Diplomarbeit

Route-Choice Strategies for Shared-Ride Trip Planning in Geosensor-Networks

ausgeführt am
Institut für Geoinformation und Kartographie
Technische Universität Wien

und dem
Department of Geomatics
The University of Melbourne

unter der Anleitung von
O.Univ.Prof. DI Dr. Andreas Frank
Dr.-Ing. habil. Stephan Winter

durch
Christian Gaisbauer
Zollergasse 16/6
1070-Wien

Wien, November 2006
Acknowledgements

This thesis was started at the Department of Geomatics at the University of Melbourne during my 5-month stay in Australia. I want to thank all the people that supported me and made my time such a pleasurable experience.

First I want to thank all the people for the great time at the Corporate Research Center for Spatial Information (CRCSI), where I had my workplace. Special thanks go to Sue Hope for helping me out of some serious dead ends. Thanks go also to Matt Duckham and Lars Kulik for their valuable comments. I am particularly grateful to Lars, who took a lot of time for discussing my work and problems. Further thanks go to my colleagues and friends with whom I have worked on the shared-ride trip planning project. It was very exciting to work with Lin Jie Guan and Yun Hui Wu, giving me the chance to get in touch with China while spending my time in Australia. Furthermore I want to thank ‘Uncle’ Carlos Vieira for all the good times we had, and for sharing his valuable experience.

I want to express my gratitude to my thesis supervisors. I want to thank Prof. Andrew Frank in particular for supporting me during the preparation phase of my stay in Australia, which helped me so much. Special thanks go to my supervisor in Australia, Dr. Stephan Winter. I want to thank him for giving me the chance to work on this exciting project, and for making it such a great experience, both academically and personally.
Finally I get to thank my parents for their love and support through all my life. Thanks go also to my sisters Eva and Ilse for always being there for me. To Ilse I’m also thankful for correcting grammar and style of this thesis.
für meine Familie
Abstract

Shared-ride trip planning is one potential field of application within the research domain of geosensor-networks. Such a system is envisioned to provide people in need of transportation (clients) and transportation providers (hosts) with an ad-hoc, peer-to-peer communication, positioning, and planning infrastructure within an urban environment. By doing this, shared-ride trip planning can contribute to increasing the efficiency of contemporary transportation networks. Hence shared-ride trip planning can help reduce traffic load and environmental stress.

Route choice plays an important role within such a system, as it is highly dynamic in terms of transportation and communication network topology. As a result, clients might not be able to get a ride from start to destination right away. This leaves the client with a risky decision: which of all currently reachable locations is the most promising one for the client’s ongoing trip?

This thesis deals with the problem stated above. Using heuristics, a strategy is proposed in order to enable free route choice and limit trip duration. The derived strategy is implemented within an agent-based computer simulation and the results are analyzed and compared to previous work.
Zusammenfassung


## Contents

1 Introduction ........................................... 1  
   1.1 Motivation ........................................... 1  
   1.2 Goal and Hypothesis ................................. 4  
   1.3 Methodology ......................................... 5  
   1.4 Contribution ......................................... 5  
   1.5 Organization of this Thesis ......................... 6  

2 Literature Review ..................................... 7  
   2.1 Related Projects in Ride Sharing .................... 7  
      2.1.1 Operating Car Sharing Services ................. 7  
      2.1.2 Related Research and Industry Projects ........... 8  
   2.2 Wireless Sensor Networks ........................... 9  
      2.2.1 Geosensor Networks .............................. 10  
      2.2.2 Mobile Ad-Hoc Geosensor Networks ............... 11  
      2.2.3 Communication Strategies in Geosensor Networks .. 12  
   2.3 Shared-Ride Trip Planning ......................... 13  
      2.3.1 Concepts ........................................ 14  
      2.3.2 Route Choice Strategy ............................ 14  
      2.3.3 Mobility Model for Simulation ................... 15  
   2.4 Network Algorithms and Graphs ..................... 18  

VII
5 Evaluation

5.1 Evaluation Methodology ........................................ 56
  5.1.1 Dispersion of Trip Durations .......................... 57
  5.1.2 Methodology for the HLR-Strategy .................. 57
  5.1.3 Methodology for the HLR\textsubscript{P}-Strategy ...... 57

5.2 Simulation Parameters ............................................ 58

5.3 Results and Discussion .......................................... 59
  5.3.1 Performance of the HLR-Strategy .................... 59
  5.3.2 Performance of the HLR\textsubscript{P}-Strategy .......... 62

6 Conclusion and Outlook ............................................. 68

6.1 Summary ........................................................... 68

6.2 Conclusion with Regard to the Hypothesis .................. 69

6.3 Outlook ............................................................. 70
  6.3.1 Further Modeling and Evaluation ..................... 70
  6.3.2 Trip Quality and Communication ..................... 71
  6.3.3 Competition .................................................. 71
  6.3.4 Feasibility and Data Structures ....................... 72

Bibliography ............................................................ 72

List of Figures .......................................................... 78

List of Tables ........................................................... 80
Chapter 1

Introduction

The first chapter introduces the problem statement and presents the research hypothesis. Furthermore, the methodology for proving the hypothesis, as well as the scientific relevance of this research are discussed. The chapter concludes with an overview of the organization of the thesis.

1.1 Motivation

Intelligent transportation systems, or ITS, have long been in the focus of research in the USA, Europe, and Japan. Its goals are to provide technologies able to improve traffic efficiency and driver safety within transportation networks. Communication systems play an important role in this context, as they are not prone to the limitations of autonomous systems (radars, cameras, and other passive sensors). The joint development of communication- and positioning technologies promises to provide new and innovative approaches for current and future transportation issues, such as pollution, traffic congestion, or inefficient capacity usage [1] (Figure 1.1).

Wireless sensor networks have recently become more prominent, as they
seem to have the potential of revolutionizing various segments of our lives. Among these, there is environmental monitoring, manufacturing, health-care, business asset management and transportation engineering [40]. In the near future, cars are expected to communicate via short-distance radio communication that will allow new applications within intelligent transportation systems\(^1\).

Geosensor networks have recently been introduced as a special type of mobile wireless sensor network, designed to gather all types of geospatial data within the region of deployment. Equipped with positioning- and other sensing devices, geosensor nodes are able to autonomously observe the area or event of interest. Ad-hoc mobile geosensor networks are consisted of mobile sensor nodes that frequently enter and leave the area of an event\(^2\). As a result, the topology of the communication network is subject to constant change, and therefore quite complex [21].

*Shared-ride trip planning* is a new paradigm within the technical envi-

\(^1\)http://www.spiegel.de/auto/werkstatt/0,1518,267313,00.html

\(^2\)An event may be anything that can be sensed by nodes in the geosensor network.
CHAPTER 1. INTRODUCTION

...
1.2 Goal and Hypothesis

The goal of this thesis is to provide a solution to the problem of intermediate location choice within shared-ride trip planning, as stated above. In order to achieve this, we develop a new route-choice strategy and algorithms for location choice. To capture the dynamics of a shared-ride trip planning system and provide effective trip planning, we use methods from artificial intelligence, called heuristics [24, 28]. We thus formulate the research hypothesis in the following way:

1. Heuristic route-choice strategies can provide a solution to the problem of intermediate location choice, thus enabling free route choice compared to the strategy employed before: the pattern matching strategy. [32, 34].

