Definition 14: *tsta(tr$_k$, br$_i$)* returns a set of β remaining tasks that are associated with the root task *tr$_k$* that still need to be achieved after the branches (bids) from the root node up to and including branch *br$_i$* are selected (i.e. returns the set of tasks associated with *tr$_k$* in the child node of *br$_i$*). Using Definition 10, *diff(task(br$_i$), bid(br$_i$))* $\in$ *tsta(tr$_k$, br$_i$)* if *root(br$_i$) = tr$_k$*, and *diff(task(br$_i$), bid(br$_i$))* $\notin$ *tsta(tr$_k$, br$_i$)* otherwise.

Definition 15: *pdcb(tr, B)* and *tdcd(tr, B)* are functions which take in as their input, a root task *tr* and a set of bids *B*, which are selected to fully or partially achieve *tr*, and return values proportional to the dissatisfaction in price and time, respectively, for the bid(s) *B* to achieve *tr*.

Definition 16: *pdte(tr, T)* and *tdte(tr, T)* are functions that take in as their input, a root task *tr* and a set of tasks *T* that still need to be achieved in order to achieve *tr*, and return a value proportional to the *expected* dissatisfaction in price and time, respectively, to find bids to achieve the remaining tasks in order to achieve *tr*. Therefore, *pdte(tr, T)* and *tdte(tr, T)* require suitable heuristics to estimate the expected price and time required to achieve the collection of remaining unachieved tasks for the root task *tr*.

Using Definition 12, Definition 14, Definition 15, and Definition 16, *pd( )* and *td( )* in *Eq 47* at a particular branch *br$_i$* in MA's search tree for any root task *tr$_k$* are defined as

Eq 48:
$$\begin{aligned} pd(tr_k, br_i) &= pdcb(tr_k, abip(tr_k, br_i)) + pdte(tr_k, tsta(tr_k, br_i)) \\ td(tr_k, br_i) &= tdcb(tr_k, abip(tr_k, br_i)) + tdte(tr_k, tsta(tr_k, br_i)) \end{aligned}$$

Definition 17: The TA receives a task *t* for an associated root task *tr* from the MA, and submits a bid *b* to fully, or partially, achieve it. The bid *b*, when received by the MA, becomes a branch *br$_i$* in its search tree. From Definition 10, *t = task(br$_i$), tr = root(br$_i$)* and *b = bid(br$_i$).* We define *parentTask(bid(br$_i$)) = task(br$_i$).* The announced task *task(br$_i$)* contains the bid evaluation function details *f*, which TA's *cv( )* function uses in order to allow the TA to calculate the *cv* for its potential bids, to submit the bid *bid(br$_i$)* with the lowest *cv* value.

The bid evaluation function *cv( )* for the TA (the TA calculates the *cv* for the bid *bid(br$_i$)* rather than the branch *br$_i$*) is defined as

Eq 49:
$$cv_{TA}(bid(br_i)) = w(root(br_i)) \cdot pd(root(br_i), bid(br_i)) + (1 - w(root(br_i))) \cdot td(root(br_i), bid(br_i))$$

The elements of *cv( )* for the TA are calculated with respect to the announced task *task(br$_i$)*'s associated root task *root(br$_i$)*, and not *task(br$_i$)* itself. The required information regarding the root task is contained within *task(br$_i$)* (in *f*).

Using Definition 13, Definition 15 and Definition 16, we define *pd( )* and *td( )* for the TA as:

Eq 50:
$$pd(root(br_i), bid(br_i)) = pdcb(root(br_i), bid(br_i)) + pdte(root(br_i), diff(task(br_i), bid(br_i)))$$
$$td(root(br_i), bid(br_i)) = tdcb(root(br_i), bid(br_i)) + tdte(root(br_i), diff(task(br_i), bid(br_i)))$$

Definition 18: *Conditions for the bid evaluation function cv( ).*

Two conditions are required for the bid evaluation function *cv( )* for MA and TA. First, if the TA believes that the MA prefers bid *bid(br$_1$)* over bid *bid(br$_2$)* for a particular task, then the MA actually does, i.e. the MA prefers branch *br$_1$* over branch *br$_2$*:

Eq 51: $cv_{TA}(bid(br_1)) < cv_{TA}(bid(br_2)) \rightarrow cv_{MA}(br_1) < cv_{MA}(br_2)$

where *parentTask(bid(br$_1$)) = parentTask(bid(br$_2$)) = task(br$_1$) = task(br$_2$) = t.* The second condition is that *Eq 51* applies if the same bid for the same task is associated with different nodes in MA's search tree. This means that the MA does not need to re-announce the task to the TA for bidding when a new node is created that contains the task (see Figure 48), saving communication and computation time. We define bid *b$_1$ = bid(br$_1$) = bid(br$_3$)*, and bid *b$_2$ = bid(br$_2$) = bid(br$_4$)*, where *br$_1$ ≠ br$_3$*, *br$_2$ ≠ br$_4$*, and *parentTask(b$_1$) = parentTask(b$_2$) = task(br$_1$) = task(br$_2$) = task(br$_3$) = task(br$_4$) = t.* If the TA believes that the MA prefers bid *b$_1$* over *b$_2$* for a particular task, then the MA prefers branch *br$_1$* over *br$_2$*, and branch *br$_3$* over *br$_4$*:

Eq 52: $cv_{TA}(b_1) < cv_{TA}(b_2) \rightarrow [cv_{MA}(br_1) < cv_{MA}(br_2)] \& [cv_{MA}(br_3) < cv_{MA}(br_4)]$

Theorem 9: Both conditions *Eq 51* and *Eq 52* in Definition 18 are satisfied for the bid evaluation function *cv( )* if the functions *pdcb( )*, *pdte( )*, *tdcb( )* and *tdte( )* in *Eq 48* and *Eq 50* for MA and TA, respectively, are *linear* functions, i.e. satisfy

Eq 53: $$g(x \cup y) = g(x) + g(y)$$

Proof

There are two parts to the proof – showing that *Eq 53* satisfies both conditions *Eq 51* and *Eq 52*.

Proof part 1: Eq 53 satisfies condition Eq 51

From *Eq 51*, substituting *Eq 46*, *Eq 47*, *Eq 48*, *Eq 49*, *Eq 50*, tr_v = *root(br$_i$)* for so $1 \le v \le \rho$, extracting the *cv* for the root task tr_v in the summation in *Eq 46*, and using Definition 18 where the root task for *t* is tr_v, we have

Eq 54:
$$\begin{aligned}
&w(tr_v)\cdot[pdcb(tr_v,bid(br_1)) + pdte(tr_v,diff(t,bid(br_1)))] + \\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,bid(br_1)) + tdte(tr_v,diff(t,bid(br_1)))] < \\
&w(tr_v)\cdot[pdcb(tr_v,bid(br_2)) + pdte(tr_v,diff(t,bid(br_2)))] + \\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,bid(br_2)) + tdte(tr_v,diff(t,bid(br_2)))] \rightarrow \\
&[w(tr_v)\cdot[pdcb(tr_v,abip(tr_v,br_1)) + pdte(tr_v,tsta(tr_v,br_1))] + \\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,abip(tr_v,br_1)) + tdte(tr_v,tsta(tr_v,br_1))]]\cdot s(tr_v) + \\
&\quad \sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_1)\cdot s(tr_k) < \\
&[w(tr_v)\cdot[pdcb(tr_v,abip(tr_v,br_2)) + pdte(tr_v,tsta(tr_v,br_2))] + \\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,abip(tr_v,br_2)) + tdte(tr_v,tsta(tr_v,br_2))]]\cdot s(tr_v) + \\
&\quad \sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_2)\cdot s(tr_k)
\end{aligned}$$

and from Definition 12, where br_1 and br_2 are associated with the root task tr_v, we have

Eq 55:
$$\begin{aligned}
abip(tr_v,br_1) &= \{bid(br_1)\} \cup \{b \mid b \neq bid(br_1)\} = \{bid(br_1)\} \cup B \\
abip(tr_v,br_2) &= \{bid(br_2)\} \cup \{b \mid b \neq bid(br_2)\} = \{bid(br_2)\} \cup B
\end{aligned};$$

from Definition 13, say

Eq 56:
$$\begin{aligned}
diff(t,bid(br_1)) &= T_1 \\
diff(t,bid(br_2)) &= T_2
\end{aligned}$$

and therefore, from Definition 14 and *Eq 56*, and since tr_v is the root task for *t*, we have

Eq 57:
$$\begin{aligned} &tsta(tr_v, br_1) = T_1 \cup \{t' \mid t' \notin T_1\} = T_1 \cup T \\ &tsta(tr_v, br_2) = T_2 \cup \{t' \mid t' \notin T_2\} = T_2 \cup T \end{aligned}$$

substituting *Eq 55*, *Eq 56* and *Eq 57* in *Eq 54*, we get

Eq 58:

$$\begin{aligned}
&w(tr_v) \cdot [pdcb(tr_v, bid(br_1)) + pdte(tr_v, T_1)] + (1 - w(tr_v)) \cdot [tdcb(tr_v, bid(br_1)) + tdte(tr_v, T_1)] < \\
&w(tr_v) \cdot [pdcb(tr_v, bid(br_2)) + pdte(tr_v, T_2)] + (1 - w(tr_v)) \cdot [tdcb(tr_v, bid(br_2)) + tdte(tr_v, T_2)] \rightarrow \\
&\big[w(tr_v) \cdot [pdcb(tr_v, \{bid(br_1)\} \cup B) + pdte(tr_v, T_1 \cup T)] + \\
&\qquad (1 - w(tr_v)) \cdot [tdcb(tr_v, \{bid(br_1)\} \cup B) + tdte(tr_v, T_1 \cup T)]\big] \cdot s(tr_v) + \sum_{k \in \{1..\rho\}, k \neq v} cv(tr_k, br_1) \cdot s(tr_k) < \\
&\big[w(tr_t) \cdot [pdcb(tr_v, \{bid(br_2)\} \cup B) + pdte(tr_v, T_2 \cup T)] + \\
&\qquad (1 - w(tr_v)) \cdot [tdcb(tr_v, \{bid(br_2)\} \cup B) + tdte(tr_v, T_2 \cup T)]\big] \cdot s(tr_v) + \sum_{k \in \{1..\rho\}, k \neq v} cv(tr_k, br_2) \cdot s(tr_k)
\end{aligned}$$

Making the left hand side (first two lines) of *Eq 58* equal to $n_1 < n_2$, and using *Eq 53* for *pdcb( ), pdte( ), tdcb( ), tdte( ),* we have

Eq 59:

$$\begin{aligned}
&n_1 < n_2 \rightarrow \\
&\big[w(tr_v) \cdot [pdcb(tr_v, bid(br_1)) + pdcb(tr_v, B) + pdte(tr_v, T_1) + pdte(tr_v, T)] + \\
&\qquad (1 - w(tr_v)) \cdot [tdcb(tr_v, bid(br_1)) + tdcb(tr_v, B) + tdte(tr_v, T_1) + tdte(tr_v, T\})]\big] \cdot s(tr_v) + \\
&\qquad \sum_{k \in \{1..\rho\}, k \neq v} cv(tr_k, br_1) \cdot s(tr_k) < \\
&\big[w(tr_v) \cdot [pdcb(tr_v, bid(br_2)) + pdcb(tr_v, B) + pdte(tr_v, T_2) + pdte(tr_v, T)] + \\
&\qquad (1 - w(tr_{tv})) \cdot [tdcb(tr_v, bid(br_2)) + tdcb(tr_{tv}, B) + tdte(tr_v, T_2) + tdte(tr_v, T)]\big] \cdot s(tr_v) + \\
&\qquad \sum_{k \in \{1..\rho\} k \neq v} cv(tr_k, br_2) \cdot s(tr_k)
\end{aligned}$$

which becomes

Eq 60:

$$\begin{aligned}
&n_1 < n_2 \rightarrow \\
&\big[w(tr_v)\cdot[pdcb(tr_v,bid(br_1))+pdte(tr_v,T_1)]+(1-w(tr_v))\cdot[tdcb(tr_v,bid(br_1))+ \\
&\qquad tdte(tr_v,T_1)]\big]\cdot s(tr_v)+\big[w(tr_v)\cdot[pdcb(tr_v,B)+pdte(tr_v,T)]+ \\
&\qquad (1-w(tr_v))\cdot[tdcb(tr_v,B)+tdte(tr_v,T)]\big]\cdot s(tr_v)+\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_1)\cdot s(tr_k) < \\
&\big[w(tr_v)\cdot[pdcb(tr_v,bid(br_2))+pdte(tr_{tv},T_2)]+(1-w(tr_v))\cdot[tdcb(tr_v,bid(br_2))+ \\
&\qquad tdte(tr_v,T_2)]\big]\cdot s(tr_v)+\big[w(tr_v)\cdot[pdcb(tr_v,B)+pdte(tr_v,T)]+ \\
&\qquad (1-w(tr_v))\cdot[tdcb(tr_v,B)+tdte(tr_v,T)]\big]\cdot s(tr_v)+\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_2)\cdot s(tr_k)
\end{aligned}$$

Looking at the summations in *Eq 60*, and using *Eq 47* and *Eq 48*,

Eq 61:

$$\begin{aligned}
&\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_1)\cdot s(tr_k) = \sum_{k\in\{1..\rho\},k\neq v} [w(tr_k)\cdot pdcb(tr_k,abip(tr_k,br_1))+pdte(tr_k,tsta(tr_k,br_1)) \\
&\qquad +(1-w(tr_k))\cdot tdcb(tr_k,abip(tr_k,br_1))+tdte(tr_k,tsta(tr_k,br_1))]\cdot s(tr_k) \\
&\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_2)\cdot s(tr_k) = \sum_{k\in\{1..\rho\},k\neq v} [w(tr_k)\cdot pdcb(tr_k,abip(tr_k,br_2))+pdte(tr_k,tsta(tr_k,br_2)) \\
&\qquad +(1-w(tr_k))\cdot tdcb(tr_k,abip(tr_k,br_2))+tdte(tr_k,tsta(tr_k,br_2))]\cdot s(tr_k)
\end{aligned}$$

Since the summations in *Eq 61* do not consider root tasks that are associated with the branches br_1 and br_2 ($k \neq v$), then from Definition 12, $abip(tr_k, br_1) = abip(tr_k, br_2)$ [32] and from Definition 14, $tsta(tr_k, br_1) = tsta(tr_k, br_2)$ [33], therefore

Eq 62: $$\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_1)\cdot s(tr_k) = \sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_2)\cdot s(tr_k)$$

[32] The only difference in the set of bids (branches) in the path from the root node to the branches br_1 and br_2 (which are at the same node) are $bid(br_1)$ and $bid(br_2)$, respectively. *abip( )* only returns bids $bid(br_1)$ and $bid(br_2)$ in the list of bids associated with the root task of br_1 and br_2, i.e. root task tr_v – see *Eq 55*. Therefore the list of bids returned that are in the paths ending at br_1 and br_2 for all the other root tasks are the same.

[33] Similarly, the difference in the tasks in the new node created by branches br_1 and br_2 are $diff(tr_v, bid(br_1))$ and $diff(tr_v, bid(br_1))$, respectively, which *tsta( )* only returns them in the list associated with the root task of br_1 and br_2, i.e. tr_v – see *Eq 57*. Thus, the list of tasks that are in the new node associated with br_1 and br_2 for the other root tasks are the same.

so from *Eq 62*, substituting the relevant n_1 and n_2, and making all constant values (for different *bid(br₁)* and *bid(br₂)*) on either side of the inequality equal to c_1, *Eq 60* reduces to

Eq 63: $n_1 < n_2 \rightarrow n_1 \cdot s(tr_v) + c_1 < n_2 \cdot s(tr_v) + c_1$

Eq 63 is consistent, and thus we complete the first part of the proof.

