433-482 Project 2: Research
Resolving Task Ordering using CILP

Wern Li Wong

May 2008

Abstract

In the cooking domain, multiple robotic cook agents act under the direction of a human chef to prepare dinner for a large number of people. The human chef acts as a central facilitator/controller, assigning specific tasks to each agent in order. This paper will explore the possibility of using Collaborative Inductive Logic Programming (CILP to allow the agents to work out amongst themselves, based on local knowledge, what tasks require completion with minimal input from a human controller.

1 Introduction

As described in Project 1, the cooking domain consists of several robotic cook agents carrying out tasks to prepare a meal under the direction of a human chef. Each robotic cook can function as either a retriever, cleaner, chopper or timer. In addition, a waiter agent serves out the food and a dishwasher agent cleans dishes and utensils between uses.

The human acts as a facilitator (by assigning specific tasks to each agent in order) and knowledge base for the system, as each agent has no knowledge of their own beyond how to complete role-specific tasks. Typically, in the process of assigning tasks, the human chef must also perform some sort of path planning and optimization, as each course must be served by a certain time.

In other words, the act of completing tasks is automated, but the agents have no ”initiative” - for example, they would not be able to reason that cooking a particular dish required certain ingredients to be prepared and carry out that preparation task; that would still have to be initiated by the human chef.

Inductive logic is a form of reasoning that works by generalizing theories from a number of specific examples[2]. It can be considered the inverse of deductive logic, which works by drawing conclusions from a set of general statements (premises). To overcome this drawback, this paper will explore the use of collaborative inductive logic programming (CILP) to automate the process and turn the robotic kitchen into a true multi-agent domain.

Ideally, this approach would allow agents to work out what tasks need to be done, and in what order, without further input, and thus remove the need for a human chef to assign specific tasks. It would also not require a central repository of background knowledge - each agent would still only maintain a local knowledge base of how to complete its role-specific tasks - while still keeping communication costs between agents at a minimum.

2 Collaborative Inductive Logic Programming (CILP)

Collaborative inductive logic is an extension of inductive logic programming techniques as applied to a multi-agent domain by tightly integrating processes of induction between agents[1]. It emphasizes a minimalistic approach to knowledge sharing; predicates and hypotheses are shared only when required. It is important because it eliminates the need for a shared knowledge base and central controller while keeping communication costs low, two of the main drawbacks of multi-agent learning systems: it is inefficient to share *all* knowledge on the domain (effectively mirroring the knowledge across all agents), nor is it feasible to communicate all knowledge during the path planning setup.

In the paper, CILP was formally defined as:

> [W]hen provided with some [collective] background knowledge *B* and examples *E* of a concept, an ILP algorithm constructs a hypothesis *H* such that $B \wedge H \models E$.

Essentially, the agents collaboratively generate a hypothesis *H* consistent with their collective background knowledge *B* and *H*, without further external input.

2.1 Described Approach

The paper[1] describes an algorithm for path planning using collaborative inductive logic. The example given is route planning between two points in a network of links - each agent holds some subset of the total knowledge in the domain, not all, in the form of:

```
link(A,B) → reachable(A,B)
reachable(A,B) ∧ reachable(B,C) → reachable(A,C)
```

Capital letters refer to locations or states the particular agent is in at a given time. Each agent would also have a knowledge base of link information in the form `link(a,b), link(a,c)` and so on. There may be some overlap in the contents of knowledge bases of individual agent, but generally what each agent 'knows' can be considered to be distinct.

In order to to solve the problem, the agents collaborate by sharing hypotheses. This allows the agent interested in finding a path to work out which agents know what, without having to communicate the actual data, and only request the data once a path has been found.

3 Application to the Cooking Domain

3.1 Challenges

Adapting this approach to the cooking domain presents some challenges of its own. First, the described approach implies that knowledge is the only thing that is distributed - a single agent is interested in the solution, and once one is found, that same agent is the only one to carry out all the steps. However, in the cooking domain, no one agent can prepare a dish on its own - collaboration is necessary.

Second is the idea of having one agent initiate the induction process. As a single agent is interested in the solution, that agent is the one to pose the "question" to the rest of the agents and collate the answers, but this is not the case in the cooking domain. In a way the human chef played this role in the original system, but no single robotic cook agent is capable of this.

Thirdly, the example given implies that all links from one location to another are of equal distance, which does not hold true in the cooking domain; some tasks take longer to complete than others. Thus the knowledge base and heuristics must be extended to support timing information.

A simple solution to these problems, without heavily modifying either the CILP algorithm or the cooking domain, is to introduce an agent to act as a central controller. This agent will have no background knowledge of its own, but would act as a go-between to facilitate communication between the human chef and robotic agents. The human chef would inform the central controller that a particular dish is to be cooked; the controller would communicate this to all the agents, receive and collate hypotheses, decide on the optimal sequence of tasks and assign the agents to tasks.

Timing data would also have to be included in the knowledge bases and representation of state information - each agent would have to ”know” how long the tasks they can perform take to accomplish.

Each agent would thus function in more or less the same manner, albeit with the addition of CILP programming. As the central controller merely acts as a facilitator or decision-maker (deciding which hypotheses returned by the agents to choose), the multi-agent nature of the domain is preserved as is the inductive logic principles of the computation.