2. Heuristic route-choice strategies achieve shorter trip durations on average, compared to the route-choice technique currently employed in
CHAPTER 1. INTRODUCTION

shared-ride trip planning: the pattern matching strategy.

1.3 Methodology

Solutions to the problem of intermediate location choice are developed and presented in this thesis. The developed theory is implemented and tested in an agent-based computer simulation environment. This simulation environment will be used for testing if the heuristic strategy presented here can in fact provide a solution to the problem of intermediate location choice. Also, average trip duration will be investigated and compared to the pattern matching strategy.

Communication, network topology, and agent behavior are implemented in a simplified manner. All limitations however apply to the pattern matching strategy as well, hence the simulation environment provides enough basis for proving the hypothesis.

1.4 Contribution

Heuristic choice of intermediate trip locations is an important research question in shared-ride trip planning [33, 35]. A heuristic evaluation of reachable locations would enable free route choice, greater flexibility, and increased robustness within the system. Average travel times are expected to substantially decrease compared to the technique employed so far. Hence, proving the hypothesis is an important research-step towards the implementation of such a service in the future.
CHAPTER 1. INTRODUCTION

1.5 Organization of this Thesis

The next chapter gives an overview on research conducted in related areas, as well as in shared-ride trip planning itself. Furthermore, tools and theories are outlined that are relevant for this thesis. Chapter 3 contains the strategies and algorithms developed in the process of proving the hypothesis. Chapter 4 describes the simulation environment, while Chapter 5 gives an analysis of the simulation results. Chapter 6 completes the thesis with the conclusion and an outlook on future research.
Chapter 2

Literature Review

This chapter gives an overview on material relevant to this thesis. It discusses related projects in industry and research, gives background information on geosensor networks and shared-ride trip planning, and presents tools necessary for proving the hypothesis.

2.1 Related Projects in Ride Sharing

There are services that are already in operation which achieved pioneer work in the field of ride sharing and showed that shared-ride services (also called car sharing services) work. In this section we briefly discuss some of these existing services as well as research currently being conducted.

2.1.1 Operating Car Sharing Services

Numerous shared-ride systems are already in use, such as LiftShare, RideNow, SuperShuttle or Mitfahrzentrale. Mitfahrzentrale already has more than 700000 registered users, making it the largest shared-ride organization in Europe. These systems connect private hosts and clients. On the other
CHAPTER 2. LITERATURE REVIEW

hand, systems like Google Transit Trip Planner focus on public means of transportation.

All these shared-ride systems are steadily growing in acceptance and participation. Despite this, these systems are inflexible as they require centralized and (mostly) web-based planning in advance. In contrast, shared-ride trip planning is envisioned to enable peer-to-peer client-host communication and planning in real time, providing high levels of flexibility to all participants.

2.1.2 Related Research and Industry Projects

It is predictable that wireless sensor networks within transportation will become increasingly important and ubiquitous at some point [11]. One contemporary contributor in this area of research is the Car2Car Consortium\(^1\). Founded by six major European car manufacturers, its aim is to create industry standards for future inter-vehicle communication. Expected benefits are traffic congestion warnings, accident prevention and many more. The already finished CarTALK 2000 project partly had these goals for application. Its main aim, however, was the development of a communication protocol for inter-vehicle communication [25].

The RoboCup Rescue project is situated in the area of disaster management, which emerged out of the RoboCup Soccer project. It aims at supporting disaster management teams in large-scale emergencies under rapidly changing conditions. Autonomous robots communicate via wireless technology, and are supposed to considerably increase rescue performance and safety. The current status of the project is the development of a realistic disaster simulation environment, as this kind of simulation environment is not yet

\(^1\)www.car-2-car.org
available [16].

In [36, 37, 39] the authors develop a service where specially equipped cars sense *spatio-temporal resources*, i.e., parking slots. The closer and the younger the resource is to a participant, the higher its potential benefit. Cars communicate the sensed resources through integrated local-area wireless protocols among each other in an opportunistic manner. In this context, opportunistic resource exchange means that cars relay the transmissions of others under the premise that their own transmissions are relayed as well. The received information about spatio-temporal resources is ranked for their potential interest to the driver, with more relevant parking slots kept in the memory while others are being purged.

This overview shows that wireless communication and positioning technologies are becoming increasingly incorporated and accepted in related fields. The work from [36, 37, 39] is of particular interest and shows various parallels—in terms of application area, system requirements, and approach—to the problem investigated in this thesis.

### 2.2 Wireless Sensor Networks

Rapid advances in sensor technology and miniaturization are opening new ways of gathering data about our environment. It is now becoming increasingly feasible to deploy large numbers of low-cost, low-powered platforms with on-board sensors capable of short-range wireless communication.

The range of applications encompasses a variety of commercial and military applications. In the future, tiny and cheap sensors may actually be sprayed onto roads, walls etc. forming a *digital skin* that senses different physical phenomena of interest. These sensed phenomena might be traffic
data, wildlife habitats or forest fires and the likes [40]. Last but not least, geosensor networks—as a type of wireless sensor network—form the technological framework of shared-ride trip planning.

### 2.2.1 Geosensor Networks

Geosensor networks [21] is an emerging field that applies these latest developments in wireless sensor network technology to the geospatial domain.

Nodes in geosensor networks are static or mobile (Section 2.2.2) platforms that combine sensors such as GPS, cameras, and other sensing devices with wireless communication capabilities. Research aims at developing infrastructures consisting of large numbers of nodes that work unattendedly, untethered and collaboratively, with non-renewable power-supply. Since the sensor nodes are envisioned to be tiny, the providable energy imposes limitations onto communication capabilities. As a result, sensor nodes need to communicate with peers within their spatial proximity [20].

Nodes within geosensor networks collect, aggregate, analyze, and monitor geospatial data. Aggregation and analysis can be carried out either locally in real-time, or offline in centralized repositories, i.e. more powerful nodes with comparably large processing, storage, communication and energy capabilities.

In [21] a geosensor network is defined as such that it "...monitors phenomena in geographic space". Geographic space varies in scale, it can range from the confined environment of a room to whole ecosystem regions. Also, the number of sensor nodes may vary considerably due to the scalability of the decentralized architecture.

Creating a working computational infrastructure that delivers useful information from raw data, gathered by independent sensor nodes on a large
scale, is a challenging endeavor. The final goal for development is a generic, reusable sensor network that is applicable to a wide variety of applications.

In shared-ride trip planning, however, mobility is a crucial part. Hence, shared-ride trip planning relies on mobile ad-hoc geosensor networks.