Proof part 2: Eq 53 satisfies condition Eq 52

Using br_3 and br_4 in Definition 12, Definition 13 and Definition 14 (see *Eq 55*, *Eq 56*, *Eq 57*), we have

Eq 64:
$$\begin{aligned} abip(tr_v, br_3) &= \{bid(br_3)\} \cup \{b \mid b \neq bid(br_3)\} = \{bid(br_3)\} \cup B' \\ abip(tr_v, br_4) &= \{bid(br_4)\} \cup \{b \mid b \neq bid(br_4)\} = \{bid(br_4)\} \cup B' \end{aligned}$$

Eq 65:
$$\begin{aligned} diff(t_v, bid(br_3)) &= T_3 \\ diff(t_v, bid(br_4)) &= T_4 \end{aligned}$$

Eq 66:
$$\begin{aligned} tsta(tr_v, br_3) &= T_3 \cup \{t' \mid t \notin T_3\} = T_3 \cup T' \\ tsta(tr_v, br_4) &= T_4 \cup \{t' \mid t \notin T_4\} = T_4 \cup T' \end{aligned}$$

Consider $cv_{MA}(br_3) < cv_{MA}(br_4)$ in *Eq 52*, using *Eq 46*, *Eq 47*, *Eq 48* , *Eq 64*, *Eq 65*, and *Eq 66*, we obtain

Eq 67:

$$\begin{aligned} &\big[w(tr_v) \cdot [pdcb(tr_v, \{bid(br_3)\} \cup B') + pdte(tr_v, T_3 \cup T')] + \\ &\quad (1 - w(tr_v)) \cdot [tdcb(tr_v, \{bid(br_3)\} \cup B') + tdte(tr_v, T_3 \cup T')]\big] \cdot s(tr_v) + \sum_{k \in \{1..\rho\}, k \neq v} cv(tr_k, br_3) \cdot s(t_k) < \\ &\big[w(tr_v) \cdot [pdcb(tr_v, \{bid(br_4)\} \cup B') + pdte(tr_v, T_4 \cup T')] + \\ &\quad (1 - w(tr_v)) \cdot [tdcb(tr_v, \{bid(br_4)\} \cup B') + tdte(tr_v, T_4 \cup T')]\big] \cdot s(tr_v) + \sum_{k \in \{1..\rho\}, k \neq v} cv(tr_k, br_4) \cdot s(t_k) \end{aligned}$$

using *Eq 53* for *pdcb( ), pdte( ), tdcb( ), tdte( )*, we have

Eq 68:

$$
\begin{aligned}
&\big[w(tr_v)\cdot[pdcb(tr_v,bid(br_3))+pdte(tr_v,T_3)]+(1-w(tr_v))\cdot[tdcb(tr_v,bid(br_3))+\\
&\quad tdte(tr_v,T_3)]\big]\cdot s(tr_v)+\big[w(tr_v)\cdot[pdcb(tr_v,B)+pdte(tr_v,T)]+\\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,B)+tdte(tr_v,T)]\big]\cdot s(tr_v)+\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_3)\cdot s(tr_k)<\\
&\big[w(tr_v)\cdot[pdcb(tr_v,bid(br_4))+pdte(tr_v,T_4)]+(1-w(tr_v))\cdot[tdcb(tr_v,bid(br_4))+\\
&\quad tdte(tr_v,T_4)]\big]\cdot s(tr_v)+\big[w(tr_v)\cdot[pdcb(tr_v,B)+pdte(tr_v,T)]+\\
&\quad (1-w(tr_v))\cdot[tdcb(tr_v,B)+tdte(tr_v,T)]\big]\cdot s(tr_v)+\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_4)\cdot s(tr_k)
\end{aligned}
$$

similarly with *Eq 62*, the summations in *Eq 68* are equivalent

Eq 69: $\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_3)\cdot s(tr_k)=\sum_{k\in\{1..\rho\},k\neq v} cv(tr_k,br_4)\cdot s(tr_k)$

so from *Eq 69*, substituting n_1 and n_2 from *Eq 58* and *Eq 59*, and making all constant values (for different *bid(br₃)* and *bid(br₄)*) on either side of the inequality equal to c_2, *Eq 68* reduces to

Eq 70: $n_1\cdot s(tr_v)+c_2<n_2\cdot s(tr_v)+c_2$

Using *Eq 63* and *Eq 70* to form *Eq 52*, we have

Eq 71: $n_1<n_2\rightarrow[n_1\cdot s(tr_v)+c_1<n_2\cdot s(tr_v)+c_1]\,\&\,[n_1\cdot s(tr_v)+c_2<n_2\cdot s(tr_v)+c_2]$

which is consistent, and thus we complete the second part of the proof.

Q.E.D.

7.3.6 Bid Evaluation Function Components

We define B_k as a set of bids associated with root task tr_k, and T_k as a set of (remaining) tasks associated with tr_k or the empty set (if no tasks remaining). *eprt*(tr_k) returns the price that the MA expects to pay to achieve the root task tr_k, *price(b)* returns the price of the bid *b*, and *expectedPrice(t)* returns the price that the MA or TA expects the MA would need to pay, in order to achieve the (remaining) task *t*. *etqrt*(tr_k) returns the product of the expected delivery time and quantity of all the packages to be delivered in the root task tr_k, *quantity(x)* returns the quantity specified in the bid or task *x*, *destination*(tr_k) returns the delivery destination for packages associated with root task tr_k, *deliveryTime( )*

will return the actual delivery time of a bid, and *expectedDeliveryTime( )* will return the expected delivery time of (remaining) tasks. In our implementation, the functions *pdcb( ), pdte( ), tdcb( ),* and *tdte( ),* that are used in the bid evaluation function *cv( )*, for both MA and TA, are therefore defined as

Eq 72: $$pdcb(tr_k, B_k) = \frac{\sum_{\forall b \in B_k} price(b)}{eprt(tr_k)}$$

Eq 73: $$pdte(tr_k, T_k) = \frac{\sum_{\forall t \in T_k} \exp ected \Pr ice(t)}{eprt(tr_k)}$$

Eq 74: $$tdcb(tr_k, B_k) = \frac{\sum_{\forall b \in B_k} deliveryTime(b, destination(tr_k)) \cdot quantity(b)}{etqrt(tr_k)}$$

Eq 75: $$tdte(tr_k, T_k) = \frac{\sum_{\forall t \in T_k} \exp ectedDeliveryTime(t, destination(tr_k)) \cdot quantity(t)}{etqrt(tr_k)}$$

The functions above are consistent with Theorem 9. Functions for the dissatisfaction in price, *pdcb( )* and *pdte( )*, are straight forward. *expectedPrice(t)* depends on the quantity and distance of the transportation task *t*, the type of TA (their price) that service the route, transfer prices, and the number of expected transfers required to complete the transport. Transfer prices are the prices to transfer the package between TA (for partial route transport), and the source and destination, and includes prices for loading, unloading, and storage.

Functions for the dissatisfaction in time, *tdcb( )* and *tdte( )*, are a little more involved, where time is a real number, representing hours from some arbitrary start point in time. $etqrt(tr_k) = etrt(tr_k) \cdot quantity(tr_k)$, where $etrt(tr_k)$ returns the time that the MA expects is required in order to deliver all the package(s) specified in the root task tr_k, and $quantity(tr_k)$ is the quantity to be transported in root task tr_k. *etqrt( )* is used to normalise the value returned in the numerator of *Eq 74* and *Eq 75* for the specific root task, as in *Eq 72* and *Eq 73*. The delivery time is a function of the root task's destination ($destination(tr_k)$), as opposed to the destination specified in the announced task that the

TA is bidding for. The destinations may be different as partial route bids are allowed, and thus the announced task's destination may be to achieve a route part of the way to the root task destination. In this case, the delivery time associated with the announced task does not affect the delivery time of the root task. Hence, *deliveryTime( ) = expectedDeliveryTime( ) = 0. expectedDeliveryTime( )* depends on the quantity and distance of the transport, the type of TA (their speed) that service the route, transfer times, and the number of expected transfers. Transfer times depend on the time to transfer the package between TA (for partial route), and the source and destination, and includes times for loading, unloading, and refuelling. *expectedDeliveryTime( )* also considers the number of trips that may be required by the TA if the quantity is large. Since resources are limited, even if all the available TA were used, they may not be able to carry the complete quantity in one trip, and hence all the TA must make multiple trips.

The calculation for the dissatisfaction in time multiplies the delivery time with the quantity of packages that is delivered at that time, which keeps *tdcb( )* and *tdte( )* linear (satisfies Theorem 9). Therefore, delivering a large quantity early (at τ_1) together with a small quantity much later (at τ_2) may be better than delivering the complete quantity at a time τ_3, where $\tau_1 < \tau_3 << \tau_2$. If we had chosen to consider the latest time that the last package was delivered for the root task, regardless of what was delivered before then, the dissatisfaction in time could be $td(\ ) = [max(deliveryTime(B_k,\ destination(tr_k)) \cup expectedDeliveryTime(T_k,\ destination(tr_k))) / etqrt(tr_k)]$, see *Eq 48* and *Eq 50*, and in this case, $etqrt(\ ) = etrt(tr_k)$. Therefore, *td( )* would not be linear (not satisfy Theorem 9), and thus will not satisfy condition *Eq 52* in Definition 18, requiring the MA to re-announce the task every time it appears in a new node in its search tree. Note that condition *Eq 51* in Definition 18 can be satisfied in this case, if the MA provides information with the announced task regarding the maximum delivery time of the currently selected bids, and remaining tasks, for the root task at the particular node (and path). The TA can take the maximum of the time supplied by the MA, and the time that TA's bid delivers the package. The MA will do the same when calculating the *cv( )* for the same bid.

Both heuristic functions *expectedPrice( )* and *expectedDeliveryTime( )* for MA and TA should return the same value when the same input is used, otherwise they may have differing beliefs of which bids are preferred (lowest *cv*). In our prototype, the MA and TA were coded such that the functions within the bid evaluation function were the same, i.e. had the same formulae and values, ensuring that all agents' bid evaluation functions produce consistent outputs. In a deployed system, developers may not have control over which formulae and values others may use in their agents. In such a case, the MA may announce the complete bid evaluation function (formulae and values) to the TA, which only needs to be done once when one of the initial root tasks is announced.

Due to the military sensitivity of the information, we are unable to provide further details of the internal functions and values used to calculate the bid evaluation function *cv( )*.

7.3.7 Bid Evaluation Function Details

All announced tasks contain the bid evaluation function details *f*, which the TA requires to calculate the bid evaluation function *cv( )*. Using Definition 17, *Eq 49*, *Eq 50*, *Eq 72*, *Eq 73*, *Eq 74* and *Eq 75*, the TA has a bid *b = bid(brᵢ)* for an announced task *t = task(brᵢ)*, that is associated with the root task *root(brᵢ)* which must transport a package to location *destination(root(brᵢ))* as soon as possible with minimum price. The bid evaluation function *cv( )* for the TA is

Eq 76:

$$
\begin{aligned}
cv_{TA}(bid(br_i)) = {} & w(root(br_i)) \cdot \left(\frac{price(bid(br_i))}{eprt(root(br_i))} + \frac{\sum_{\forall t \in diff(task(br_i), bid(br_i))} expected\,Price(t)}{eprt(root(br_i))} \right) + \\
& (1 - w(root(br_i))) \cdot \left(\frac{deliveryTime(bid(br_i), destination(root(br_i))) \cdot quantity(bid(br_i))}{etqrt(root(br_i))} + \right. \\
& \left. \frac{\sum_{\forall t \in diff(task(br_i), bid(br_i))} expectedDeliveryTime(t, destination(root(br_i))) \cdot quantity(t)}{etqrt(root(br_i))} \right)
\end{aligned}
$$

which becomes

Eq 77:

$$cv_{TA}(bid(br_i)) = \frac{w(root(br_i))}{eprt(root(br_i))} \cdot \left(price(bid(br_i)) + \sum_{\forall t \in diff(task(br_i), bid(br_i))} \exp ected \Pr ice(t) \right) +$$

$$\frac{(1 - w(root(br_i)))}{etqrt(root(br_i))} \cdot \big(deliveryTime(bid(br_i), destination(root(br_i))) \cdot quantity(bid(br_i)) +$$

$$\left. \sum_{\forall t \in diff(task(br_i), bid(br_i))} \exp ectedDeliveryTime(t, destination(root(br_i))) \cdot quantity(t) \right)$$

and therefore, from *Eq 77*, information *f* that the MA must provide to the TA to calculate the *cv( )* for the specific announced task $t = task(br_i)$ associated with root task $root(br_i)$ is (assuming the TA contains all other formulae and values to calculate the functions using task $t = task(br_i)$ and bid $b = bid(br_i)$ as inputs):

Eq 78: $$f = \left\{ \frac{w(root(br_i))}{eprt(root(br_i))}, \frac{(1 - w(root(br_i)))}{etqrt(root(br_i))}, destination(root(br_i)) \right\}$$

7.4 Theoretical Analysis

7.4.1 Time Complexity

The time complexity for MA's planning was discussed in section 5.3.5. We now discuss the time complexity for TA to find suitable bids for the MA's announced task. PAP's bidding deadline should take into consideration the time for TA to find and submit bids.

The complexity for the TA to compute a bid for a task for the MA is dependent on the number of actions in the TA's local plan, and the number of routes that they can service. If there are θ actions in TA's local plan, then there is *(θ + 1)* gaps in the local plan, assuming that the time window that the transportation task must be achieved within is large enough to encapsulate the complete local plan. Therefore, there would be at maximum θ piggyback bids possible, one for each action, and *((θ + 1) × r)* gap bids possible, where *r* is the number of possible routes that can be serviced by the TA. The total number of bids (*tnb*) that can be made is

Eq 79: $tnb = \theta + (\theta + 1) \cdot r = \theta + \theta \cdot r + r$

For each bid made, calculations for capacity (uses maximum possible capacity), time (uses earliest time possible) and bid evaluation function *cv( )* (price and time dissatisfaction for the bid and up to a *maximum* of three tasks, which the bid does not achieve) are easy – the time complexity is fixed for each bid. The price of each bid is dependent on the number of actions in the TA's local plan, and in the worst case, there are no confirmed inventory items in actions, $\Omega < 100\%$ and $pth = 0.0$ (see section 7.3.2) so the TA considers all actions in the plan to calculate the bid price, and the bid is half way ($\theta/2$) through the plan. To calculate the deviation prices p^t and p^f, the price for the individual deviations between the bid and all other actions is required, hence θ prices need to be considered. For the existing deviation p^e, for each action a_i on one side of the bid in the local plan, the deviation price between a_i and all the actions a_j on the other side of the bid in the local plan must be considered. Therefore, $(\theta/2)\times(\theta/2) = (\theta^2/4)$ prices must be considered. The time complexity for calculating the price (*ccp*) for each bid is

Eq 80: $ccp = \theta + \frac{\theta^2}{4}$

When a task is received, all the routes *r* are scanned to calculate the amount of the task that the route achieves. Then, for every gap being checked for bids, the routes are scanned twice to calculate the amount of the gap that the route achieves, and a list that is the combination of both lists. The time complexity for calculating the lists of routes *clr* during the bidding process is

Eq 81: $clr = r + (\theta + 1)\cdot(2\cdot r) = 2\cdot\theta\cdot r + 3\cdot r$

The overall time complexity of calculating bids for the TA is $(tnb \times ccp) + clr$, so using *Eq 79*, *Eq 80* and Eq 81, we have

Eq 82: $(tnb \times ccp) + clr = (\theta + \theta\cdot r + r)\cdot(\theta + \frac{\theta^2}{4}) + (2\cdot\theta\cdot r + 3\cdot r)$

Therefore, the time complexity is $O((tnb \times ccp) + clr)$, which is of the order

Eq 83: $\textit{Time complexity for TA bidding} = O(\theta^3 \cdot r)$

This complexity applies for finding updated bids for rejected or withdrawn bids. When checking for updated better bids for received tasks, in the worst case, *pcth = 0.0* (see section 7.3.4) and the task's time window encapsulates the whole plan, so bids throughout the TA's complete local plan are checked if their prices have changed for the better. If δ is the number of received tasks that the TA checks for better bids, all possible bids for all tasks need to checked for better bids, and therefore the time complexity is given by $O(\delta \times ((tnb \times ccp) + clr))$, which is of order

Eq 84: Time complexity for updated better bids = $O(\theta^3 \cdot r \cdot \delta)$

With *tnb*, θ is limited by the task's time window and *r* is limited by the route threshold *rth* (see secion 7.3.1) and the number of routes that the TA will consider for each gap *N*. With *ccp*, θ is limited by the probability threshold *pth* or presence of any confirmed bids. With *clr*, θ is limited by the task's time window, but all *r* must be checked in order to see if the route is suitable. For updated better bids, θ in *tnb* and *clr* is limited by the threshold *pcth* and the task's time window. The number of tasks δ checked for better bids can be limited by having a threshold on the number of tasks checked, or having a time limit, where the TA checks as many tasks as it can within some time frame. Although δ can grow quite large as tasks are continually received throughout the life of the agent, some tasks may be withdrawn, and old tasks may be considered to be expired, in which case these tasks can be deleted.