3.2 Adapting the CILP approach

The cooking domain would therefore contain four types of agents: a Central Controller agent, Robotic Cook agents (retriever, cleaner, chopper, timer), a Waiter agent, and a Dishwasher agent
Each of those agents would have the following background knowledge, in the same format as given in the original approach:

```
action(A,B) → doable(A,B)
doable(A,B) ∧ doable(B,C) → doable(A,C)
```

Each individual agent would internally define `action(A,B)` in a manner specific to their role. For example, the Cleaner agent would define the action of cleaning some item as `action(A,B) :- clean(A,B)` where A and B refer to the states before and after the cleaning action was carried out.

For the purposes of optimal path planning, state information would also include timing information. To expand on the example above, the act of cleaning potatoes could be described as `clean(item(potatoes, 0), item(clean_potatoes, 10))` (if the action took 10 units of time).

The knowledge bases of individual agents would reflect this as well. Each agent would ’know’ how long its tasks take to complete.

3.3 Algorithm

The interaction between agents can be summarized as follows:

1. The human chef initiates creation of a specific dish by giving the Central Controller the name of the dish and the required time of completion.

2. The Central Controller delegates out the command to the robotic chef agents in the form of a query.
3. The agents return hypotheses based on their local knowledge.
4. The Controller picks the most promising hypotheses (using the least time to overall completion heuristic) among all returned by agents and re-queries if necessary.
5. Repeat 3 and 4 until a hypothesis indicates path has been found.
6. The Controller instructs the Agents to carry out tasks according to the collaboratively discovered path.

Unlike the approach described in the paper, the central controller agent does not need to request path information from each agent whose hypothesis it chose. Rather, it merely instructs the agent to carry out the task it indicated it has knowledge on how to accomplish. This is because there is no need for the central controller to know *how* a task is carried out, merely that it is done.

3.4 Example Execution

For example, say the problem is to create a dish of `Fried Rice`.

This would involve the raw materials `rice, water` (to cook the rice in), `chicken, sauces, eggs, vegetables, onion`, and `garlic` to be retrieved from storage and measured to specification (`eggs` merely need to be counted). The `rice, chicken` and `vegetables` need to be washed as well as measured. The `chicken` and `vegetables` also need to be diced, the `onion` chopped into small slices/chunks, and `garlic` has to be finely chopped/crushed. Finally, the `rice, chicken` and `vegetables` must be cooked separately before they are combined into a single dish. This knowledge would be distributed across all the robotic cook agents (the dishwasher would also know what utensils are required to carry out each task, but no more than that).

The human chef would instruct the Central Controller to initiate preparation for the dish `Fried Rice` and that it must be done by some specific time.

The central controller has no background knowledge of its own, but it knows that this will mean something to the robotic chef agents. It therefore queries all the agents for path data.

After several iterations of querying, retrieving hypotheses and querying again, it concludes:

1. The Retriever agent should retrieve the `rice, water, chicken, onion, garlic, vegetables, sauces`, and `egg` in that order. This is because the rice needs the water and takes longest to cook; chicken, onion, and garlic are cooked together and take second longest; vegetables take the shortest to cook; and the rest are only required when combining.
2. The Cleaner agent would wash the `rice` first (as it does not need to be chopped), followed by the `chicken` and `vegetables`.
3. The Chopper agent would chop the `onion` and `garlic`, followed by the `chicken` and `vegetables` after the Cleaner agent has cleaned them.

It then falls to the human to cook all the raw ingredients (for the durations given by the Timer agent), divide it into portions and pass them to the waiter agent to serve.

While the robotic chefs carry out their respective tasks, the Dishwasher agent silently supplies each agent with clean utensils and containers appropriate to the tasks they conduct.

This differs from the original setup in that the human does not need to explicitly tell each agent to carry out each task. Rather, the agents collaborate to find the most efficient pipeline (with the least-time-to-overall-completion heuristic) and carry it out, without any more external input from the human chef.

This saves the human the trouble of having to do the optimization problem and then delegate tasks to the agent - the agents will work out amongst themselves what path is best and do it. This is ideal as one problem in most kitchens is the head chef (or whoever does the pipeline) is not necessarily the best at optimizing task ordering.

4 Conclusion

The original intent of introducing CILP-based path planning (in the form of task ordering) in the cooking domain was to eliminate the need for the human chef to explicitly instruct each task it required a particular agent to do, and by extension remove the need for the chef to optimize the order in which tasks were carried out to allow courses to be served on time. Theoretically, this approach would succeed in accomplishing that goal: it allows the agents to optimize task ordering among themselves with minimal input from outside sources.

Ideally, the need for a central controller/facilitator (and thus a single point of failure for the system) would be eliminated entirely, but it does not appear to be possible without heavily modifying the algorithm, which requires a single agent to act as a query initiator and executor. Introducing a separate agent to initiate queries and then delegate the decided-upon tasks to individual agents seems to be a reasonable compromise while still keeping with the spirit of a multi-agent system.

References

[1] Jian Huang and Adrian R. Pearce. Collaborative inductive logic programming for path planning. *International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[2] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*. Springer, 1997.