2.2.2 Mobile Ad-Hoc Geosensor Networks

Mobile ad-hoc geosensor networks can be seen as a special type of MANET (mobile ad-hoc networks), i.e., a self-configuring, wireless network of mobile nodes. The geosensor networks community is also using the term MAGNET, for mobile ad-hoc geosensor network. MAGNETs capture data relevant in a geographic context, in immediate proximity of an event. Geographic position—and therefore network communication topology—changes continuously, making MAGNETs a particularly challenging field of research. This location-dependency contrasts with data storage in conventional networks, where the storage location may be completely independent of the location to which the data refers to [20].

Sensor nodes frequently leave the neighborhood of an event. Hence, efficient sharing of information is an important research question which has already begun to be answered in [36, 37, 39] for MANETs—[20] extends this work on MAGNETs.

Preserving battery life and communication bandwidth are crucial to prolong system lifetime once a sensor network is deployed. Battery packs are in general not replaceable, and communication bandwidth is a scarce resource as well. Developing efficient communication protocols for information routing allows to not overly stress bandwidth. It is also a promising way of achieving longer battery lifetimes, since communication is a significantly higher drain on energy compared to processing information locally on a sensor node.
CHAPTER 2. LITERATURE REVIEW

Conventional communication networks typically use routing protocols that are address-based (IP-addresses), whereas sensor networks employ data-centric routing strategies. A data-centric routing protocol uses properties of the data that are external to the communication network, such as the physical location of nodes [40]. One of the advantages in data-centric routing is scalability. It enables large, decentralized, peer-to-peer information dissemination that has the potential to overcome traditional bottlenecks of centralized service architectures and databases.

Shared-ride trip planning is situated within the area of mobile ad-hoc geosensor networks. Neighborhood is constantly changing, and efficient communication and bandwidth usage are also important challenges for shared-ride trip planning. Energy consumption is relevant for clients and their handheld devices. This is not the case for hosts, since vehicles in general provide sufficient energy for all communication tasks.

2.2.3 Communication Strategies in Geosensor Networks

Communication in mobile ad-hoc sensor networks is generally radio-based [40]. Messages are broadcasted and received by nodes within radio range of the transmitter. Communication does not happen instantly, there is always latency, or delay, involved in the process. Forwarding messages has to be done by using multi-hop communication, if a message is to be sent deeply into the network. Multi-hop communication is a technique were sensor nodes, who receive a transmission from another sensor node, forward these transmissions to the other nodes within their vicinity.

Communication is the most energy-consuming task in geosensor networks. For reasons of energy efficiency, communication is performed in short and synchronized communication windows. For shared-ride services, these windows
must be appropriate to realize requests, offers and bookings. Additionally, time geography has recently been proposed as a technique for reducing the amount of broadcasted messages, and therefore also reducing energy drain in the network [35].

Several multi-hop communication protocols are proposed [20, 34] and evaluated [5] in the context of geosensor networks:

- **Flooding:** agents always inform all other agents within communication range about the travel needs of clients.

- **Epidemic:** clients only inform the first $n$ hosts they encounter within their communication range about their travel needs.

- **Location-constrained:** agents inform other peers within communication range only as long as these agents are within a certain threshold distance from the destination.

### 2.3 Shared-Ride Trip Planning

Shared-ride trip planning is a service designed to operate within the technological framework of MAGNETs, recently proposed in [32, 33, 34]. It aims at providing an ad-hoc, peer-to-peer [27] communication infrastructure that enables ride sharing within an urban environment.

A communication network consists of *clients*—in need of transportation—and *hosts* which offer rides along their predefined routes. All nodes in the network have in common that they are mobile, have communication capabilities and are location-aware. Clients try to reach their destination through sharing rides with hosts. For that purpose, clients communicate their destination to hosts. Contacted hosts—after having decided whether they can
CHAPTER 2. LITERATURE REVIEW

Contribute to the client’s trip or not—will broadcast their route to the client, including time schedule. The client then is able to calculate the optimal trip to the destination, according to criteria such as travel time, costs, transfer stops, and so forth. After the trip has been composed, the client books the necessary hosts and starts traveling.

2.3.1 Concepts

Three related components have been identified in [32] that make up the envisioned shared-ride system. These components are:

1. Shared-ride trip planning: this component consists of identifying and formulating the client’s transportation needs, searching for trip opportunities, and booking the requested hosts.

2. Shared-ride trip revision: trip revision is required since the planning environment is dynamic, and travel opportunities can newly arise or get canceled frequently. I.e., clients search for new travel opportunities that provide a better trip, book hosts and cancel obsolete bookings. This part is in essence the regular repetition of component (1).

3. Shared-ride trip traveling: this component finally stands for the physical act of traveling conducted by the clients.

2.3.2 Route Choice Strategy

So far, route choice in shared-ride trip planning has not been in the focus of research. A temporary substitute—the pattern matching strategy—was introduced in [34]. Using the pattern matching strategy, the client is only allowed to follow a predefined route, i.e., the shortest path from the current location of the client to the desired destination.
CHAPTER 2. LITERATURE REVIEW

In a system where clients follow a predefined route, hosts can effectively decide whether or not they can contribute to the client’s trip using the pattern matching strategy. Clients broadcast their chosen (and constant) route to hosts, which apply pattern matching with their own route. This means that hosts compare if their route and the client’s route overlap. If they do, the host sends an offer with the part of the client’s route where the host can provide a ride, given that the host still has free transportation capacities.

This approach was sufficient to evaluate the different communication strategies mentioned in Section 2.2.3; evaluation and results are to be found in [5]. Though sufficient for previous research, the actual trips conducted by clients are inflexible, and prone to producing longer travel times on average. This can happen if street segments\(^2\) of the predefined path are experiencing a low frequency of passing hosts.

2.3.3 Mobility Model for Simulation

Agent-based computer simulations of a shared-ride trip planning system demand a mobility model that simulates the movement of motorized nodes. Two classes of mobility models can be distinguished, namely traces and synthetic models. Traces are mobility patterns that can be observed in real life systems, especially if they involve large numbers of participants and comparably long observation times. Traces are difficult to model for ad-hoc networks, therefore the predominant types of mobility models are synthetic ones [4]. One completed agent-based computer simulation in shared-ride trip planning is described in [5], based on a random walk mobility model.

\(^2\)A street segment is considered to be the part of a street that is connecting two street intersections.
CHAPTER 2. LITERATURE REVIEW

Figure 2.1: Traveling pattern of an agent using the random walk mobility model (from [4]).

Random Walk Mobility Model

So far, the route that computer-simulated hosts chose in [5] was merely based on a type of random walk mobility model (RWMM), sometimes referred to as brownian motion. RWMMs are widely used to mimic movements through arbitrarily choosing speed and direction from predefined ranges, \([speed_{min}, speed_{max}]\) and \([0, 2\pi]\), respectively. Movements happen either within a constant time interval \(t\) or after having traveled distance \(d\), after which new speed and direction are calculated. If an agent hits the boundary of the simulation area, the agent bounces off with an angle dependent on the incoming direction. Several derivatives exist, supporting all types of dimension (1-D to d-D). Models of type 2-D are of particular interest, since they are suitable for modeling the surface of the earth.

In general, the RWMM is a memoryless mobility pattern, no information
of past movements is considered for route choice decisions. This can result in unrealistic behavior, especially when agents are supposed to act within transportation networks.