The greater the restriction that TA places on the search for bids, i.e. increasing rth, pth, pcth, and decreasing N, the greater the chance that the quality of the solution will suffer. A bid which may be better than those found may have been overlooked as it fell outside of the restrictions. θ in *tnb* and *clr* can be limited for a very large task time window, without sacrificing the quality of the solution, by using the *cv( )*. Since *cv( )* is dependent on time, there may be a point in time in the local plan where if the TA checks for bids beyond that point, the time dissatisfaction in the *cv( )* for the bids will be large enough such that the *cv( )* will not be less than the smallest *cv( )* of a bid that has been currently found.

7.4.2 Communication

General communication requirements for PAP were discussed in section 5.3.6. Here we present communication requirements for our particular PAP application, the global transportation scheduling implementation, with greater accuracy than the general equations presented in chapter 5. Additionally, we present the communication saved from not having to re-announce tasks that appear in multiple nodes (see Figure 48). From *Eq 4* section 5.3.5, if *br* is the branching factor in MA's search tree and *m* is the depth, then

Eq 85: Number of nodes and branches in MA's search tree $= \sum_{i=1}^{m} br^i$

Each node has transportation tasks associated with it, where only new tasks are announced. Since each node may contain two new additional tasks than the node before it, the maximum number of tasks that would have to be announced ψ is (using *Eq 85*)

Eq 86: $\psi = \rho + 2 \cdot \sum_{i=1}^{m} br^i$

where ρ is the number of root tasks in the root node. The MA may communicate the ψ tasks to all the TA, therefore the maximum number of tasks that the MA must send to TA *mta(MA)*[34], where κ is the maximum number of TA, is

Eq 87: $mta(MA) = \kappa \cdot \psi$

Each TA may receive all the announced tasks, therefore the maximum number of tasks that each TA may receive *mta(TA)*, is

Eq 88: $mta(TA) = \psi$

For each announced task, TA may send a maximum of β bids, where

Eq 89: $\beta \cdot \kappa \cdot \eta = br$

where η is the maximum number of tasks that can be contained within a node, given by

[34] *ac* (auctioneer) and *bd* (bidder) are used chapter 5 section 5.3.6, which are analogous to MA and TA, respectively.

Eq 90: $\eta = \rho + 2 \cdot m$

Therefore, the maximum number of bids that TA may submit during the planning process *mb(TA)* is

Eq 91: $mb(TA) = \psi \cdot \beta$

Thus, the maximum number of bids that the MA may receive *mb(TA)* is

Eq 92: $mb(MA) = \psi \cdot \beta \cdot \kappa$

Note that the number bids received by the MA may be less than the number of branches searched in the planning tree (*Eq 85*) because tasks may be repeated in different nodes, and not re-announced (if conditions for the bid evaluation function *cv( )* are met – see Definition 18). Therefore, the bids (branches) associated with the repeated tasks are also repeated at the different nodes.

In the worst case, every branch searched may require a *maximum* of three speech acts – provisional grant, provisional grant accepted, and either provisional reject or confirm grant (for the branches in the path of the solution). The MA must send the speech acts for each branch in the planning tree, therefore, the maximum number of speech acts that the MA must send or receive *mns(MA)* is

Eq 93: $mns(MA) = 3 \cdot \sum_{i=1}^{m} br^{i}$

where *(2/3)·mns(MA)* is the maximum number of speech acts that the auctioneer sends, and *(1/3)·mns(MA)* is the maximum number of speech acts that the auctioneer receives. The TA only communicates speech acts for bids that it submits to the TA. From *Eq 89*, there are *κ* TA that send a maximum of *β* bids for each task, creating *κ· β* branches for each task in every node in MA's planning tree. Therefore, each TA contributes *1/κ* branches of MA's complete search tree. The maximum number of speech acts that the TA must send or receive *mns(MA)* is

Eq 94: $mns(TA) = 3 \cdot \frac{\sum_{i=1}^{m} br^{i}}{\kappa}$

where *(2/3)·mns(TA)* is the number of speech acts that the bidder receives, and *(1/3)·mns(TA)* is the number of speech acts that the bidder sends.

The total maximum communication requirements for MA and TA are

Eq 95: $tc(MA) = mta(MA) + mb(MA) + mns(MA)$

Eq 96: $tc(TA) = mta(TA) + mb(TA) + mns(TA)$

As previously mentioned, if the bid evaluation function *cv( )* satisfied two conditions specified in Definition 18, only new tasks in MA's search tree need to be announced. This saves communication by not having to re-announce tasks that appear in multiple nodes. The maximum number of tasks *mxt* in all nodes in the planning tree is

Eq 97: $mxt = \sum_{i=0}^{m} (\rho + 2 \cdot i) \cdot br^i$

The amount of communication saved for announced tasks *csat* is therefore (using *Eq 86* and *Eq 97*)

Eq 98: $csat = mxt - \psi = \left(\sum_{i=0}^{m} (\rho + 2 \cdot i) \cdot br^i \right) - \left(\rho + 2 \cdot \sum_{i=0}^{m} br^i \right)$

where the amount of communication saved for *mta(MA), mta(TA), mb(MA),* and *mb(TA),* are *κ·csat, csat, β·κ·csat,* and *β·csat*, respectively. This could be a significant saving in communication.

7.4.3 Convergence, Livelock and Deadlock

Theorem 10: Our global transportation scheduling implementation will converge, and thus prevent livelock, and will not deadlock.

Proof:

From Theorem 8, planning using PAP will not deadlock. From Theorem 7, planning using PAP will converge if (a) there are a finite number of bids and (b) each bid for a task (or set of tasks) is granted a finite number of times. Condition (b) is satisfied because our current implementation of PAP discards withdrawn or rejected bids, and therefore, bids can only be provisionally granted once.

Condition (a) is also satisfied for the following reasons. In our implementation, the TA stores all bids so that they do not send the same or similar bid more than once. Say *r* is the maximum (finite) number of routes that a TA can service. Consider a static environment where the TA's local plan does not change, and there are a finite number of actions in the local plan (within the time window that the transportation task must be achieved). In each gap, the TA submits bids that use maximum capacity possible and starts at the earliest time possible. Since the capacity and time is fixed for each route in the gap, the maximum number of bids per gap is r, one for each route. The maximum number of bids possible for a task is one piggyback bid for each action in the local plan, plus *r* bids for each gap in the local plan.

Consider the case where TA's local plan is continually changing. Changing actions and gaps can potentially allow unlimited number of possible bids for the TA, as every change can allow a new type of bid to be submitted – a slight change in capacity or time for bids in the new gap or action, or changes in the local plan altering prices of current bids. An important condition is either that the transportation task contains a time window that it must be achieved within, or that the time in TA's local plan does not extend to plus or minus infinity, thus constraining the time window in the local plan that can be considered for bids. In our implementation, the former is the case. The TA in our implementation do not submit bids with the same route and *similar* time and price (e.g. time within a few minutes, and price below some percentage threshold) to a previously sent bid. Therefore, bids with the same route and capacity will not be submitted if they have infinitesimal differences in time and price, thus the number of possible bids within the time window is finite (assuming the possible price range is finite, which is the case in our implementation). The capacity in bids for a TA can be defined as *{p, w},* where *p* is the number of sub-packages (which can not be broken down into smaller packages, and is finite) and *w* is the weight of each sub-package. The capacity (or quantity) in the bid that is to be transported is therefore the product of *p* and *w*. The maximum number of bids, with the same route and time, that has different capacity, is *p*. Hence there is a finite number of possible bids with different capacity. Therefore, in a dynamic environment, each TA will submit a finite number of bids.

Q.E.D.

7.4.4 Scalability

We will briefly look at how our implementation of the global transportation scheduling implementation using PAP scales well as TA and MA are added to the system. As TA are added, computation time for other TA should not be affected. Ideally, TA should be executed on different processors, and therefore adding TA should not affect the computation of other TA. Computation time of the MA is dependent on the bidding deadline and the processing of bids (to select a bid for granting). The bidding deadline is only affected if the communication time to announce the task to many TA, and receive bids from many TA, is significant. The MA must therefore increase its bidding deadline to allow enough time for communication. Otherwise, the MA will continue planning after the bidding deadline has passed, regardless of how many TA there are, and how many bids it has received. Bid processing time is likely to increase with the number of TA as more bids may be received. The computation time increases linearly with the number of bids (at worst, if bids are not sorted, the MA must scan each submitted bid once and store the best one), and therefore scales well.

Increasing the number of MA may affect the computation time of the TA because the TA may receive many tasks from different MA simultaneously. Therefore, the TA may not have enough computation resources to find bids for all the tasks within the specified bidding deadline. In such a case, the TA may need to prioritise, selecting tasks that are most suited to it, or placing strict restrictions on the search for bids, allowing it to find a bid quickly, but at the cost of the quality of the bid. If s is the number of transportation tasks received simultaneously, and Ob is the (non-exponential) time complexity of TA bidding bids (*Eq 83*), then the time complexity increases by $s \cdot Ob$, which is not exponential.

Increasing the number of TA will likely increase memory requirements for the MA because the number of TA that may submit bids increases, and thus more bids for a task may need to be stored. Since the MA only stores one bid per TA per task, unless a TA submits updated better bids, then the extra memory required to store the extra bids for each task is approximately equal to the number of extra TA that have entered the system. Similarly, increasing the number of MA may lead to the number of tasks announced to increase, and thus result in an increase in memory requirements for the TA to store the extra tasks, and any bids that have been submitted for the tasks. Only one bid needs to be submitted and stored for each task, unless the bid is rejected, withdrawn, or an updated better bid is found.

Although time complexity and memory may not be significantly affected by scalability, communication (bandwidth) may be a potential problem. As the number of TA increases, the amount of communication that must be sent and received between the single MA and the many TA increases. Since the communication (of tasks and bids) must be done within a short time frame, there may be bandwidth problems for the MA to send tasks to many TA, or receive bids from many TA, at once, causing a bottleneck. Communication in PAP has been restricted, only allowing TA to submit one bid per task (unless a better bid becomes available), with the MA not rejecting bids that are not used. The maximum number of tasks sent, and bids received, at one time is equal to the maximum number of TA that are available (κ). If the bandwidth is limited, then the MA must spend time waiting for the tasks and bids to be sent and received, respectively, by setting the bidding deadline accordingly. This may increase the time for the MA to find a transportation plan. In a (reasonably greedy) depth-first search approach, a larger bidding deadline may not cause a significant increase in planning time, unless a large amount of backtracking is required (many infeasible solutions encountered). Increasing the number of MA may cause many MA to send tasks to a single TA at one time, again resulting in bandwidth problems, but for the TA.

Increasing the number of TA will likely increase the overall communication requirements of the planning process for the MA. The number of tasks sent and received (see *Eq 87* and *Eq 88*), and bids sent and received (see *Eq 91* and *Eq 92*), is dependent on the number of TA, and the number of branches, or bids, which will likely increase as the number of TA increases. Increasing the number of MA will likely lead to extra communication for the TA, since it is likely to receive more tasks, and thus send more bids to achieve the tasks. Sandholm and Lesser suggest a mutual monitoring and communication charge to restrict message congestion (Sandholm and Lesser 1995). With mutual monitoring, agents are monitored and punished if they submit too many tasks, or submit tasks to an appropriate audience. This could be extended so that agents that send inappropriate bids for tasks are also punished. With communication charge, a fee must be paid in order for the receiving agent to process the message.

7.4.5 Greater Flexibility

Our global transportation scheduling implementation (or framework) is able to address a greater range of transportation problems than some current approaches. Fischer *et al.* (Fischer and Kuhn 1993; Fischer, Muller et al. 1996) addressed a transportation problem that allowed TA to bid a partial quantity of a transportation task. If TA only submit bids with full routes, then our implementation would be able to address Fischer's transportation problem.

Allowing only bids with full quantity and routes, our implementation is able to address the transportation problem specified in chapter 4, Figure 12, which is similar to the (multiple vehicle) Pickup and Delivery problem with time constraints, and thus the Dial-a-Ride Problem and the Vehicle Routing Problem (Savelsbergh and Sol 1995) (see chapter 2). A TA may submit a gap bid (see section 7.3.1) to fully achieve a transportation task, according to the required schedule. The TA may check for piggyback bids in its local plan to allow other transportation tasks to share the same ride, as long as it fully achieves the task. If the sharing of rides are not required (as in the Dial-a-Ride Problem), then only gap bids are used (no piggyback bids). Actions can be inserted into TA schedules so that the TA commences and finishes at a particular location (central depot). Many solutions to these problems are typically centralised or assume a static environment (Savelsbergh and Sol 1995), and therefore are not appropriate for the decentralised and dynamic accounts of this problem domain, which our implementation is well suited to.

Our implementation can address the multi-commodity, multi-modal network flow problem with time windows (Haghani and Oh 1996), which is equivalent to our global transportation problem, but in a decentralised and dynamic environment.

The greyhound scheduling problem (Dean and Greenwald 1992; Dean and Greenwald 1992) is similar to the global transportation scheduling problem where many TA are required to transport a package the complete route. The only difference is that the TA's transport schedules are fixed (when and where they travel is predetermined), similar to a commercial bus, train and airline. Our implementation can address this problem by having actions with an empty inventory in TA's local plan already defined and fixed before planning commences. In this case, only piggyback bids need to be searched for.

7.5 Experimental Evaluation

7.5.1 Military Logistics Scenarios

Over 30 scenarios were executed, which were taken from military logistics training exercises (Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004). Due to the sensitivity of the information contained within the scenarios, details of the scenarios cannot be discussed. We manually determined a plan for each scenario, which is how military planners form schedules in training exercises, and then compared the actual solution produced to that plan. Due to the complexity, the optimal plan could not be computed.

The scenarios comprised up to two MA, ten heterogeneous TA, and an agent to provide MA and TA with distances between locations. TA could travel up to 130 routes, and the transportation tasks were to transport one or two large quantities of resources a large distance. TA had $pth = pcth = 0.5\%$, $\Omega = 80\%$, $rth = 1.5$ and $N = 50$. Only ten TA were run at once because of the limited computing resources available at the time – one PIII 900MHz to run all agents and communication software. The agents were developed using the ATTITUDE multi-agent architecture, and the CoABS grid (Kettler) was used to allow agents to communicate.

In the worst case scenario, there was a single MA with two *large* transportation tasks, requiring a depth in the planning tree of over 60, and four TA, and hence limited available services resulting in considerable backtracking. The bidding deadline duration was set to 2 minutes. The duration was extended to ensure that TA had *more* than enough time to compute bids, with our limited computing resources and relaxed bidding restrictions (large *rth* and *N* so that many/all of TA routes were checked during bidding), before the MA progressed with the planning. A plan was produced in approximately 4 hours. Ideally, each TA will have its own processor and greater bidding restrictions, allowing it to compute bids quickly, and with the bidding deadline duration set to say 5 seconds [35], providing a plan within approximately 10 minutes. Manually calculating a *detailed* complete plan for this scenario (as our implementation did) took us much longer than 10 minutes, while having all the information regarding TA capabilities available. In reality, this may not be the case. These transportation services may need to be extracted from these TA (transport organisations) by the logistics planners via some means, for example, telephone or Internet. This will likely add to the time required to find a transportation schedule, as well as add to the complexity for a planner to manually find a transportation plan.

In the worst case scenario, the MA stored a maximum of 135 tasks and 514 bids, sent a total of 460 tasks to TA (each task sent 4 times, one for each TA), received a total of 479 bids, and communicated (sent and received) 331 speech acts (grants, rejects and withdrawn messages). The MA backtracked 18 times. The worst case TA stored 115 tasks and 113 bids, received 115 tasks and sent 179 bids (the other TA sent approximately the same number of bids as the number of received tasks) and communicated 163 speech acts. Such communication and memory requirements were easily manageable in our implementation.

The transportation plans produced were similar, or the same, to the plans we expected to be produced. There were a few minor differences. The TA occasionally took an unexpected route, primarily because the TA had a gap in its local plan such that it could perform a cheap partial route bid, where the route was a slight deviation from the direct route. In some plans, the TA may take two or more trips to transport packages, rather than one complete trip. As we will discuss in section **Error! Reference source not found.**, this can be reduced by increasing the expected cost in the *cv( )*.

With the scenarios that were executed, our agent-based global transportation scheduling implementation compared well with manual approaches, in terms of time and quality

[35] Running only a single TA with an extensive local plan (many actions), and simulating the MA's task, the TA was able to form a bid for the under 5 seconds with the available computing resource.

(Perugini, Lambert et al. 2004; Perugini, Lambert et al. 2004). Even if a scenario was found in which the agent-based implementation did not perform better than doing it manually (e.g. required the same time and output a similar plan), planners can still benefit by automating this complex and tedious scheduling process – particularly having agents to do the 'running around' for the planner in order to extract and assemble transport services from various transportation organisations.

7.5.2 Computation of bids

Experiments were performed showing the effect on TA computation time by varying the route threshold *rth* and the number of routes *N* that the TA will consider for bids. Scenarios comprised one MA and four TA. Results are shown in Figure 52 and Figure 53, showing the worst case bidding time for any TA in each scenario. As expected, the TA are able to form bids faster with low *rth* and *N* values as less routes need to be considered for bids. We found that setting *rth* or *N* too low sometimes resulted in a bad or infeasible plan for the MA because a vital route required for the plan was not considered for a bid.

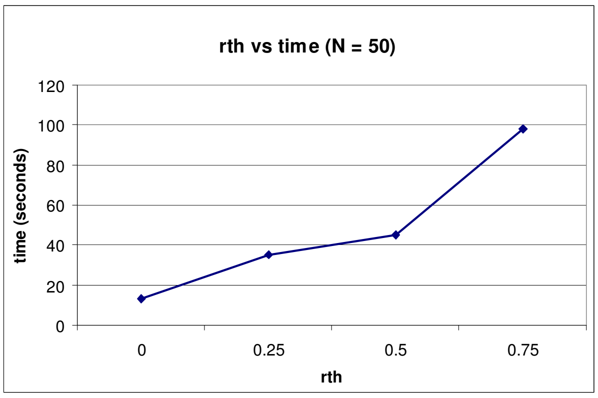


Figure 52. rth versus time.

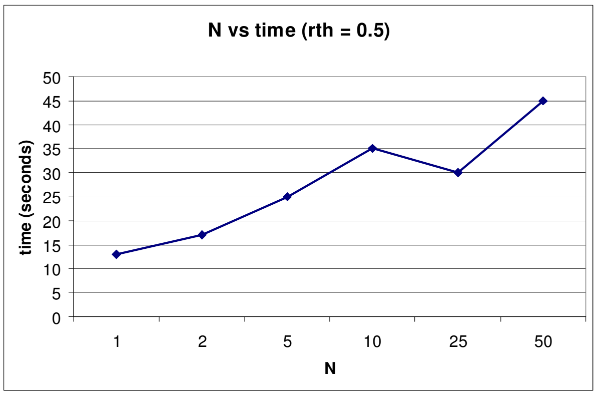


Figure 53. N versus time.

7.5.3 Multiple MA

Experiments were performed using two MA. If a transportation task is split between two MA, then the resulting plan was worse than if one MA achieves the task. Two MA can compete for services in order to achieve their tasks, resulting in the inefficient use of transportation services – tragedy of the commons. Similar behaviour was found when applying PAP to the combinatorial auction domain with multiple auctioneers in the previous chapter.

These experiments also showed that the transportation implementation using PAP was able to cope in a dynamic environment, where services for a MA were continually changing because other MA were also using them. The occurrences of withdrawn bids increased because submitted bids were being used by other MA by the time a MA tried to grant it. MA with a shorter bidding deadline produced a better plan than those with a longer deadline because it allowed the MA to secure received bids before another MA could use it. Having the bidding deadline too short resulted in the MA proceeding with planning before receiving all the bids, as bids take time to compute, usually resulting in a worse plan, or no plan at all (the MA believes that there are no bids for its root tasks).

Note that the abovementioned effects are a result of operating with self-interested MA. Social policies could be introduced to coordinate allied/cooperative MA, such as combining separate MA's tasks into one and then perform the planning.

7.5.4 Comparison with Related Work

In the following two sections, the generality of our transportation scheduling framework, which incorporates PAP, is explored through empirical testing on several related problems from the literature. A survey of the literature has identified two problems that share similarity with our problem: the multi-commodity, multi-modal network flow problem (Haghani and Oh 1996) and the pickup and delivery problem with time windows (Ropke and Pisinger Forthcoming).

We have endeavoured to empirically test the ability of our transportation implementation to solve these problems, as closely as possible. Since the problem we address in this thesis is inherently more dynamic and decentralised in nature than these problems, our comparison mainly serves the purpose of demonstrating the wider applicability of the approach to logistics problems in general.

7.5.5 Disaster Relief: Multi-Commodity, Multi-Modal Network Flow

In Haghani and Oh's multi-commodity, multi-modal network flow problem (Haghani and Oh 1996), they define a class of common problems typical of disaster relief. The model outlines a scenario where disasters could include natural disasters (earthquakes, tsunamis) and terrorism, which are common military problems. In their paper, a multi-modal time (corresponds to distance in our problem sets) and capacity constraints problem is described, where they take a centralised approach to optimisation. They formulate the problem as a mixed integer linear minimisation problem and show how a Lagrangian relaxation approach can be utilised as a powerful heuristic for such problems.

Since disaster relief operations involve cooperation amongst (autonomous) military, government and non-government (commercial) organisations, we chose to test our transportation implementation on this problem since our agent-based approach offers a naturally decentralised and dynamic solution consistent with such open environments. Here, we aim to empirically establish the applicability of our transportation framework for providing a mechanism to coordinate the actions (transportation) of the organisations involved to provide appropriate emergency response. Note that we specifically aim to establish the ability of our (unmodified) agent implementation to solve problems of this kind. Although we used the existing heuristics for our trials, we could easily incorporate new heuristics for this specific problem in order to produce better solutions.

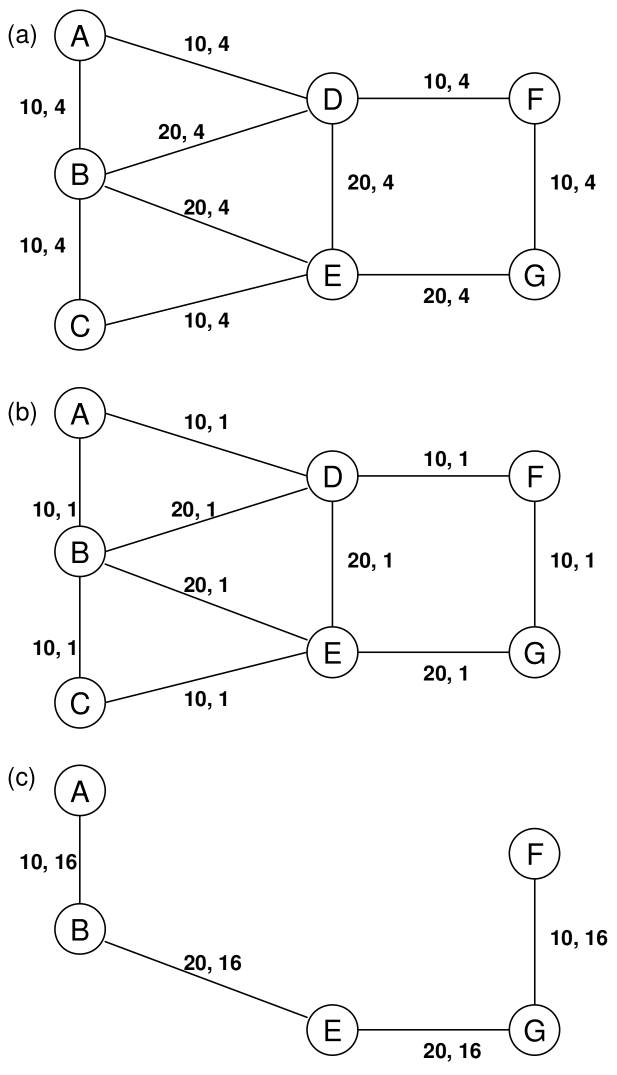


Figure 54. Disaster relief transportation network problem: adapted from (Haghani and Oh 1996), for three modes of transport: (a) mode M, (b) mode L, and (c) mode N. The values pairs on each arc (edge) correspond to distance (time at constant speed) and route capacity constraints.

We adapted the transportation network scenario used by Haghani and Oh, as shown in Figure 54, capturing the same complexities and aspects of the disaster scenarios that they have studied. Haghani and Oh specify the time, rather than distance, for each mode to travel along each route (edge). We specify the distance for each route and use the speed for each mode to determine the travel time. By specifying the route distance, we may use distance, in addition to time of delivery, in evaluating our transportation plans. Haghani and Oh also limit the number of TA that can travel along each route at one time. The route capacity in their scenarios denote the maximum quantity of resources that can be transported along the route simultaneously by a collection of vehicles of a particular mode. We do not restrict the number of TA that can travel along a route, and thus the route capacity in Figure 54 denotes the capacity of each individual TA along the route.

There are three modes of transport: L, M and N. Resources may be transferred between modes at any node. Mode L can travel all routes, has a loading capacity of 4 units and a speed of 10. Mode M can travel all routes, has a loading capacity of 1 unit and a speed of 5. Mode N is unable to get to nodes C and D, has a loading capacity of 16 units and a speed of 2.5. All TA have: loading and unloading times and a price of 1 per unit; the price for transportation is set to the distance travelled to allow the MA to minimise distance and time (equal weighting on price and time was used, $w = 0.5$); and the transfer price was set to 1; $pth = pcth = 0.5\%$; $\Omega = 99\%$; $rth = 1.5$; and $N = 25$. One MA is used to allocate the transportation tasks.

Haghani and Oh plan only 16 time steps into the future and use a large number of TA (hundreds). In our experiments, we were successfully able to solve the scenario in Figure 54 using only between 3-6 TA (plus one MA), but plan much further into the future (up to 398 time steps). Since we assume agents do not have information regarding other agents capabilities and schedules, TAs that schedule actions a long way into the future can submit a partial route bid at the end of the transportation task. They do this assuming that other TAs could service the beginning route of the transportation task since a large amount of time is available. In actual fact, there are no TA that can perform the beginning route in time, as they too are fully allocated in the future. Therefore the MA must backtrack and this potentially results in intractable planning. To prevent this problem, we only allow TAs to perform partial routes that start at the source of the announced task.

# TA Each Mode	# TA	Quantity	Unidirectional or Bidirectional	Total Distance	Delivery Time	# Individual Trips
1	3	10	Unidirectional	1820	190	79
1	3	10	Bidirectional	3160	358	154
2	6	20	Unidirectional	3060	170	146
2	6	20	Bidirectional	6080	398	284

Table 4. Summary of results (disaster relief) using our transportation implementation. Quantity is the quantity of resources that must be transported from each of nodes A, B and C to nodes F and G (for unidirectional and bidirectional) and from nodes F and G to nodes A, B and C (only for bidirectional).

Haghani and Oh simulations transport resources in one direction in the network, from nodes A, B and C to nodes F and G. We also look at transporting resources in two directions, (i.e. also from nodes F and G to nodes A, B, and C), modelling the transport of emergency supplies into the disaster area and the transport of injured persons out of the disaster area.

With our simulations, the quality of plans is simply evaluated based on the total distance travelled by TA (by setting the price for transportation equal to the distance travelled) and the earliest time that all resources are delivered. Results are shown in Table 4.

It is instructive to observe that our implementation has the capacity to generate solutions that reflect bidirectional routing, without further modification, in addition to the unidirectional routing performed by Haghani and Oh, 1996. Although the data used in this scenario is static, in reality it is likely to be dynamic and distributed. This is one of the direct benefits of the more dynamic, and distributed approach used in our framework.

7.5.6 Pickup and Delivery Problem with Time Windows

We applied our transportation implementation *as is* to the pickup and delivery problem with time windows (PDPTW). In the PDPTW, TA must pick up resources within the specified time window (between the earliest pickup time and the latest pickup time) and deliver the resources within the specified time window (between the earliest delivery time and the latest delivery time), and cannot transfer resources between TA (TA must perform the complete route). The TA start and finish at specified locations (depots).

Our transportation implementation does not use time windows for both pickup and delivery, rather it uses an earliest start time for pickup and a latest finish time for delivery. We found that the single time window was appropriate for the military scanarios observed. In our simulations, we used the earliest pickup time in the PDPTW datasets as the earliest start time and the latest delivery time in the PDPTW datasets as the latest finish time. The ability to specify time windows for both pickup and delivery can be easily incorporated into our implementation.

We used datasets by Ropke and Pisinger (Ropke and Pisinger Forthcoming), which were adapted to suit our implementation. These datasets were chosen because they involved complexities that other PDPTW datasets did not, such as: have multiple depots (not all TA start and finish at the same locations); not all tasks need to be served; and vehicles may not be able to serve some transportation tasks. The datasets specify a service time (loading and unloading) for both pick up and delivery. We used a constant service time of 400 (time includes both pickup and delivery service times) since service times in our military scenarios were generally dependent on the type of TA rather than the type of resources transported. For all TA, one unit distance takes one unit time to travel. The price for transportation is set to the distance travelled, and thus the MA is aiming to minimise the distance travelled.

We successfully ran datasets with 15 TA and 50 transportation tasks (allocated via one MA). TA were only allowed to submit full route bids for announced transportation tasks. The results of the simulations are shown in Table 5.

Problem	# Tasks Achieved	# Tasks Not Achieved	# TA Used	Total Distance Travelled
Prob50A	50	0	13	23888.8
Prob50B	49	1	14	29296.1
Prob50C	50	0	15	23010.9
Prob50D	50	0	15	25018.9
Prob50E	48	2	12	17623.8
Prob50F	49	1	14	19663.6
Prob50G	49	1	13	14050.1
Prob50H	49	1	13	17704.5
Prob50I	50	0	13	24022.5
Prob50J	50	0	14	26353.4
Prob50K	50	0	13	19216.6
Prob50L	49	1	13	21883.1

Table 5. Summary of results using our transportation implementation (for PDPTW problem instances adapted from Ropke and Pisinger). TA may not be able to achieve all 50 tasks even though not all TA are used because some TA may not be able to perform some transportation tasks.

The quality of our results are not likely to be as good as those obtained by OR (centralised) approaches to this particular problem. In addition to OR approaches having access to all information while planning, they assume a *cooperative* setting. Hence committed TA plans may be freely altered as transportation tasks are processed. In our implementation, TA's local plans may not be altered after MA have chartered/contracted these services. As a result, other than piggyback bids, TA are unable to amalgamate one or more actions in the local plan with a transportation task in order to form an altered sequence of actions that service both tasks. Since transportation tasks in the datasets do not repeat routes, the TA are unable to take advantage of piggyback bids to transport two or more transportation tasks in one trip. The TA therefore transport only one task per trip, wasting spare capacity.

This particular problem assumes a static environment and does not require decentralisation. Our implementation is appropriate for decentralised and dynamic problems in a non-cooperative setting, where TA actions that are contracted must remain fixed (unless the contract for transportation services specify otherwise). With such problems, the OR approach would not be applicable, whereas our transportation framework can be applied to both types of problems. Finally, better results may have been obtained by devising new heuristics for this specific problem.