Consider that agents in such a model go from one street intersection to another one along a street segment. When they arrive at the new intersection, they have the same probability of going back to the street intersection they came from, i.e., taking a u-turn, compared to going along another street segment. In extreme cases this can cause an agent to oscillate back and forth between two nodes, resulting in unrealistic motion of an agent. Furthermore, the initial starting position of an agent is heavily influencing future movements, as the agent tends to stay in the area around its starting position (Figure 2.1).

RWMM is also used for the simulation environment of this thesis, with some modifications (Section 4.2.1). In general, transportation networks are not of type 2-D, e.g. underpasses that run under another street occur in a real-world transportation network. However for our simulation purposes a 2-D model is sufficient. Also, a RWMM of hosts is simplified, however this does affect the pattern matching strategy to the same extend as any other newly developed route choice strategy. Oscillation around the initial position has an effect on the number of transfer stops, since the region of activity for a host is limited. In this thesis we only focus on average trip duration and not on the number of transfer stops, hence basic conclusions about the effectiveness of a developed route choice strategy can be drawn.

Mobility in Shared-Ride Trip Planning Simulations

The mobility model used in [5] is, in essence, based on a RWMM as described above. Simulations take place in a 10x10 grid world (Figure 2.2). This means
that each street intersection—that is not part of the border of the simulation environment—has four connecting street segments to intersections in all four cardinal directions. The lifetime of hosts can be adjusted as necessary. Hosts are able to travel a constant one street segment per time unit or wait at the current intersection, the travel direction is random and all four directions, as well as the option of waiting, have equal probability of being chosen. The deployment and withdrawal of hosts is done in a way so that there is always the predefined number of hosts active within the simulation environment.

2.4 Network Algorithms and Graphs

In the following we first introduce basic graph- and complexity theories. The section concludes with a selection of relevant algorithms.
CHAPTER 2. LITERATURE REVIEW

2.4.1 Basic Graph Theory

Graph theory is part of the broader field of combinatorics, which has experienced a particularly fast development in recent years. It forms the mathematical basis of areas such as network analysis, computer science and transportation engineering. Graph theory is the fundamental tool to abstract street- and transportation networks to models that can be computationally treated [15].

A graph, or undirected graph $G$ is a pair $G = (V, E)$ consisting of a finite\(^3\) set of elements called vertices—with $V \neq \emptyset$. Set $E$ consists of unordered, two-element subsets of $V$. These subsets $e = (u, v)$ are called edges with end vertices $u$ and $v$. We say that $u$ and $v$ are incident with $e$ as well as adjacent to each other. A graph where edges consist of ordered pairs of vertices is called directed graph. A directed edge $e = (u, v)$ is considered to be directed from $u$, the tail, to $v$, the head.

The literature on graph theory generally lacks consistency [15]. An alternative notation that is frequently occurring is to describe the above defined graph $G$ as $G = (N, A)$. In this notation, set $N$ is the set of nodes that was formerly defined as vertices. Set $A$ is the collection of arcs in the graph, the equivalent to edges. In this thesis we use both terminologies. The two notations are employed to distinguish between different graphs.

Graphs can be associated with weights to model attributes such as travel cost or time, length, etc., depending on the application. Weights can be associated either with vertices or edges. The former is called vertex-labeled, the latter edge-labeled\(^4\). Directed graphs with associated weights are often called networks. Networks are commonly appearing in application fields of

----
\(^3\)Graph theory also deals with infinite graphs.
\(^4\)We consider edge-labeled graphs only.
Graph theory, transportation engineering being a prominent example.

A graph of the form \((V', E')\) is called a subgraph of \(G\), if \(V' \subset V\) and \(E' \subset E\). The number of vertices of \(G\) is denoted \(n\) (\(|V| = n\)), the number of edges is denoted \(m\) (\(|E| = m\)). Given any vertex \(v\), the degree of vertex \(v\)—\(\deg(v)\)—is the number of edges incident with \(v\), with \(\sum_v \deg(v) = 2m\). For a directed graph, \(\deg^-\) denotes the number of outgoing edges, \(\deg^+\) denotes the number of incoming edges.

Let \(e_1, \ldots, e_n\) be a sequence of edges in \(G\). If there are vertices \(v_0, \ldots, v_n\) so that \(e_i = (v_{i-1}, v_i)\) for \(i = 1, \ldots, n\), then the sequence of edges is called a walk. A walk where the \(v_j\) are distinct is called a path. A path with the special case \(v_0 = v_n\) is called a cycle. Two vertices in \(G\) are called connected if there exists a walk between the two. If all pairs of vertices are connected, then \(G\) is called connected. If this criterion only holds for subgraphs of \(G\), then these subgraphs are labeled connected components. Graphs can be further categorized as planar and non-planar. Planar graphs can be embedded into the plane so that edges do not intersect, whereas this does not hold for non-planar graphs. A graph that is acyclic, i.e. contains no cycles, and connected is called a tree, a set of trees is called a forest. A spanning tree \(T\) consists of all vertices and a subset of edges \(T\) of graph \(G\), connecting all vertices of \(G\) without forming cycles. Therefore, a minimal spanning tree is a spanning tree that minimizes the sum of weights \(w(T) = \sum_{e \in T} w(e)\). A popular algorithm for calculating the minimal spanning tree is Prim’s algorithm [8].

### 2.4.2 The Complexity of Algorithms: O-Notation

Complexity theory deals with the time and memory that an algorithm needs to finish, expressed as a function of size \(n\) of the input data. This approach enables the comparison of algorithms for solving the same problem. A strictly
formal theory for that matter was first introduced in [31].

For practical use however, calculating the exact runtime complexity is in general not necessary [15]. The common approach is to estimate the asymptotic runtime of an algorithm [8, 29]:

\[ O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0 \} \]

If \( f(n) = O(g(n)) \) then \( f \) has a growth rate of at most \( g(n) \). E.g., consider the quadratic \( f(n) = an^2 + bn + c \). We say that it has quadratic time complexity \( O(n^2) \), since \( n^2 \) is the highest power of \( f(n) \). Other complexities would be constant \( O(1) \), linear \( O(n) \), or logarithmic \( O(\log n) \).

### 2.4.3 Graph Traversal

There are two basic techniques available for graph traversal: breadth first search (BFS) and depth first search (DFS).

BFS is one of the most fundamental methods in algorithmic graph theory [15]. It is an uninformed\(^5\) search algorithm that first visits all neighboring nodes of a currently investigated node. Then for all those neighboring nodes, it explores their unexplored neighboring nodes and so forth, until the algorithm terminates.

DFS, on the other hand, takes the first neighboring node of the current node, and recursively repeats this process until the solution is found or a sink, i.e. a dead end in the graph, is reached. If the latter is the case, the algorithm returns to the previously reached node and investigates another neighbor.

\(^5\)Uninformed search strategies search a graph without using information about that graph to increase efficiency, in contrast to informed search strategies.
Both BFS and DFS have a runtime complexity of $O(|V| + |E|)$. BFS is important to various problems in graph theory, such as finding connected components, or finding all nodes within such a connected component. DFS is very often part of more complex algorithms. BFS is of special interest here, as it is the underlying principle for the shortest path algorithms presented in Section 2.4.4. It also forms the basis for Algorithm 2.