7.6 Discussion

In this section, we discuss some issues discovered with our global transportation scheduling implementation. In this chapter, we have provided details of our implementation to address the complex global transportation scheduling problem. In the end, the plans produced by our implementation are only as good as the quality of the bid evaluation function *cv( )* used, i.e. its only as good as the knowledge and heuristics used. Since the implementation of PAP is relatively greedy (greedy with backtracking), the quality and time taken to arrive at a solution relies heavily on the *rationality* and *intelligence* of the *cv( )*, in order for the MA to make smart rational choices for the selection of bids. Deliberative approaches, e.g. A*, require an underestimate for *cv( )* as many paths are tested to provide optimality. Even if deliberative approaches (e.g. anytime algorithms) do not aim for optimality, they generally "search" many paths to find a suitable plan, and thus even a bad *cv( )* can still produce a suitable plan. Greedy approaches rely on intelligent rational decisions as choices cannot usually be altered – in our implementation backtracking is not allowed if a *bad solution* is pursued, only if an infeasible solution is pursued. Additionally, choosing suitable bids will also minimise infeasible solutions, and hence, backtracking and time.

The *cv( )* in our implementation takes into consideration the current cost of the plan (as a function of price and time), and the expected cost to achieve the remaining tasks. If the expected cost returns a low value, then the resulting plan may have the TA performing a route in incremental steps rather than one complete trip, which is less efficient. This is due to the MA selecting partial routes, thinking that a faster and cheaper option is available for the remainder route. When a faster and cheaper option is not found, the TA will again submit another partial bid to perform another step of the route, and so on. Making the expected cost conservative will force the MA to select large complete routes over many partial routes. This is a rational approach, because in most circumstances, one large trip is better than many small trips.

Scenarios with heterogeneous TA with wide ranging services/capabilities (speed and price) resulted in the worst plans as it was difficult to accurately determine the expected cost (speed and time) for the remaining tasks in the *cv( )*. For example, a TA may want to transport a package starting at some location X, but is not sure when the package is able to arrive. If the package arrives at X by plane, the TA can start the transport in 6 hours. If the package arrives by ship, the TA can start in the transport 3 days. What time does the TA send in a bid for the start of the transport? Choosing 6 hours or an average of the two times results in an infeasible solution if the package arrives by ship. Choosing 3 days results in wasted time, and a bad plan, if the package arrives by plane. Therefore, in such circumstances in our scenarios, the *cv( )* associated with the bids may not be accurate, resulting in the MA not selecting the most appropriate bid. Two suggested solutions to this problem are: (1) select the bid (plans or ship) to get the package to X first before selecting TA's bid to transport the package from X; or (2) the MA could assume one (the plane or ship), and if this is incorrect, then use PAP's backtracking facility to select the other option.

Similar to the problem above, and potentially the most debilitating problem with the decentralised nature of our problem (and other similar problems) is the lack of knowledge (partial observability) regarding services, or available bids, by the TA in the environment. This prevents the agents from making good decisions as it impedes on the expected cost of the *cv( )*. Therefore, if a particular course of action is taken, agents may not have a clear idea of how well the TA are able to perform any remaining tasks that still need to be achieved.

In our experiments, we tried to plan with very little information and domain knowledge regarding the environment and the TA services. As a result, decisions were often based on incorrect information regarding expected TA services [36] to achieve remaining tasks. This lead the expected cost of the *cv( )* to be incorrect, and sometimes caused bad decisions to be made – *like making decisions in the dark.*

There are a few potential solutions to the partial observability problem. Available a priori domain knowledge should be used where available. Most application domains, including our global transportation scheduling application, are likely to have constraining factors. For example, tasks may be performed a certain way, e.g. for a particular transportation task a package must always pass through locations X and Y – this reduces the search space of the planning problem. Additionally, information regarding the environment or agents' services may be known, e.g. a particular TA always services a certain route – provides agents with knowledge regarding TA services.

Knowledge regarding the environment and agents' services can also be obtained during planning by agents monitoring the environment or conversing with other agents to extract this information or form (informed) predictions of what services are available. In addition to domain knowledge, agents may be able to make better decisions with little information using good heuristics. For example, in our implementation, if the MA backtracked, the MA would not consider any other bids (for the task at that node) which are similar to the backtracked bid. These bids were likely to also result in backtracking since it creates the same remaining tasks that could not be achieved when the backtracked bid was selected. In some scenarios, this heuristic resulted in a significant saving in time to find a solution. Heuristics may be specific to a particular application domain or generally that perform well for a class of application domains. It will be useful to determine general heuristics, if they exist, and the type of information they require.

In addition to the problem of partial observability, decentralised planning usually has problems with commitment, communication and time. In a centralised system, the MA contains all information regarding TA services. The MA can internally search many options relatively fast, without commitment (holding resources) and communication (and hence time) until the final plan is found and secured. In a decentralised system, the MA

[36] Since this had to be predicted with little/no information – uninformed predictions.

may not be able to search various options quite as easily. The MA must communicate, which takes time, in order to determine the options (bids) it has available. In order to check the suitability of an option, the MA must commit to a bid (provisionally grant a bid) and hold that bid (the TA cannot use the bid elsewhere) before it can determine the suitability of that particular option by progressing with planning – further communication and time by announcing the remaining task(s) and receiving bids for it, and so on. If an option is unsuitable, it requires further communication and time to backtrack, and de-committing from a bid can be disruptive to the TA as it could have used the bid elsewhere. Therefore, there are greater restrictions with a MA's deliberative behaviour using a decentralised approach compared with a centralised approach. Thus, regardless of the complexity of the domain, a strongly deliberative approach to decentralised problems may not always be appropriate – particularly if the cost of commitment, communication and time is high. This also enforces the fact that the *cv( )* should be suitable to select the most appropriate bids in the first instance, eliminating the need for backtracking.

Although decentralised problems may not be suited to deliberative approaches, it may be necessary to deliberate (backtrack) to some degree – to a greater degree than a depth-first search – for certain application domains, including our global transportation scheduling domain. In (Perugini, Lambert et al. 2003) we investigated using PAP to perform an A* search. This could be modified to allow the MA (or agent developer) to adjust the deliberation (from greedy to optimal search, or anytime search) depending on the particular circumstances, such as the complexity of the application domain, time spent planning and the cost of communication.

In our implementation, the *cv( )* used did not always work well for transportation tasks with very large quantities. A few heuristics that we devised for the *cv( )* to solve the problem did not satisfy the restrictive conditions for the *cv( )* (see Definition 18). Additionally, our *cv( )* depends on the time that parts of a package are delivered *multiplied* by the quantity of the parts that are delivered. We would have preferred to have a *cv( )* that depends on the time of the last delivered package, but a *cv( )* for this which met the required conditions could not be easily found (see section 7.3.5). It could be the case that a suitable *cv( )* may not meet the required conditions, and thus the extra communication for repeated tasks may be required. Note that the problem above was resolved if the large task was split into two smaller (but still large) tasks.

Many of these issues are applicable to the general type of planning problems that PAP aims at addressing, such as issues due to decentralisation. Further work is required in order to address these issues.

7.7 Summary

PAP was applied to our complex global transportation scheduling problem, which required partial route, in addition to partial quantity, bids. Details of the implementation, theoretical analysis and experimental results were discussed. Our bid pricing approach allows more informed bid pricing than an approach used by Sandholm (Sandholm 1993). Due to the flexibility of our implementation (or framework), it is able to address a greater range of transportation problems than previous approaches, as they are instances of the global transportation scheduling problem. These problems include Fischer *et al.*'s (partial quantity only) transportation problem, multi-commodity, multi-modal network flow problem with time windows, the greyhound scheduling problem, and instances of the (multi-vehicle) pickup and delivery problem with time windows (PDPTW), dial-a-ride problem and vehicle routing problem. We applied our implementation to both the multi-commodity, multi-modal network flow problem and the PDPTW. Most existing approaches to transportation planning are typically centralised, using Operations Research techniques to address them. We have presented a decentralised approach to transportation scheduling, which is able to cope with a dynamic environment.

Experimental results in our military scenario were encouraging. Our implementation performed well with the scenarios used. The plan produced was the same or similar to forming plans manually, which is how planners currently form transportation plans. With ideal computing resources, our implementation could have potentially reduced the time to form a complete detailed transportation plan. Even when the implementation does not perform a great deal better than forming the plan manually, planners would still benefit from automating this complex, tedious and time consuming process.

We found that breaking a task up among two MA generally resulted in a worse plan than one MA achieving the task because the MA competed for resources (or bids). This resulted in inefficient allocation of resources as one MA obtained a resource that the other required. Similar results were obtained with the combinatorial auction experiments in the previous chapter. PAP was not designed to optimise the global outcome of multiple MA. It was developed to allow a self-interested MA to obtain a suitable plan and allocation for their task, in the presence of other MA that they are potentially in competition with. This is consistent with many organisational interactions in the real world, and could potentially increase the likelihood that organisations would use such a protocol. An organisation may be reluctant to use a protocol which did not act in its own interests, and could potentially benefit its competition. In our implementation, if two or more MA are not in competition with each other (i.e. are collaborative since part of the same organisation), then it is beneficial that they aggregate their tasks and plan as a single MA.

With multiple MA, MA benefit from having shorter bidding deadlines as they are able to secure bids before other MA. Having the bidding deadline too short such that it does not allow enough time for bids to be received resulted in a worse plan or no plan.

We discussed issues regarding our global transportation scheduling implementation, and hence the general type of problems that PAP was developed to address. Due to the predominantly greedy approach of our implementation, the quality of plans produced relies heavily on the bid evaluation function *cv( )*. The *cv( )* should allow the MA to make rational and intelligent decisions for the selection of bids. This enables the MA to make the correct decision the first time as backtracking is only used if an infeasible solution is encountered, not a bad solution. The decentralisation is also an issue with planning, due to partial observability, commitment, communication and time. Partial observability limits the agents in making informed decisions. This is a greater problem when agents (TA) exist with wide ranging services available (e.g. not homogeneous), making it difficult for agents to determine the type of services to expect if they take a particular option. Greater commitment, communication and time are required in decentralised planning than centralised planning. This limits the amount of deliberation (backtracking) that is possible with the decentralised approach if commitment, communication and time costs are high. These issues require further investigation.

Chapter 8

8 Conclusion and Future Work

8.1 Suitability of (BDI) Agents for Modern Military Logistics

In this thesis, we show the ease, and methodology, by which (BDI) agent concepts and technology can model organisations' behaviour. Agents model organisations' business processes and interactions (social processes – protocols), and expertise. The ease of mapping between organisational processes and expertise and agent plans and inference rules, respectively, is due to the similarity between agent ideology and concepts, and those of organisations. Organisations' behaviour is thought of in terms of goals, desires, intentions, success and failure, their information (or beliefs), processes (or plans) by which they achieve their goals, and rules by which they make their inferences and deductions. Agents are also described by these characteristics, where our agents are implemented in ATTITUDE that uses by the BDI paradigm procedurally and Horn clause logic for declarative inference rules. Therefore, agents provide a suitable conceptual model to easily map organisations, their processes and expertise, into software. The examples given in chapter 3 indicate this.

Chapter 3 discussed the ease (and methodology) with which BDI agents can be developed to respond and recover from failure, and respond and react to changes that may occur in the environment. Not only does this provide software with intelligent behaviour to perform tasks in logistics, but increases its robustness as agents are unlikely to crash when they fail or come across a wide variety of (possibly changing) situations. Rather, agents are likely to continue and try alternative means of achieving their tasks, or alter their behaviour when certain situations occur, respectively.

Types of domains where BDI, or procedural, agent approaches are appropriate are in highly constrained domains or domains where there is little time available for planning (e.g. real-time systems). In lowly constrained real-time systems, considerable effort may be required to develop the agents due to the large number of plans required to consider many possible situations. In our logistics domain, most organisational processes are highly constrained, which includes their individual processes and social processes (protocols), making it ideal for BDI agents. Agents' social plans in our domain are not highly constrained as it depends on services available by indeterminate agents at the time. Since agents have a reasonable amount of time to find a plan, a first-principles approach to distributed agent planning is appropriate. Therefore, the way organisations do business, their individual processes and social processes (protocols, such as PAP), are usually relatively constrained, but in our domain, their distributed plan (process) to achieve their business goals are not – e.g. PAP is a constrained social agent process to find an unconstrained social agent distributed plan.

Components of organisations' logistics processes and expertise were successfully modelled and implemented, and thus automated, using (ATTITUDE BDI) agents. An agent modelled and implemented ship's logistics process for forming a medevac logistics plan. Additionally, we modelled and implemented components of a Bde organisation's processes and expertise to (partially) form a logistics plan to deloy a Bde (army) unit. As discussed in chapter 3, the agent development for such a purpose is reasonably easy and straight forward, providing a suitable and *practical* approach in developing (components of) a software support system for the complex military logistics planning domain.

Logistics planning required many calculations, which are tedious, time consuming and error prone. Software, and thus agents, can perform calculations quickly and without error. From results in our implementation in chapter 3, there was a considerable time saving from using agents to perform these calculations. Therefore, performing calculations is one aspect of logistics planning where agents (software) could potentially have a large impact and benefit.

Our current implementation showed that the agents were able to connect and automatically gather information and services from various (geographically) distributed sources, eliminating the need for planners to do the 'running around' and searching for them. The agents were able to use rules (expertise) to collate and analyse the information and plans to produce logistics facts, such as risks with logistics plans, and check constraints, which is vital in logistics planning. Having agents analysing information and plans to provide only required information for planners, and concealing unnecessary details, allows agents to act as information filters. This prevents information overload for planners, potentially making their planning task more manageable and effective.

In DARPA's CoAX project (chapter 3), the medevac agent played a logistics plan execution and monitoring role, in addition to logistics planning. As the medevac agent's logistics plan was executed, the medevac agent monitored the environment (number of casualties). If changes occurred that affected the plan (more casualties detected), then the medevac agent responded (*reacted*) by performing dynamic replanning, changing the relevant components of the logistics plan. This shows that agents are capable of performing a plan execution and monitoring role in military logistics. Additionally, the reactive characteristic (or behaviour) of agents is vital in military logistics because of its dynamic nature, which includes logistics planning and plan execution and monitoring. Agents are able to adapt their logistics plans (dynamic replanning) to changes in the environment. Chapter 3 discusses how this reactive behaviour can be embedded within agents.

There may be extensive expertise, or rules, associated with a particular organisation (such as the Bde army unit), requiring the determination of many logistics facts and checking of many constraints. An agent, if it has the relevant information in its knowledgebase, has the potential to process all of these rules very quickly compared to a human. Planners may have time to investigate only some of the rules in the time given, which was the case in the logistics planning course that was attended, potentially degrading the quality of the plan.

Planners may be forming a logistics plan for an organisation that it knows little about. For example, a planner associated with the *navy* could be planning the logistics to deploy the Bde *army* unit. The navy planner may not be aware of all the Bde organisational details, such as their processes and expertise, in order to form a logistics plan. Additionally, the organisational details may not be documented for others to access. The navy planner must, for example, find and call the relevant person(s) to obtain the required information. Developing an agent for an organisation provides a way of documenting their processes, expertise and interactions, and thus the agent can provide this information to a planner.

This eliminates the need for planners to find and call the correct person in order to obtain the required information. More importantly, agents can use the organisational processes, expertise and interactions to perform the relevant planning *for the planner*, such as forming a logistics plan to deploy a Bde unit. The planner would not require information regarding organisational details. Hence, planners can potentially find and access the services and information (e.g. a logistics plan) from agents (organisations) reasonably easy, without knowing any of their details.

MALT is able to incorporate external organisations in the planning process. The CoAX project demonstrates that it is possible for various international organisations to separately develop agents, connect them via a network, and allow them to cooperate, as long as they follow the same social protocol and ontology. This is important because MALT, and thus the modern military logistics, requires external organisations to be able to develop their own agents (with their private information and to act in their interests) to connect to, and cooperate with, other agents.

An agent social protocol, the Provisional Agreement Protocol (PAP), was developed to allow agents to perform planning and task allocation required by modern military logistics. Agents can use PAP in order to facilitate the cooperation with other agents in MALT, as it provides a means for distributed agents, and hence organisations, to collectively plan together. PAP also considers external organisations' decentralised nature, and does not require them to release services or information that they do not want to, during the planning process.

Agents and PAP provide a suitable approach for military global transportation scheduling. Current transportation scheduling approaches are typically centralised (e.g. Operations Research) and are usually not conducive to a dynamic environment. PAP is able to cope with a changing environment, allowing planning to proceed if changes occur. It has been demonstrated that agents, using PAP and our transportation scheduling framework, could facilitate this complex transportation planning. From our experiments and the scenarios used, there were time benefits with the agent transportation scheduling implementation. Agents were able to improve transportation scheduling, which is a major component in logistics planning. Even when there is little benefit from using agents to automate transportation scheduling (i.e. outputs the same quality plan, in the same amount of time, as a planner that does it manually), planners may still benefit from the automation of this complex and tedious task, and thus can perform more important tasks.

Changes to the logistics system may occur, for example, because organisations may change or add to their processes, expertise, interactions, functions or capabilities. Due to an agent system's (and hence MALT's) inherent modular design, a developer should only need to change or add code (e.g. organisational processes and expertise) in the individual agents concerned, rather than change or add code to various elements in the system. Even in modular designs, particularly if not designed well, a change in one component may affect other components, and hence many components need to be modified. With an agent system, there is less likely to be dependence between modular components (agents) due to its decentralised characteristics – unless there are changes to the social protocols, as these are common between agents. Agents are autonomous, and thus contain their own information and make their own decisions. Additionally, agents are typically developed by different organisations, and thus each are responsible for their own agent's development and modifications. Not only does this distribute development and modification of the agent system to a number of developers, but each developer need only be concerned with a small component of the agent system (one or a few agents), making it easy to keep the complete agent system (MALT) maintained. However, it places more emphasis on social agent specifications.

8.1.1 When are Agents NOT Suitable

Agents are suitable for many aspects of modern military logistics, but it may not be appropriate for all aspects of modern military logistics, or other logistics and related domains. Agents in our application domain were used to model organisations' behaviours. An example for which agents are not suited is the sorting of a list. One does not need to think about solving the problem in terms of goals, beliefs and intentions. If the application domain cannot easily be described in terms of goals, intentions and beliefs, then agents may not be well suited.

Agents may not be suited to application domains that are not decentralised (e.g. distributed, autonomous, self-interested and with private information) and that can be practically implemented centrally. Decentralisation may result in unnecessary communication for planning and agent coordination (particularly if all the information does not need to be communicated from distributed sources). Decentralisation typically results in partial observability, which is likely to reduce the quality of the plan formed as it may be difficult to make informed decisions. Commitment during decentralised planning may cause problems. During planning resources may be held, preventing others from using them, when later it may discovered that they are not required. A centralised system with total observability may exhaustively search all possible options before committing to a plan. Although dividing a task into sub-tasks and distributing them among agents has benefits, such as distributed processing, there can be problems if the agents' behaviours are not coordinated effectively. Negative conflicts and the absence of identifying synergies among agent's behaviours and requirements may result in a worse solution than if it was formed centrally as one large task – due to tragedy of the commons (see chapter 6). This is likely to be of greater concern if a non-cooperative (self-interested) agent approach is taken, as agents are focused on maximising their own local utility rather than the global utility.

8.2 The Provisional Agreement Protocol

The Provisional Agreement Protocol (PAP) enables agents (their associated organisations) to plan and allocate tasks in a decentralised, dynamic and open environment, within a many-to-many setting through contracting. PAP allows organisational (e-commerce) interaction within the social complexities of the modern military logistics environment that arise primarily from its increasing decentralised nature. Agents using PAP are able to form flexible and agile enterprises (or coalitions) to achieve their associated organisations' business goals.

PAP, which is an extension of ECNP, overcomes shortfalls that other contract net approaches (CNP, ECNP and CNP-ext) for our particular planning and task allocation requirements. This includes allowing backtracking, bidders' negotiating with multiple auctioneers (or MA) simultaneously, and planning with dynamic bids and tasks. With backtracking, as well as discarding provisionally rejected bids, PAP enables a decentralised depth-first search, which from our knowledge, has not been previously done. PAP has greater flexibility in planning and task allocation than CNP, ECNP and CNP-ext, while still being able to address problems that they are able to. PAP prevents the eager bidder problem for bidders, allowing them to bid for many tasks for potentially many auctioneers simultaneously. With PAP's ability to withdraw bids and tasks, and with its updated and updated better bid features, agents are able to cope with a dynamic environment, where bids and tasks may emerge and retract during the planning process. PAP is relatively consistent with contracting, as required in our (e-commerce) domain. Allowing auctioneers to deliberate before forming full contracts may reduce broken contracts. We have shown that PAP has less communication than CNP, ECNP and CNP-ext under certain circumstances, and does not fall into livelock (converges to either a solution or no solution) or deadlock. Finally, PAP delegates processing because the auctioneer does little bid processing. Rather, the auctioneer submits a bid evaluation function for the bidders to process their own bids. This can have beneficial scalability properties.

PAP was applied to combinatorial auctions (chapter 6) and our global transportation scheduling (chapter 7) problems. Applying PAP to two domains shows its generality. Combinatorial auctions were primarily used to empirically evaluate PAP. In addition, we were also able to present benefits of PAP over current (centralised) one-shot combinatorial auction approaches, and use it to facilitate the novel dynamic, and multiple (dynamic), combinatorial auction problems. There are a few benefits of PAP over one-shot combinatorial auctions. Using PAP, bidders are not required to submit all their bids and their dependencies to a centralised auctioneer, which may not be practicable to do so. PAP also requires less communication than one-shot auctions if bidders possess many bids. Even though we applied a greedy PAP approach to combinatorial auctions, its ability to interact with a changing environment during the auction (planning process) may allow a better solution to be found than a one-shot auction which determines the optimal solution from an initial submission of bids. Our experiments showed that the saving in communication of PAP over ECNP increases with the number of bidders or depth of the search. Experiments confirmed that PAP performs a depth-first search.

With multiple auctions, PAP performed better locally and globally if resources (bids per auctioneer) were plentiful. As resources became scarce, dynamism and competition, and thus the tragedy of the commons phenomenon, increased. This may result in an inefficient allocation of resources, both locally and globally, leading to a solution which is not pareto optimal nor globally optimal. This issue requires further investigation.

In most scenarios, our PAP implementation used a greedy approach to finding an allocation of bids for the auctioneer, as presented in previous literature. With the single auction (static bids) scenario, we showed that PAP's backtracking facility, with a suitable heuristic, can enable a better solution to be found than the first greedy solution. In the multiple auction case, it was discovered that as resources became scarce, primarily due to dynamism, backtracking was detrimental. Due to the environment dynamics, previous states are not recoverable. Therefore, a potentially better solution may no longer exist when the auctioneer backtracks (or during backtracking), and a solution that was given up may no longer exist when the auctioneer tries to regain it. Implementing a suitable backtracking heuristic may also be difficult due to the presence of partial observability. Agents may not have information regarding whether better solutions are available, whether they will become available, and how to reach a better solution.

With backtracking, we found two problems with the current implementation of PAP. First, our implementation did not allow auctioneers to regain a previously granted allocation of bids as provisionally rejected bids are currently discarded (to ensure convergence). Second, auctioneers may hold on to bids and prevent other auctioneers from using them, and later release them. Auctioneers currently discard provisionally withdrawn bids (to ensure convergence) rather than wait and see if they become available again. Bidders are unable to have conflicting bids from other auctioneers granted until the bid is rejected, which results in a problem similar to the eager bidder problem. This is the subject of future work.

Our global transportation scheduling implementation was able to overcome shortfalls with Fischer *et al.*'s transportation implementation for our domain by allowing partial route, in addition to partial quantity, bids. It is also able to address a greater range of transportation problems [37] than current approaches, which are typically centralised. The various transportation problems are instances of the global transportation scheduling problem. We demonstrated the flexibility of our transportation implementation by applying it to both the multi-commodity, multi-modal network flow problem with time windows and the pickup and delivery problem with time windows. Our implementation addresses these problems in a decentralised manner and can cope with a dynamic environment, and is suitable for variants of these problems with decentralised and dynamic properties.

Our implementation allowed a more informed pricing of bids than previous approaches. We used a bid evaluation function that allowed the auctioneer to carry over unachieved tasks to child nodes in its search tree without having to re-announce them, reducing communication. Experimental results on our military scenarios showed that our implementation performed well with the scenarios used. Plans (transportation schedules) formed were the same or similar to plans that we formed manually, which is how planners currently form them in training exercises. Ideally, the time required for our implementation to form plans should be an improvement over forming them manually. Even if our implementation were not to perform much better than planning manually, planners may still benefit from automating these tedious tasks, freeing their time for other important tasks.

There were a few issues that emerged, mostly as a consequence of the type of domain that we are studying, rather than specific to our transportation application. It is primarily due to decentralisation, and its associated issues of *commitment*, *communication*, *time* and *partial observability*. With centralised planning, all information is usually available. An agent may deliberate by exhaustively searching all options quickly [38] without committing to them, before finally committing to a (potentially optimal) final solution. With

[37] Fischer *et al.*'s (partial quantity only) transportation problem, multi-modal, multi-commodity, multi-modal network flow problem with time windows, the greyhound scheduling problem, and instances of the (multi-vehicle) pickup and delivery problem with time constraints, dial-a-ride problem and vehicle routing problem.

[38] Compared to decentralised – extracting information from memory is usually much faster than over a network.

decentralised planning, greater deliberation requires greater communication. Communication can be costly and introduce a potentially large (compared to centralised) communication time overhead to access the available options, increasing the time required to deliberate. Commitment is also required to hold resources during planning as the options are being examined, and then releasing them if they are not required. This can be a problem, for example (as mentioned above), as holding resources may prevent others from using them if they are later release. Agents in decentralised domains may need to trade off deliberation for intelligent rational decision making. This allows good decisions on possible plan options to be made the first time, reducing the need for deliberation (i.e. backtracking). In order to achieve this, greater reliance must be placed on domain knowledge and heuristics. This is complicated due to partial observability, which increases the chances of making uninformed decisions as capabilities available to achieve goals are not known when decisions are made – decisions are made in the dark. The problem with partial observability worsens as the range of possible capabilities available becomes wider. This makes it difficult to accurately determine capabilities that might be available to achieve agents' goals. Suitable heuristics to deal with issues arising from partial observability requires further investigation. Although extensive deliberation may not be suitable with decentralised planning, greater deliberation than the PAP currently provides may be required – i.e. more deliberation than a depth-first search. This is the subject of future work.

Bibliography

Adali, S. and L. Pigaty (2003). "The DARPA Advanced Logistics Project." Annals of Mathematics and Artificial Intelligence **37**: 409-452.

Ahuja, R. K., T. L. Magnanti, et al. (1993). Network Flows: Theory, Algorithms, and Applications, Printice-Hall.

Airiau, S. and S. Sen (2003). "Strategic Bidding for Multiple Units in Simultaneous and Sequential Auctions." Group Decision and Negotiation **12**: 397-413.

Aknine, S., S. Pinson, et al. (2004). "An Extended Multi-Agent Negotiation Protocol." Autonomous Agents and Multi-Agent Systems **8**(1): 5-45.

Allen, J., J. Hendler, et al. (1990). Readings in Planning, Morgan-Kaufmann.

Allsopp, D. N. and e. al. (2002). "Coalition Agents Experiment: Multiagent Cooperation In International Coalitions." IEEE Intelligent Systems (see also http://www.aiai.ed.ac.uk/project/coax/)(May/June 2002): 26-35.

Andersson, A., M. Tenhunen, et al. (2000). Integer Programming for Combinatorial Auction Winner Determination. Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000).

Anthony, P. and N. R. Jennings (2003). "Developing a Bidding Agent for Multiple Heterogeneous Auctions." ACM Transactions on Internet Technology **3**(3): 185-217.

Arknine, S., S. Pinson, et al. (2004). "An Extended Multi-Agent Negotiation Protocol." Autonomous Agents and Multi-Agent Systems **8**(1): 5-45.

Ausubel, L. M. and P. Cramton (2004). "Auctioning Many Divisible Goods." Journal of the European Economic Association **2**: 480-493.

Babaioff, M. and N. Nisan (2004). "Concurrent Auctions Across the Supply Chain." Journal of AI Research **21**: 595-629.

Bazzan, A. L. C., F. Klugl, et al. (2004). Agents in Traffic and Transportation (ATT 2004). New York, USA.

Becker, M. A. and S. F. Smith (2000). Mixed-Initiative Resource Management: The AMC Barrel Allocator. Proceedsings 5th International Conference on AI Planning and Scheduling, Breckenridge, CO, USA.

Blanchard, B. (1992). Logistics Engineering and Management, Prentice Hall.

Bongaerts, L. (1998). Integration of Scheduling and Control in Holonic Manufacturing Systems (PhD Thesis). Heverlee (Leuven), Belgium, Katholieke Universiteit Leuven.

Bose, R. (1996). "Intelligent Agents Framework for Developing Knowledge Based Decision Support Systems for Collaborative Organisational Processes." Expert Systems With Applications **11**(3): 247-261.

Bouzid, M. (2003). On-line Transportation Scheduling using Spatio-Temporal Reasoning. Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic (TIME-ICTL'03), IEEE.

Bramel, J. and D. Simchi-Levi (1997). The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management. New York, Springer-Verlag.

Bratman, M. E. (1987). Intentions, Plans , and Practical Reason, Harvard University Press: Cambridge, MA.

Bratman, M. E., D. J. Israel, et al. (1988). "Plans and Resource-Bounded Practical Reasoning." Computational Intelligence **4**(4): 349-355.

Brazier, F. M. T., B. M. Dunin-Keplicz, et al. (1997). "DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework." International Journal of Cooperative Information Systems **6**(1): 67-94.

Brinn, M. and T. M. Carrico (2000). "The Language of Dynamic Planning: Inter-Agent Communications in ALP." Fourth International Conference on Autonomous Agents (poster).

Burckert, H., K. Fischer, et al. (2000). "Holonic Transport Scheduling with TELETRUCK." Applied Artificial Intelligence **14**: 697-725.

Busetta, P., N. Howden, et al. (2000). Structuring BDI Agents in Functional Clusters. Proceedings of the 6th International Workshop on Agent Theories, Architectures and Languages, ATAL-99, LNAI Volume 1757, Lecture Notes in Artificial Intelligence, Berlin, Springer.

Bussmann, S. (1998). An Agent-Oriented Architecture for Holonic Manufacturing Control. Proceedings of the first International Workshop on Intelligent Manufacturing Systems, Switzerland.

Byde, A., C. Priest, et al. (2002). Decision Procedure for Multiple Auctions. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy.

Carter, M. W. and C. C. Price (2000). Operations Research: A Practical Introduction, CRC Press.

CDM Technologies, I. (2005). "www.cdmtech.com."

Chaib-Draa, B. and F. Dignum (2002). "Trends in Agent Communication Language." Computational Intelligence **2**(5): 89-101.

Chen, Y., Y. Peng, et al. (1999). A Negotiation-Based Multi-Agent System for Supply Chain Management. Third Conference on Autonomous Agents, Workshop on Agent based Decision-Support for Managing the InternetEnabled Supply-Chain, Seattle.

Cheng, J. and M. P. Wellman (1997). "The WALRAS Algorithm: A Convergent Distributed Implementation of General Equilibrium Outcomes." Computational Economics.

Cheng, S.-F., E. Leung, et al. (2005). "Walverine: A Walrasian Trading Agent." Decision Support Systems **39**: 169-184.

Choi, J., Y. Kim, et al. (2002). "Agent-Based Product-Support Logistics System using XML and RDF." International Journal of Systems Sciences **33**(6): 467-484.

Cicirello, V. A. and S. F. Smith (2001). Ant Colony for Autonomous Decentralised Shop Foor Routing. Fifth International Symposium on Autonomous Decentralised Systems, Dallas, Texas.

Cicirello, V. A. and S. F. Smith (2001). Wasp Nests for Self-Configurable Factories. Proceedings of the 5th International Conference on Autonomous Agents, Quebec, Canada.

Cicirello, V. A. and S. F. Smith (2004). "Wasp-Like Agents for Ditributed Factory Coordination." Autonomous Agents and Multi-Agent Systems **8**: 237-266.

CLIPS. (2005). "CLIPS Home Page, http://www.ghg.net/clips/CLIPS.html."

Cohen, P. R. and H. J. Levesque (1991). "Teamwork." Nous **25**(4): 487-512.

Cohen, S. S. (1985). Operational Research, Edward Arnold.

Collins, J., C. Bilot, et al. (2001). "Decision Processes in Agent-Based Automated Contracting." IEEE Internet Computing: 61-72.