### 2.4.4 Shortest-Path Algorithms

Shortest-path problems have been among the most extensively studied areas in network flow optimization problems, with a lot of different applications in numerous fields. Since the end of the 1950s, more than two thousand scientific works have been published on this matter. One of the application fields of shortest-path problems is transportation engineering, where various problems need to be solved. Since there is no best algorithm currently known, research in this area has shifted to the design of ad-hoc shortest-path procedures, which can capture specific peculiarities [22, 26].

In this section we will introduce the classic algorithm for shortest-path calculations—Dijkstra’s algorithm—as well as $A^*$-search. $A^*$-search is a modification of Dijkstra’s algorithm, which uses heuristics to increase search speed.

**Dijkstra’s Algorithm**

Dijkstra’s algorithm [10] solves the so-called single-source shortest-path problem on a weighted graph $G = (V, E)$, in case all weights are nonnegative. Therefore, weights are expected to fulfill $w_{u,v} \geq 0$ for all $(u, v) \in E$ [8].

The algorithm is initialized with the graph $G$, the set of weights $w$ and start node $s$. Then, $d[u]$ containing the shortest paths of vertices $u$ is initial-
Algorithm 1 Dijkstra’s Algorithm

1: function DIJKSTRA(G, w, s)
2:     for all $v \in V[G]$ do $\triangleright$ Initialization
3:         $d[v] := \infty$
4:     end for
5:     $d[s] := 0; S := \{\}; Q := V[G]$
6:     while $Q \neq \{\} \triangleright$ Start of Computations do
7:         $u := \text{extract}_\text{minimum}(Q)$
8:         $S := S \cup u$
9:     for all $(u, v)$ do $\triangleright$ Relax Condition
10:        if $d[u] + w_{u,v} < d[v]$ then
11:            $d[v] := d[u] + w_{u,v}$
12:        end if
13:     end for
14:     end while
15: end function

ized and set to infinity. Set $Q$ contains all vertices of $G$ which have yet to be visited. Set $S$ of vertices, whose final shortest-path weights from source $s$ have already been determined, is maintained. The algorithm is constantly selecting vertex $u \in Q$ with the minimum shortest-path estimate (Line 7), includes $u$ in $S$, and investigates all edges leaving $u$. Edge relaxation is performed in case that adding the currently investigated edge $(u, v)$ would decrease $d[v]$.

The algorithm terminates, once the shortest paths to all vertices have been computed. If one is only interested in the shortest path from $s$ to one specific vertex, the algorithm can be altered to terminate in this case. If the graph is not connected, shortest paths are only found within the con-
The runtime complexity of Dijkstra’s algorithm is $O(|V|^2)$ using a brute force implementation. This style is appropriate for dense graphs, sparse graphs profit from different approaches. In fact, runtime complexity can be reduced to as much as $O(|V|\log |V| + |E|)$ using a Fibonacci heap for vertex selection in Line 7.

Dijkstra’s algorithm bears similarities to both BFS and Prim’s algorithm [8]. It is mostly used for shortest-path problems, although there exist modifications to the algorithm that prove more effective in specific situations. Dijkstra’s algorithm can also be found as part of the Open Shortest Path First (OSPF) routing algorithm [19].

A*-search

A*-search is a shortest-path algorithm, first described in [13] and later revised in [14]. It is a graph search strategy similar to Dijkstra’s algorithm (Algorithm 1), but adding a heuristic to reduce runtime complexity. It belongs to the class of informed search algorithms. A*-search is discussed here since its core, Equation 2.1, was a major inspiration for the design of the heuristic presented in Section 3.2.

The main difference to Dijkstra’s algorithm is the evaluation function, which has a heuristic component:

$$f(n) = g(n) + h(n)$$ (2.1)

The function $g(n)$ is the minimal path cost to $n$, $h(n)$ is an estimate for the remaining cost from $n$ to the destination\(^6\). The predominant heuristic for $h(n)$ is the Euclidean distance. Hence, $f(n)$ can be regarded as an estimate for travel costs from start to destination through vertex $n$.

\(^6\)For $h(n) = 0$, the algorithm essentially becomes Dijkstra’s algorithm.
The Euclidean distance has another favorable property: it never over-estimates the cost to the goal, if the cost is geometric distance. Using the Euclidean distance in $h(n)$ is by nature optimistic, since it expects the remainder of the shortest path to be on a straight line. Even though this is not the general case, the algorithm is not in danger of overlooking another, possibly shorter, trip. These types of heuristics are called *admissible heuristics*.

### 2.5 Modeling Paradigms for Simulation

Software modeling and simulation of urban systems had a hard stand throughout the 1970s [18], when claims arose that this approach would not live up to expectations and, in fact, may well lead to wrong conclusions. Though relevant at the time, the field of geographic modeling and simulation was revived by numerous advances in related areas of research. Breakthroughs in the understanding of cities in general, as well as developments in mathematics and computer science, have enabled new approaches in urban geographic modeling. In particular, the rise of the object-orientation paradigm in computer science provides both an intuitive way for modeling, as well as an effective technique for software implementation [2].

Object behavior in geographic simulation is often based on the concept of *automata*. An automaton is a system with characteristics that change over time, due to current internal characteristics and input from the outer world. Specific rules determine how such an automaton changes its state, due to its current state and the input. Automata are abstractions of behaving objects, and quite versatile in application.

Formally, a finite automaton $A$ can be represented through a finite set of
states $S$ and a set of transition rules $T$:

$$A \sim (S, T) \quad (2.2)$$

Transition rules change a current state $S_t$ to $S_{t+1}$, depending on $S_t$ and the input $I_t$ at the current discrete time step $t$:

$$T : (S_t, I_t) \rightarrow S_{t+1} \quad (2.3)$$

In this formulation, the nature of sets $S$, $T$ and $I$ is not defined and may take any form. Automata are the conceptual basis for all further mentioned modeling approaches.

### 2.5.1 Cellular Automata

Cellular automata (CA) are currently the most popular modeling concept in urban geography [2]. Cellular automata are standard automata as described above, but the input is defined in a cellular context. In CA, each cell represents an automaton that is neighboring another automaton in a grid of cells. There are various ways how neighborhood may be defined in CA, the most popular approaches are the five-cell von Neumann neighborhood and the nine-cell Moore neighborhood.

Formally spoken, the definition of a CA is

$$A \sim (S, T, R) \quad (2.4)$$

Here, $R$ denotes the cellular neighborhood of $A$, therefore defining the boundaries of input $I$. Changes of set $S$ are again expressed through transition rules, virtually every possible discrete spatial process may be translated into transition rules for CA. Although cells are stationary in CA, information
can be propagated, depending on the implemented neighborhood as stated above. In geographic modeling, CA is particularly popular for representing areas with varying land use as cellular units.