Collins, J., W. Ketter, et al. (2002). "A Multi-Agent Negotiation Testbed for Contracting Tasks with Temporal and Precendence Constraints." International Journal of Electronic Commerce **7**(1): 35-57.

Conen, W. (2002). Economically Coordinated Job Shop Scheduling and Decision Point Bidding - An Example for Economic Coordination in Manufacturing and Logistics. Proceedings of the ECAI 2002 Workshop on Agent Technologies in Logistics, Lyon, France.

Corkill, D. D. (1991). "Blackboard Systems." AI Expert **6**(9): 40-47.

Cougaar (2005). "Cougaar Homepage, http://www.cougaar.net."

Cramton, P., Y. Shoham, et al. (2006). Introduction to Combinatorial Auctions. Combinatorial Auctions (forthcoming), MIT Press.

Currie, K. and A. Tate (1991). "O-PLAN - the Open Planning Architecture." Artificial Intelligence **51**(1): 49-86.

Dang, V. D. and N. R. Jennings (2002). Polynomial algorithms for clearing multi-unit single item and multi-unit combinatorial reverse auctions. Proc. 15th European Conf. on AI (ECAI-2002), Lyon, France.

DARPA (2000). "Advanced Logistics Project Homepage, Defense Advanced Research Projects Agency,." http://www.darpa.mil/iso2/alp.

DARPA(b) (2000). "Ultra*Log Homepage, Defense Advanced Research Projects Agency,." http://www.ultralog.net/.

Davidsson, P., L. Henesey, et al. (2004). Agent-Based Approaches to Transport Logistics. Proceedings of the International Workshop on Agents in Traffic and Transportation (ATT 2004), as part of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, Ney York, USA.

Dean, T. and L. Greenwald (1992). A Formal Description of the Transportation Problem: Technical Report CS-92-14, Department of Computer Science, Brown University.

Dean, T. and L. Greenwald (1992). Package Routing in Transportation Networks with Fixed Vehicle Schedules: Formulation, Complexity Results and Approximation Algorithms: Technical Report CS-92-26, Department of Computer Science, Brown University.

Decker, K. S. (1996). TAEMS: A Framework for Environment Centred Analysis and Design of Coordination Algorithms. Foundations of Distributed Artificial Intelligence. G. M. P. O'Hare and N. R. Jennings, John Wiley and Sons**:** 429-447.

Decker, K. S. and V. Lesser (1995). Designing a Family of Coordination Algorithms. Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA.

Dilts, D. M., N. P. Boyd, et al. (1991). "The evolution of control architectures for automated manufacturing systems." Journal of Manufacturing Systems **10**(1): 79-93.

DoD-US-Doc (2000). Doctrine for Logistics Support of Joint Operations. Joint Publication 4-0.

DoD-US. (2006). "Network Centric Warfare: Department of Defence Report to Congress, www.dod.mil/nii/NCW/." from www.dod.mil/nii/NCW/.

Dong, J.-W. and Y.-J. Li (2003). Agent-Based Design and Organization of Intermodal Freight Transportation Systems. Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi,an.

Duffie, N. A. and R. S. Piper (1987). Nonhierarchical control of a flexible manufacturing cell. Robotics & Computer-Integrated Manufacturing, vol.3, no.2, 1987, pp.175-9. UK.

Durfee, E. H. (1999). Distributed problem solving and planning. Multiagent Systems -- A Modern Approach to Distributed Artificial Intelligence. G. Weiss, MIT Press**:** 121-163.

Durfee, E. H. and V. Lesser (1987). Using Partial Global Plans to Coordinate Distributed Problem Solvers. Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87), Milan, Italy.

Durfee, E. H., V. R. Lesser, et al. (1989). "Trends in cooperative distributed problem solving." IEEE Transactions on Knowledge & Data Engineering **1**(1): 63-83.

Ephrati, E. and J. S. Rosenschein (1993). Multi-Agent Planning as a Dynamic Search for Social Consensus. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France.

Ephrati, E. and J. S. Rosenschein (1994). Divide and conquer in multi-agent planning. Proceeding of the Twelfth National Conference on Artificial Intelligence. MIT Press. Part vol.1, 1994, pp.375-80 vol.1. Cambridge, MA, USA.

Erman, L. D., F. Hayes-Roth, et al. (1980). "The Hearsay-II Speech Understanding System" Integrating Knowledge to Resolve Uncertainty." Computing Surveys **12**(2): 213-253.

Eymann, T. (2001). Markets without Makers - A Framework for Decentraliised Economic Coordination in Multiagent Systems. Lecture Notes in Computer Science 2232 - in Proceedings of the Second International Workshop on Electronic Commerce, Springer-Verlag.

Faltings, B. and M. Yokoo (2005). "Introduction: Special Issue on Distributed Constraint Satisfaction." Artificial Intelligence **161**: 1-5.

Fatima, S. and M. Wooldridge (2001). Adaptive Task and Resource Allocation in Multi-Agent Systems. Proceedings of the 5th International Conference on Autgonomous Agents (AGENTS'01), Quebec, Canada, ACM.

Fikes, R. and N. Nilsson (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." Artificial Intelligence **2**: 189-208.

Finin, T., Y. Labrou, et al. (1995). KQML as an Agent Communication Language. Software Agents. J. M. Bradshaw. Cambridge, MIT Press.

FIPA. (2005). "http://www.fipa.org."

Fischer, K. (1999). "Agent-Based Design of Holonic Manufacturing Systems." Journal of Robotics and Autonomous Systems **27**(1-2): 3-13.

Fischer, K., P. Funk, et al. (2001). Specialised Agent Applications. EASSS01, LNAI 2086:365-382, Springer-Verlag.

Fischer, K. and N. Kuhn (1993). A DAI Approach to Modeling the Transportation Domain, DFKI Report: RR-93-25.

Fischer, K., J. P. Muller, et al. (1996). Intelligent Agents in Virtual Enterprises. Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK.

Fischer, K., J. P. Muller, et al. (1996). "Cooperative transportation scheduling: an application domain for DAI." Applied Artificial Intelligence **10**(1): 1-33.

Foundation for Intelligent Physical Agents. "FIPA Communicative Act Library Specification, http://www.fipa.org/specs/fipa00037/."

Funk, P., G. Vierke, et al. (1999). A Multi-Agent Systems Perspective on Intermodal Transport Chains. Logistik-Management-Tagung LMT-99.

Garcia-Sanchez, F., R. Valencia-Garcia, et al. (2005). "An Integrated Approach for Developing E-commerce Applications." Expert Systems With Applications **28**: 223-235.

Garey, M. R. and D. S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, W.H. Freeman and Company.

Georgeff, M. P. and A. L. Lansky (1986). "Procedural Knowledge." Proceedings of the IEEE Special Issue on Knowledge Representation **74**(10): 13831398.

Graham, I. (1997). "Modeling Business Process with Agents." Object Magazine **7**(9): 21-3.

Greenwald, A. and J. Boyan (2005). "Bidding Algorithms for Simultaneous Auctions: A Case Study." Autonomous Agents and Multi-Agent Systems **10**: 67-89.

Grosz, B. J. (1996). "Collaborative Systems." AI Magazine **17**(2): 67-85.

Grosz, B. J. and S. Kraus (1996). "Collaborative Plans for Complex Group Action." Artificial Intelligence **86**(2): 269-357.

Grosz, B. J. and C. Sidner (1990). Plans for Discourse. Intentions in Communication. P. R. Cohen, M. J. and M. E. Pollack, Bradford Books, MIT Press.

Haghani, A. and S.-C. Oh (1996). "Formulation and Solution of a Multi-Commodity, Multi-Modal Network Flow Model for Disaster Relief Operations." Transportation Research **30**(3): 231-250.

Hameri, A.-P. and A. Paatela (1995). "Multidimensional Simulation as a Tool for Strategic Logistics Planning." Computers in Industry **27**: 273-285.

Hardin, G. (1968). "The Tragedy of the Commons." Science **162**: 1243-1248.

Hatvany, J. (1985). "Intelligence and cooperation in heterarchic manufacturing systems." Robotics & Computer-Integrated Manufacturing **2**(2): 101-4.

He, M., N. R. Jennings, et al. (2003). "On Agent-Mediated Electronic Commerce." IEEE Transactions on Knowledge & Data Engineering **15**(4): 985-1003.

He, M., H. Leung, et al. (2003). "A Fuzzy-Logic Based Bidding Strategy for Autonomous Agents in Continuous Double Auctions." IEEE Transactions on Knowledge & Data Engineering **15**(6): 1345-1363.

Hendler, J., A. Tate, et al. (1990). "AI Planning: Systems and Techniques." AI Magazine **11**(2): 61-77.

Henoch, J. and H. Ulrich (2000). Agent-Based Management Systems in Logistics. Proceedings of the ECAI 2000 Workshop 13.

Henoch, J. and H. Ulrich (2000). HIDES: Towards an Agent-Based Simulator. In Proceedings of Agent Based Simulation: International Workshop 2000, Passau, Germany.

Hildum, D. W., N. M. Sadeh, et al. (1997). Blackboard Agents for Mixed-Initiative Management of Integrated Process-Planning/Production-Scheduling Solution Across the Supply Chain. Proceedings of the ninth Conference on Innovative Applications of Artificial Intelligence.

Hofmann, O., D. Deschner, et al. (1999). Agent-Supported Information Retrieval in the Logistics Chain. Proceedings of the 32nd Hawaii International Conference on Systems Sciences.

Hristozova, M. and L. Sterling (2002). An eXtreme method for developing lightweight ontologies. In Stephen Cranefield, Tim Finin, and Steve Willmott, editors, proceedings of the Workshop on Ontologies in Agent Systems (OAS 2002) held in conjunction with the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03). Republished in CEUR Workshop Series, Bologna, 2002.

Huber, M. (1999). Jam: a BDI-Theoretical Mobile Agent Architecture. Proceedings of the 3rd International Conference on Autonomous Agents (Agents 99), Seattle, WA.

Hunsberger, L. and B. J. Grosz (2000). Combinatorial Auction for Collaborative Planning. Proceedings of the fourth International Conference on MultiAgent Systems (ICMAS-2000), IEEE Computer Society.

Jennings, N. R. (1993). "Specification and Implementation of a Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving." Journal of Intelligent and Cooperative Information Systems **2**(3): 289-318.

Jennings, N. R. (1995). "Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions." Artificial Intelligence **75**(2): 195-240.

Jennings, N. R., P. Faratin, et al. (1996). "Agent-Based Business Process Management." International Journal of Cooperative Information Systems **5**(2&3): 105-130.

Jennings, N. R., E. H. Mamdani, et al. (1996). "Using ARCHON to develop Real-World DAI Applications for Electricity Transportation Management and Particle Acceleration Control." IEEE Expert **11**(6): 60-88.

Jennings, N. R., K. Sycara, et al. (1998). "A Roadmap of Agent Research and Development." Autonomous Agents and Multi-Agent Systems **1**(1): 7-38.

Jess. (2005). "Jess Home Page, http://herzberg.ca.sandia.gov/jess/index.shtml."

Juan, T., A. Pearce, et al. (2002). ROADMAP: Extending the Gaia Methodology for Complex Open Systems. Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy.

Karageorgos, A., N. Mehandjiev, et al. (2003). "Agent-Based Optimisation of Logistics and Production Planning." Engineering Applications of Artificial Intelligence **16**: 335-348.

Kettler, B. P. e. "The CoABS Grid: Technical Vision, http://coabs.globalinfotek.com/."

Kimbrough, S. O., D. J. Wu, et al. (2002). "Computers Play the Beer Game: Can Artificial Agents Manage Supply Chains?" Decision Support Systems **33**: 323-333.

Klusch, M. and O. Shehory (1996). Coalition Formation Among Rational Information Agents. Agents Breaking Away. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '96 Proceedings. Springer-Verlag. 1996, pp.204-17. Berlin, Germany.

Kozan, E. and A. Ohuchi (2002). Operations Research / Management Science at Work, Kluwer's International Series.

Kraus, S., O. Shehory, et al. (2003). Coalition Formation with Uncertain Heterogeneous Information. Proceedings of The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia.

Kress, M. (2002). Operational Logistics: The Art and Science of Sustaining Military Operations, Kluwer Academic Publishers.

Laborie, P. (2003). "Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and New Results." Artificial Intelligence **143**: 151-188.

Lambert, D. (1999). Advisers with Attitude for Situation Awareness. Proceedings of the 1999 Workshop on Defense Applications of Signal Processing, LaSalle, Illinois.

Lambert, D. (1999). Assessing Situations. Proceedings of Information, Decision and Control, IDC'99.

Lambert, D. (1999). Ubiquitous Command and Control. Proceedings of Information, Decision and Control, Adelaide, Australia.

Lambert, D. (2003). Automating Cognitive Routines. Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia.

Lambert, D. (2003). Grand Challenges of Information Fusion. Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia.

Lambert, D. and J. Scholz (2005). A Dialectic for Network Centric Warfare. Proceedings of the 10th International Cammand and Control Research and Technology Symposium (ICCRTS), MacLean, VA.

Langer, G. and A. Bilberg (1997). Architectural Considerations for Holonic Shop Floor Control. Proceedings of the International Symposium on Manufacturing Systems (ISMS) at the World Manufacturing Congress (WMC97), Auckland, New Zealand.

Lesser, V. R. (1999). "Cooperative multiagent systems: a personal view of the state of the art." IEEE Transactions on Knowledge & Data Engineering **11**(1): 133-42.

Leyton-Brown, K., M. Pearson, et al. (2000). Towards a Universal Test Suite for Combinatorial Auction Algorithms. In Proceedings of ACM Conference on Electronic Commerce.

Martinez, M. T., P. Fouletier, et al. (2001). "Virtual Enterprise - Organisation, Evolution and Control." International Journal of Production Economics **74**: 225-238.

Mas-Colell, A., M. D. Whinston, et al. (1995). Microeconomic Theory, Oxford University Press.

McDermott, D. (1996). The current state of AI planning research. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. IEA/AIE 96. Proceedings of the Ninth International Conference. Gordon & Breach. 1996, pp.25-34. Amsterdam, Netherlands.

Montana, D., G. Bidwell, et al. (1999). Scheduling and Route Selection for Military Land Moves Using Genetic Algorithms. 1999 Congress on Evolutionary Computation, Washington DC.

Montana, D., M. Brinn, et al. (1998). Genetic algorithms for complex, real-time scheduling. SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218). IEEE. Part vol.3, 1998, pp.2213-18 vol.3. New York, NY, USA.

Montana, D., J. Herrero, et al. (2000). "A Multi-Agent Society for Military Transportation Scheduling." Journal of Scheduling **3**(4).

Moore, M. L., S. R. T. Kumara, et al. (1996). "An Architecture for Logistics Replanning." Expert Systems With Applications **11**(2): 177-190.

Moynihan, G. P., P. S. Raj, et al. (1995). "Decision Support System for Strategic Logistics Planning." Computers in Industry **26**: 75-84.

Nelson, R. J. (1982). The Logic of Mind. Dordrecht, Holland, Reidel Publishing Company.

Newell, A. and H. A. Simon (1972). Human Problem Solving. Englewood Cliffs, New Jersey, Prentice-Hall.

Nii, H. P. (1989). Blackboard Systems. New York, Addison-Wesley.

Nisan, N. (1999). Algorithms for Selfish Agents. Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99).

Nisan, N. (2000). "Bidding and Allocation in Combinatory Auctions." EC: 1-12.

Nissen, M. (2000). Supply Chain Process and Agent Design for E-Commerce. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Los Alamitos, CA, IEEE Computer Society Press.

Nowak, C. (2003). On ontologies for high-level information fusion. Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia.

Nowak, C. and D. Lambert (2005). The Semantic Challenge for Situation Assessments. Proceedings of the 8th International Conference on Information Fusion, Philadelphia, USA.

Nwana, H. S. (1996). "Software Agents: An Overview." Knowledge Engineering Review **11**(3): 205-244.

Nwana, H. S. and D. T. Ndumu (1996). "An Introduction to Agent Technology." BT Tecnology Journal **14**(4): 55-67.

Nwana, H. S. and M. Wooldridge (1996). "Software Agent Technologies." BT Tecnology Journal **14**(4): 68-78.

O'Leary, D. E., D. Kuokka, et al. (1997). "Artificial Intelligence and Virtual Organisations." Comm. ACM **40**(1): 52-59.

Padgham, L. and M. Winikoff (2002). Prometheus: A Methodology for Developing Intelligent Agents. Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy.

Park, S., E. H. Durfee, et al. (2000). "Emergent Properties of a Market-based Digital Library with Strategic Agents." Autonomous Agents and Multi-Agent Systems **3**: 33-51.

Park, S. and V. Sugumaran (2005). "Designing Multi-Agent Systems: A Framework and Application." Expert Systems With Applications **28**: 259-271.

Parkes, D. C. and L. H. Ungar (2000). Iterative Combinatorial Auctions: Theory and Practice. In Proceedings of the 17th National Conference on Artificial Intelligence.

Parkes, D. C. and L. H. Ungar (2001). An Auction-Based Method for Decentralized Train Scheduling. Pproceedings 5th International Conference on Autonomous Agents (Agents'01).

Parsons, S. and M. Wooldridge (2002). "Game Theory and Decision Theory in Multi-Agent Systems." Autonomous Agents and Multi-Agent Systems **5**: 243-254.

Pasquier, M., C. Quek, et al. (2001). "Opportunistic Planning for a Fleet of Transportation Robots." Engineering Applications of Artificial Intelligence **14**: 329-340.

Peng, Y., T. Finin, et al. (1998). A Multi-Agent System for Enterprise Integration. Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98).

Perugini, D. and I. Fabian (2001). Attitude Tutorial, DSTO Technical Report.

Perugini, D., D. Lambert, et al. (2002). Agents for Military Logistic Planning. Workshop "Agent Technology in Logistics" as part of the 15th European Conference on Artificial Intelligence (ECAI-2002), Lyon, France.

Perugini, D., D. Lambert, et al. (2003). A Distributed Agent Approach to Global Transportation Scheduling. Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), Halifax, Canada.

Perugini, D., D. Lambert, et al. (2003). Distributed Information Fusion Agents. Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia.

Perugini, D., D. Lambert, et al. (2004). Agent-Based Transport Scheduling in Military Logistics. Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04), New York, U.S.A.

Perugini, D., D. Lambert, et al. (2004). Provisional Agreement Protocol for Global Transportation Scheduling. International Workshop Agent in Traffic and Transportation as part of AAMAS04, revised version to appear in Whitestein Agent Technology series, New York, U.S., IEEE.

Perugini, D., D. Lambert, et al. (2005). From Single Static to Multiple Dynamic Combinatorial Auctions. Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2005), Compiegne University of Technology, France.

Perugini, D., S. Wark, et al. (2003). Agents in Logistics - Experiences with the Coalition Agents Experiment Project. Proceedings of the International Workshop "Agents at work", held in conjunction with the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03), Melbourne, Australia.

Petrie, C. (1992). Constrained Decision Revision. Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), Menlo Park, CA, USA.

Priest, C. (2000). Algorithm Design for Agents which Participate in Multiple Simultaneous Auctions. Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems (includes revised papers from AMEC 2000 Workshop), Springer-Verlag**:** 139-154.

Ralston, A. and E. D. Reilly (1993). Encyclopedia Of Computer Science, IEEE Press.

Rao, A. and M. P. Georgeff (1991). Asymmetry Thesis and Side-Effects Problems in Linear Time and Branching Time Intention Logics. Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia.

Rao, A. and M. P. Georgeff (1995). BDI Agents: From Theory to Practice. Technical Note 56. Melbourne, Australia, Australian AI Institute.

Rao, A. and M. P. Georgeff (1995). Formal Models and Decision Procedures for Multi-Agent Systems. Technical Note 61. Melbourne, Australia, Australian AI Institute.

Rao, A. S. and M. P. Georgeff (1991). Modeling Rational Agents within a BDI-Architecture. Proceedings of Knowledge Representation and Reasoning (KR&R-91), San Mateo, CA., Morgan Kaufmann.

Reeves, D. M., M. P. Wellman, et al. (2002). "Automated Negotiation from Declaritive Contract Descriptions." Computational Intelligence **18**: 482-500.

Reeves, D. M., M. P. Wellman, et al. (2005). "Exploring Bidding Strategies for Market-Based Scheduling." Decision Support Systems **39**(1): 67-85.

Refanidis, I., N. Bassiliades, et al. (2001). AI Planning for Transportation Logistics. Proceedings 17th International Logistics Conference.

Ricci, A., A. Omicini, et al. (2002). "Virtual Enterprises and Workflow Management as Agent Coordination Issues." International Journal of Cooperative Information Systems **11**(3&4): 355-379.

Ropke, S. and D. Pisinger (Forthcoming). "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." Transportation Science.

Rosenschein, J. S. and G. Zlotkin (1994). Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, The MIT Press.

Russell, S. and P. Norvig (1995). Artificial Intelligence: A Modern Approach, Prentice-Hall International.

Sadeh, N. M., D. W. Hildum, et al. (1999). MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling. Workshop on Agent-Based Decision Support in Managing Internet-Enabled Supply-Chain, at Agents'99.

Sadeh, N. M., D. W. Hildum, et al. (1998). "A Blackboard Architecture for Integration Process Planning and Production Scheduling." Concurrent Engineering: Research and Applications **6**(2): 88-100.

Sandholm, T. (1993). An implementation of the contract net protocol based on marginal cost calculations. Proceedings of the Eleventh National Conference on Artificial Intelligence. AAAI Press. 1993, pp.256-62. Menlo Park, CA, USA.

Sandholm, T. (1999). Distributed Rational Decision Making. Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence. G. Weiß, MIT Press**:** 201-258.

Sandholm, T. (1999). Distributed Rational Decision Making. Multiagent Systems. G. Weiss. Cambridge, MA, The MIT Press.

Sandholm, T. (2000). "Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation." Autonomous Agents and Multi-Agent Systems **3**: 73-96.

Sandholm, T. (2000). "Issues in Computational Vickrey Auctions." International Journal of Electronic Commerce **4**(3): 107-129.

Sandholm, T. (2002). "Algorithm for optimal winner determination in combinatorial auctions." Artificial Intelligence **135**: 1-54.

Sandholm, T., K. Larson, et al. (1999). "Coalition Structure Generation with Worst Case Guarantees." Artificial Intelligence **111**: 209-238.

Sandholm, T. and V. Lesser (1995). Issues in automated negotiation and electronic commerce: Extending the contract net framework. ICMAS-95 Proceedings. First International Conference on Multi-Agent Systems. AAAI Press. 1995, pp.328-35. Menlo Park, CA, USA.

Sandholm, T. and V. Lesser (2001). "Leveled Commitment Contracts and Strategic Breach." Games and Economic Behaviour **35**: 212-270.

Sandholm, T. and V. Lesser (2002). "Leveled Commitment Contracting: A Backtracking Instrument for Multiagent Systems." AI Magazine **23**(3): 89-100.

Sandholm, T., S. Suri, et al. (2005). "CABOB: A Fast Optimal Algorithm for Combinatorial Auctions." Management Science **51**(3): 374-390.

Sandholm, T. W. and V. R. Lesser (1996). Advantages of a leveled commitment contracting protocol. Proceedings of the Thirteenth National Conference on Artificial Intelligence 1996, pp.126-33 vol.1. Amherst, University of Massachusetts**:** 48.

Sandholm, T. W. and V. R. Lesser (1997). "Coalitions among computationally bounded agents." Artificial Intelligence **94**(1-2): 99-137.

Sandoz, P. A. (1990). Use of Logistics Models as a Part of Command and Control (C2) for Gaming and Simulation Exercises at the National Test Bed. Proceedings of the 1990 Summer Computer Simulation Conference, San Diego, CA, USA.

Satapathy, G., S. R. T. Kumara, et al. (1998). "Distributed Intelligenet Architecture for Logistics (DIAL)." Journal of Expert Systems Applications and Practice **14**: 409-424.

Savelsbergh, M. W. P. and M. Sol (1995). "The General Pickup and Delivery Problem." Transportation Science **29**(1): 17-29.

Schillo, M., C. Kray, et al. (2002). The Eager Bidder Problem: A Fundamantal Problem of DAI and Selected Solutions. 1st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), Bologna, Italy.

Searle, J. R. (1969). Speech Acts. Cambridge, Cambridge University Press.

Sen, S. and E. H. Durfee (1994). "The Role of Commitment in Cooperative Negotiation." International Journal on Intelligent & Cooperative Information Systems **3**(1): 67-81.

Sen, S. and E. H. Durfee (1998). "A Formal Study of Distributed Meeting Scheduling." Group Decision and Negotiation **7**: 265-289.

Shehory, O. (2002). "Optimal Bidding in Multiple Concurrent Auctions." International Journal of Cooperative Information Systems **11**(3 & 4): 315-327.

Shehory, O. and S. Kraus (1996). Formation of Overlapping Coalitions for Precedence-Ordered Task-Execution Among Autonomous Agents. ICMAS-96 Proceedings.

Second International Conference on Multi-Agent Systems, Menlo Park, CA, USA., AAAI Press.

Shehory, O. and S. Kraus (1998). "Methods for Task Allocation via Agent Coalition Formation." Artificial Intelligence **101**(1-2): 165-200.

Shehory, O. and S. Kraus (1999). "Feasible Formation of Coalitions among Autonomous Agents in Non-Super-Additive Environments." Computational Intelligence **15**(3).

Shehory, O., S. Kraus, et al. (1998). Goal Satisfaction in Large-Scale Agent Systems: A Transportation Example, Springer Verlag.

Shehory, O., K. Sycara, et al. (1997). Multi-Agent Coordination Through Coalition Formation. Intelligent Agents IV: Agent Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence No. 1365, Springer**:** 143-154.

Shoham, Y. (1993). "Agent Oriented Programming." Artificial Intelligence **60**(1): 51-92.

Sierra, C. (2004). "Agent-Mediated Electronic Commerce." Autonomous Agents and Multi-Agent Systems **9**: 285-301.

Smith, R. G. (1980). "The contract net protocol: high level communication and control in a distributed problem solver." IEEE Transactions on Computers **C-29**(12): 1104-13.

Swaminathan, J. M., S. F. Smith, et al. (1998). "Modeling Supply Chain Dynamics: A Multiagent Approach." Decision Sciences Journal **29**(3).

Sycara, K., K. S. Decker, et al. (1996). "Distributed Intelligent Agents." IEEE Expert **11**(6): 36-46.

Sycara, K. and D. Zeng (1996). "Coordination of Multiple Intelligent Software Agents." International Journal of Cooperative Information Systems **5**(2&3).

Tambe, M. (1997). "Towards Flexible Teamwork." Journal of AI Research **7**: 83-124.

Tate, A., B. Drabble, et al. (1995). An Engineers Approach to the Application of Knowledge Based Planning and Scheduling Techniques to Logistics - O-Plan Final Technical Report RL-TR-95-235.

Tate, A., B. Drabble, et al. (1994). O-Plan2: an Open Architecture for Command, Planning and Control. Intelligent Scheduling. M. Fox and M. Zweben, Morgan Kaufmann.

Taveter, K. and G. Wagner (2001). Agent-Oriented Enterprise Modeling Based on Business Rules. Proceedings of the 20th International Conference on Conceptual Modeling: Conceptual Modeling.

Tesauro, G. and R. Das (2001). High-Performance Bidding Agents for the Continuous Double Auction. In Proceedings of the 3rd ACM Conference on Electronic Commerce, Tampa, Florida, USA.

Tidhar, G., A. S. Rao, et al. (1996). Guided team selection. ICMAS-96 Proceedings. Second International Conference on Multi-Agent Systems. AAAI Press. 1996, pp.369-76. Menlo Park, CA, USA.

Tidhar, G. and J. S. Rosenschein (1992). A Contract Net with Consultants: An Alternative Architecture and Experimental Results. Proceedings of the European Conference on Artificial Intelligence.

Tidhar, G., M. Selvestrel, et al. (1995). Modelling Teams and Team Tactics in Whole Air Mission Modelling. Proceedings of IEA/AIE.

Timm, I. J., R. Schleiffer, et al. (2002). Agent Technologies in Logistics. Lyon, France.

Tumer, K. and D. Wolpert (2000). Collective Intelligence and Brarss' Paradox. Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence.

Uliera, M., S. S. Walker, et al. (2001). The Holonic Enterprise as a Collaborative Ecosystem. Workshop on Holons: Autonomous and Cooperative Agents for the Industry, 5th International Conference on Autonomous Agents, Montreal, Canada.

UMBC. (2002). "UMBC AgentWeb - http://www.cs.umbc.edu/kqml/." from http://www.cs.umbc.edu/kqml/.

van Hillegersberg, J., H. Moonen, et al. (2004). Agent Technology in Supply Chains and Networks: An Exploration of High Potential Future Applications.

IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04).

Varian, H. R. (1995). Economic Mechanism Design for Computerized Agents. Proceedings of USENIX Workshop on Electronic Commerce, New York.

Vulkan, N. and N. R. Jennings (2000). "Efficient Mechanisms for the Supply of Services in Multi-Agent Environments." Decision Support Systems **28**(1-2): 5--19.

Vytelingum, P., R. K. Dash, et al. (2004). A Risk-Based Bidding Strategy for Continuous Double Auctions. In Proceedings of the European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain.

Walsh, W. E. and M. P. Wellman (2003). "Decentralized Supply Chain Formation: A Market Protocol and Competitive Equilibrium Analysis." Journal of Artificial Intelligence Research **19**: 513-567.

Walsh, W. E., M. P. Wellman, et al. (2000). Combinatorial Auctions for Supply Chain Formation. Second ACM Conference on Electronic Commerce.

Wark, S., A. Zschorn, et al. (2003). Dynamic Agent Systems in the CoAX Binni 2002 Experiment. Proceedings of the 6th International Conference on Information Fusion, Cairns, Australia.

Weiss, G. (1999). Multiagent Systems: A Modern Approach To Distributed Artificial Intelligence, The MIT Press.

Weld, D. S. (1999). "Recent advances in AI planning." AI Magazine **20**(2): 93-122.

Wellman, M. P. (1993). "A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems." Journal of Artificial Intelligence Research **1**: 1-23.

Wellman, M. P. (1996). Market-Oriented Programming: Some Early Lessons. Market-Based Control: A Paradigm for Distributed Resource Allocation. S. Clearwater, Springer-Verlag.

Willmott, S., J. Dale, et al. (2001). Agentcities: A Worldwide Open Agent Network. Agent Link News. **8:** 13-15.

Winograd, T. (1975). Frame Representations And The Declarative/Procedural Controversy. Readings In Knowledge Representation. R. J. Brachman and H. J. Levesque. Los Altos, California, Morgan Kaufmann Publishers Inc.

Wolpert, D., K. Wheeler, et al. (2000). "Collective Intelligence for Control of Distributed Dynamical Systems." Europhysics Letters **49**(6).

Wooldridge, M. (2002). An Introduction to Multi-Agent Systems, John Wiley & Sons.

Wooldridge, M. and N. R. Jennings (1995). "Intelligent Agents: Theory and Practice." The Knowledge Engineering Review **10**(2): 115-152.

Wooldridge, M., N. R. Jennings, et al. (2000). "The Gaia Methodology for Agent-Oriented Analysis and Design." Autonomous Agents and Multi-Agent Systems **3**(3).

Wurman, P. R., W. E. Walsh, et al. (1998). "Flexible Double Auctions for Electronic Commerce: Theory and Implementation." Decision Support Systems **24**: 17-27.

Wurman, P. R. and M. P. Wellman (2000). AkBA: A Progressive, Anonymous-Price Combinatorial Auction. Second ACM Conference on Electronic Commerce, Minneapolis, ACM.

Wurman, P. R., M. P. Wellman, et al. (1998). The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents. Proceedings of the Second International Conference on Autonomous Agents (Agents-98), Minneapolis, MN, USA.

Yokoo, M., E. H. Durfee, et al. (1998). "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms." IEEE Transactions on Knowledge & Data Engineering **10**(5): 673-685.

Zhang, X., V. Lesser, et al. (2005). "Multi-Dimensional, MultiStep Negotiation for Task Allocation in a Cooperative System." Autonomous Agents and Multi-Agent Systems **10**: 5-40.