### 2.5.2 Agent-Based Systems

Agent-based modeling is oriented towards representing the behavior of individual entities, and is vastly used in economics, political science, sociology, and traffic simulation [3]. Agent-based systems specify states and, in particular, transition rules that allow autonomously behaving agents. The level of detail may range from very simple objects to entities that aim at simulating human-like behavior.

Compared to CA, agent-based systems share the concept of neighborhood, but are not confined to a specific spatial location. Mobile agents are commonly active on the nodes of a network, and are subject to frequently changing neighborhoods. These features make agent-based systems attractive for applications in a spatial context, i.e., navigation behavior, wayfinding, and spatial cognition.

### 2.5.3 Geosimulation

Geosimulation is concerned with the design and construction of spatial models, with a focus on urban areas [2]. These models are useful for exploring ideas and hypotheses about the inner works of the complex spatial systems that these models represent. Geosimulation achieves that through object-oriented simulation software, which is then applied to real world scenarios within a spatial context. It employs spatially related automata as a basic concept for modeling. GIS and remote sensing databases serve as the predominant data sources for geosimulation systems.
The special feature of geosimulation is its constituent elements. Geosimulation features a 'bottom-up' design, meaning that higher level entities such as census groupings are a product of the dynamics of animated and inanimate objects at the lowest level of modeling. These high-level entities may be considered emergent, in such as these high levels are significantly richer in characteristics compared to the atomic objects that compound them.

2.5.4 Comparison

All of the three approaches described above can be used to implement the simulation environment. Geosimulation, however, is too complex to be an option since it aims at the most realistic and detailed modeling possible today. Cellular automata are not the first choice for modeling moving objects, since CA are static in nature. Certain similarities nevertheless arise, since the street network of the simulation used here is a regular grid world. The agent-based simulation approach is the most suitable technique, since clients and hosts can be modeled as more or less independent entities with their own behavior. The level of complexity can be adjusted to make this approach feasible, making agent-based simulation the method chosen for this work.
Chapter 3

Deterministic and Heuristic Planning

This chapter discusses the developed strategies and a mobility model necessary for simulating street hierarchies that are exploited for shared-ride systems.

As described in Chapter 2, clients were using the pattern matching strategy in previous work. Although this approach enables the client to reach the destination, free route choice from routes received from hosts via peer-to-peer communication have the potential to enable shorter trip durations.

Clients are not only dependent on the street network, but also on the available means of transportation (i.e. hosts). Clients gather information about available hosts within communication range. This information has to be evaluated so that clients know which locations are currently reachable. The evaluation of received host routes can be done by forming a space-time network (Section 3.1.2). After the space-time network has been compiled, optimal trips can be calculated within the space-time network. Knowledge about hosts is in general incomplete, hence calculated trips are only optimal.
with regard to the set of hosts which could be contacted. In principle, trips
-calculated within a space-time network allow free travel without limiting
-clients to predefined paths. Dealing with the task described above is the
deterministic planning component involved in the trip planning process, as
the gathered and used information is considered to be reliable. In many
cases though, the knowledge currently available about hosts does not allow
deterministic planning. This is the case when the route choice behavior
of known hosts does not allow the composition of a complete trip to the
destination, i.e., gaps occur.

The client is then forced to choose an intermediate location out of all the
locations that can currently be reached. When the client has chosen one of
the reachable locations, a ride is booked in the hope that new options will
arise at this new location (or during the ride to this place).

The choice of this intermediate location is done by using heuristics which
make use of the knowledge of reachable locations derived within the space-
time network as described above. A heuristic strategy, which aims at choosing
the most advantageous intermediate locations with regard to average travel
time, will be introduced. Also, an extension to this strategy is presented
that aspires to predict host occurrence\(^1\). With the first technique, we aim
at accomplishing free travel routes and, on average, decreased travel times
compared to restricting the client to a predefined path. With the second
strategy, we expand this paradigm for the use of a-priori information about
hosts. This means we assume that the client has gathered information about
the previous route choices of hosts, which the client uses to further improve
average travel time.

\(^1\)In the following, if the heuristic strategy and its extension are considered, we may
refer to them as separate strategies or techniques.
In the following we introduce and use a categorization of the shared-ride system we are investigating. Three conceptual layers can be distinguished, which make up the environment of shared-ride systems:

- **Street Network** ($N_S$): the static road network relevant for both clients and hosts.

- **Transportation Network** ($N_T$): the transportation network which changes over time. It consists of all potential means of transportation, such as buses, taxis, and private hosts.

- **Communication Network** ($N_C$): the dynamic communication network. It is built by all participants of the shared-ride system who are within communication range with each other, and who transmit data via multi-hop communication.

The next section describes the problem of waiting times, and then introduces a data structure designed to solve this problem. After that, the two developed heuristic strategies and the mobility model will be discussed in detail.

### 3.1 A Data Structure for Waiting Times

Higher flexibility of clients with regard to trip choice has the potential to optimize travel time and other cost criteria. For matters of easier simulation, clients themselves are assumed to only move by traveling with hosts. Host transfer stops are limited to nodes (i.e. street intersections), not arcs. As a result of these simulation conventions, route calculations are performed on the transportation network $N_T$ only. To achieve this, we need to give clients a tool to cope with the dynamic transportation network.
3.1.1 Waiting Times

A key quality of shared-ride trip planning is the frequent occurrence of clients’ transfer stops, i.e. changing from one host to another. This aspect makes waiting between parts of the trip inevitable. E.g., a client can have the choice between two trips to two different intermediate locations, which may be composed of rides with different hosts. Each of the two intermediate locations is in the same network distance within the street network $N_S$. On one of the trips a host brings the client directly to the intermediate location. The other trip requires the client to leave the host at some point, and continue the ride with another host after a waiting period. This waiting period has to be taken account for in route calculations as well. For our simulation purposes, we associate waiting periods with nodes, not arcs, as clients can only change hosts at nodes. A shortest-path algorithm such as Dijkstra’s algorithm is not designed to deal with waiting times in its original design. Dijkstra’s algorithm considers edge weights, whereas we deal with waiting costs associated with nodes. Consequently, we need a data structure which can model the waiting times common within shared-ride trip planning.

3.1.2 The Space-Time Network

Since $N_T$ is time dependent, we now introduce space-time networks [9, 22]. Given a directed graph $G = (N, A)$, $N$ is the set of nodes with cardinality $n$. $A$ is the set of arcs of cardinality $m$. This is the graph composed of all host routes and the related time schedules that the client device is receiving during a communication window. In dynamic transportation problems a time dependent positive travel time, or delay $d_{ij}(t)$, can be associated with each arc $(i, j)$ which means: if $t$ denotes the nonnegative departure time from
node \( i \), then \( t + d_{ij}(t) \) represents the arrival time at node \( j \). Additionally, time dependent costs \( c_{ij}(t) \) could be used to measure the cost of travel in other terms than time.

Here we want to limit ourselves to the discrete case. The time variable \( t \) may vary within the discrete set \( T = \{t_1, t_2, \ldots, t_q\} \), so that \( t + d_{ij}(t) \in T \). The variable \( q \) limits the finite time frame we consider, and has to be adapted to the specific problem. A discrete model is no loss of generality, as discretization is generally performed in the field of transportation engineering.

The space-time network \( R = (V, E) \) consists of time-expanded nodes \((i, t) \in N \times T \). \( R \) is then formally defined as

\[
\begin{align*}
V &= \{i_h : i \in N, 1 \leq h \leq q\} \quad (3.1) \\
E &= \{(i_h, j_k) : (i, j) \in A, t_h + d_{ij}(t_h) = t_k, 1 \leq h < k \leq q\} \quad (3.2)
\end{align*}
\]

Note that \( R \) is a common graph with \(|V| = nq \) and \(|E| \leq (m + n)q \). In order to model the situation where waiting at nodes is allowed, we introduce waiting arcs in \( R \) in the following way: \((i_h, i_{h+1})\) represents waiting at \( i \) from \( t_h \) to \( t_{h+1} \), and is given the unit time cost \( w_i(t_h), h = 1, \ldots, q - 1 \).

Figure 3.1 depicts a space-time network with time-expanded nodes \((i, t) \in \{1, 2, 3, 4\} \times \{1, 2, \ldots, 10\} \). In this figure, waiting is only possible at node \( 1_h \), represented by the arcs \((1_h, 1_{h+1})\). In our simulation, waiting is permitted on all street network nodes. If the nodes are selected in a chronological order, we can apply common shortest-path algorithms for solving the problem of route planning with permitted waiting at nodes.

### 3.2 Choosing Routes freely: The HLR-Strategy

In a realistic scenario, communication between clients and hosts is performed periodically during so-called communication windows. After each of these
communication windows, the client composes a space-time network from the received route data of hosts where shortest paths to all reachable nodes are computed. After this step, the client further evaluates each reachable node of this set for its potential to become an intermediate location (or a transfer stop) in a shared-ride trip.

We call the proposed strategy HLR-strategy, for Heuristic Location Ranking. The intuition behind this strategy is that the relevance for future travel of a reachable location $i$ is determined by two factors. The first one is the travel time $a(N_T(t), l, i)$ to reach $i$ from the current client position $l$, using various modes of transportation. In general, $a(N_T(t), l, i)$ is calculated on $N_T$ at time instance $t$. This calculation is essentially done on the data structure termed space-time network, outlined in Section 3.1.2. The configuration of the transportation network changes continually. As a consequence of this
change, the space-time network is different in every planning cycle. Furthermore, the client computes the expected minimal travel time $b(N_S, i, d)$ from all reachable locations $i$ to the destination $d$, which is an optimistic guess. For this purpose the client refers to the knowledge of the physical configuration of the street network $N_S$. It is assumed that the street network is constant over the time of the trip, leaving $b(N_S, i, d)$ independent of $t$. In reality, closed roads etc. could be considered as well. Function $b(N_S, i, d)$ can be computed using either Dijkstra’s algorithm or, better, $A^*$-search.

Both functions $a(N_T(t), l, i)$ and $b(N_S, i, d)$ are part of another function. This function $r(N_S, N_T(t), l, i, d)$ stands for the scalar that denotes the potential of a node $i$ to enable the client to reach $d$ from $l$ at a present time instant $t$. $r(N_S, N_T(t), l, i, d)$ stands for the relevance function \[39\].

Due to the dynamics of the shared-ride system, one cannot say that both $a(N_T(t), l, i)$ and $b(N_S, i, d)$ necessarily have to have the same impact on $r(N_S, N_T(t), l, i, d)$ concerning the potential of an intermediate node $i$. For example, it might be more of an advantage if $r(N_S, N_T(t), l, i, d)$ is more sensitive to either one of these two factors. A common approach for modeling a problem like this is using a weighted linear combination of relevant factors $x_1 \ldots x_m$ \[28\]:

$$f_h(n) = c_1 x_1(n) + c_2 x_2(n) + \ldots + c_m x_m(n) \quad (3.3)$$

The constants $c_1$ to $c_n$ are adjusted to give the best fit to the given dataset in terms of solution costs for the problem. With this framework, our model becomes:

$$r(N_S, N_T(t), l, i, d) = -\alpha \cdot a(N_T(t), l, i) - \beta \cdot b(N_S, i, d) \quad (3.4)$$

with $\alpha, \beta \geq 0$. The greek letters $\alpha$ and $\beta$ stand for constants, $c_1$ and $c_2$ from Equation 3.3, respectively. They regulate the impact of both $a(N_T(t), l, i)$
CHAPTER 3. DETERMINISTIC AND HEURISTIC PLANNING

Figure 3.2: Evaluation of locations $i_1$ and $i_2$ using the HLR-Strategy, with $\alpha = 6, \beta = 14$.

and $b(N_S, i, d)$, with $\alpha + \beta = \text{const}$. The bigger the ratio $\alpha/\beta$, the more $r(N_S, N_T(t), l, i, d)$ is sensitive to the time it takes to reach the currently evaluated location $i$, while the smaller the ratio the more sensitive the relevance function is to $b(N_S, i, d)$. Here, both $\alpha$ and $\beta$ have negative sign due to their negative impact on the relevance of a location $i$ for future travel.

The values derived from $r(N_S, N_T(t), l, i, d)$ do not represent an estimate for travel time—the functional components are weighted and therefore distort the output of $r(N_S, N_T(t), l, i, d)$ in this respect. Nevertheless, an ordinal, weight-dependent ranking can be derived which conveys the potential of a node for future traveling. The client then books a ride along the path returned by $a(N_T(t), l, i)$ to location $i$ with the highest result for $r(N_S, N_T(t), l, i, d)$.

Figure 3.2 shows an example where two locations $i_1$ and $i_2$ are evaluated for their travel potential. The relevances $r_1$ and $r_2$ are calculated with predetermined weights $\alpha$ and $\beta$, which proved to be effective in the given environment. The effectiveness of the weights is determined by simulating a number of runs for each possible weight configuration. In an actual simula-
An admissible heuristic is one which never overestimates the cost to the goal [28]. For the HLR-strategy, weights of the two relevant factors $a(N_T(t), l, i)$ and $b(N_S, i, d)$ are introduced. These weights $\alpha$ and $\beta$ are adjusted so that the relevance function delivers a ranking that minimizes average travel time for a given shared-ride system configuration. The relevance function may very well overestimate the ranking of a node, but this is not avoidable since we deal with a system with limited knowledge. Due to these limitations, we cannot speak of admissibility in connection with the HLR-strategy.

The complexity of the strategy is entirely dependent on the shortest-path algorithms used for $a(N_T(t), l, i)$ and $b(N_S, i, d)$.

3.3 Predicting Host Occurrence: The $HLRP$-Strategy

So far, we have only dealt with information provided by hosts in real-time. But in a real world scenario, there is other information that can be used to increase the average duration of shared-ride trips, e.g. schedules of public means of transport or traffic density in general. Most promising could be knowledge about frequently used routes of private hosts. Since routes of private providers are not being announced in advance, we will again employ heuristics to make this class of hosts exploitable for the client.

Consider that the routes taken by hosts are learned from previous times. Some urban areas are more likely to be passed by hosts than others. Then, if the client has the choice between two otherwise equally suited intermediate
locations, the location with more hosts passing should be chosen by the client. This location should have better chances of having hosts passing it when the client arrives at this intermediate location.

Therefore, we investigate the use of data about route choices of hosts as a-priori knowledge. This a-priori data is integrated into the previously described $HLR$-strategy (Section 3.2) to further decrease the average duration of planned trips. In a nutshell, we assume that host routes are heterogeneously distributed in an urban area, and can be observed and exploited for shared-ride trip planning. The new heuristic shall be denoted $HLR_P$-strategy (Section 3.3.2).

For the $HLR$-strategy, simulation of host movements can be done using a random walk mobility model (Section 2.3.3). On the other hand the use of monitored host route choices demands simulated behavior which is repeatable, and thus, predictable. For simulation purposes, we need a mobility model in which we can cause hosts to move in similar patterns as in previous simulation runs. To simulate this correlated host behavior, a suitable mobility model is described in the next section. Section 3.3.2 then describes the heuristic in detail.

### 3.3.1 The Mobility Model

We will employ a synthetic (see Section 2.3.3) mobility model based on hierarchies for the street network, to which hosts will react to. We distinguish the following concepts that form the mobility model:

- A hierarchical street network which defines the host environment.
- A response concept that causes hosts to react to the hierarchical street network.
CHAPTER 3. DETERMINISTIC AND HEURISTIC PLANNING

These two concepts are described in the following.

**Characterizing the Street Layer**

Let us assume the mobility model is designed for a hierarchy with two layers. We distribute labels to the arcs, where \( H \) and \( L \) denote arcs with a *higher* and *lower* chance to be taken. We can define a ratio \( h_r \geq 1 \) of the probabilities \( P_L \) and \( P_H \) with

\[
h_r = \frac{P_H}{P_L},
\]

which defines the probability for an arc to be taken. E.g. by setting \( h_r = 2 \), we make it twice as likely for the host to go along an \( H \)-arc instead of an \( L \)-arc. A value of \( h_r = 1 \) makes both hierarchical levels equal and results in a standard random walk mobility model.

For the computer simulation, a considerable number of simulation runs is performed where hosts follow the chosen structure of the street layer. With the parameter \( h_r \) and the spatial distribution of the labels \( H \) and \( L \), the heterogenous behavior can be tuned.

**Probabilistic Route Choice**

With a heterogenous street network at hand, we can now describe the hosts’ reaction to it. The mathematical formulation consists of \( h_r \), as well as the hierarchical configuration of outgoing arcs at node \( u \):

\[
\begin{align*}
P_L(u) &= \left[ \deg_L^-(u) + \deg_H^-(u) \cdot h_r \right]^{-1} \quad (3.6) \\
P_H(u) &= h_r \cdot \left[ \deg_L^-(u) + \deg_H^-(u) \cdot h_r \right]^{-1} \quad (3.7)
\end{align*}
\]

so that

\[
P_L \cdot \deg_L^- + P_H \cdot \deg_H^- = 1 \quad (3.8)
\]
$P_L(u)$ and $P_H(u)$ are the resulting probabilities for the outgoing arcs of position $u$, depending on the label. Furthermore, this probability is depending on the labels of the outgoing arcs from the current host position $u$. $\text{deg}_L^-$ and $\text{deg}_H^-$ stand for the numbers of outgoing $L$- and $H$-arcs. E.g. we have $\text{deg}_L^- = 1$ and $\text{deg}_H^- = 3$ at a location where the host can choose from four arcs, one of which is an $L$-arc while the remaining three are $H$-arcs.

A host can be denied to go back along the direction from where this host came from (u-turns). This is done to make unrealistic movements in a random walk mobility model more realistic. We call this a memorizing strategy in contrast to a memoryless approach, where the host can choose every direction for traveling.

![Diagram](image)

**Figure 3.3:** A street network with hierarchical labels.

Figure 3.3 shows an example situation of an undirected graph with $h_r = 2$ and $\text{deg}_L^- = \text{deg}_H^- = 2$ at node $u$. The chance for one of the two $L$-arcs to be taken is 16.7%, while it is 33.3% for each of the two $H$-arcs. At all other intersections each street segment has a chance of 25% to be taken, since there is no hierarchical difference. In this example, no distinction is made with regard to the incoming direction, leaving the host with a choice
between four arcs. For a memorizing model, the incoming direction has to be considered. If the client approaches node $u$ via an $L$-arc for example, then the probabilities are 20\% for the remaining $L$-arc to be taken, and 40\% for each of the two $H$-arcs.

So far we have discussed a mobility model that fulfills the Markov property [23], which says that future states are only dependent on current states. The described method is related to the one proposed in [6], where the agent’s movement pattern is described as a three-state Markov chain.

If $X(t)$ is a stochastic process at time $t > 0$, the Markov property states that

$$P[X(t + h)|X(t)] = P[X(t + h)|X(s), s \leq t], \quad \forall h > 0. \quad (3.9)$$

$X(t)$ is the current state of the random function $X$, $X(s)$ can denote any state from past to present and $X(t + h)$ stands for a future state. In other words, the probability for a future state of a stochastic process $X$ depends only on the current state $X(t)$; past states $X(s)$ beyond $t$ do not change this probability. This means that past decisions of hosts do not affect the probabilities for the route choice at the present location.

For representation purposes, we can describe the situation at location $u$ using a transition matrix. This transition matrix consists of $P((u, v) | (l, u))$, which is the probability for a host to travel along arc $(u, v)$ under the condition that the incoming arc is $(l, u)$. For a memorizing strategy we have $P((u, v) | (l, u)) = 0$ for $v = l$.

In our matrix-representation we can denote east (e), north (n), west (w) and south (s) to label neighboring nodes, since we will use a grid street network for the computer simulation. For example, $P(e|n)$ stands for the probability that hosts go east when they came to the current node from north. We can organize all possible constellations in a transition matrix.
Our transition matrix $t(u, c)$ for node $u$—if arrived from cardinal direction $c$—then becomes

$$
t(u, c) = \begin{bmatrix}
0 & P(e|n) & P(e|w) & P(e|s) \\
P(n|e) & 0 & P(n|w) & P(n|s) \\
P(w|e) & P(w|n) & 0 & P(w|s) \\
P(s|e) & P(s|n) & P(s|w) & 0
\end{bmatrix}.
$$

Note that the sum of the values in a column equals one. This matrix defines a host’s route choice probabilities for each current street intersection, depending on their incoming edge. For the general case, the dimensions of the matrix are subject to change.

### 3.3.2 Design of the $HLR_P$-Strategy

With a specific hierarchy ($h_r \neq 1$), movements are not evenly distributed throughout the network. Depending on the spatial distribution of the labels $H$ and $L$, some arcs can have considerably more hosts passing it than other arcs. A two-layer hierarchy, like the one proposed, is a simplification that does not exist in that form. In the simulation clients do not know hierarchical labels, they use the recorded numbers of hosts that pass each edge of the street network graph. This is the more reliable approach compared to learning street classes from a map, also in a real world situation. Hierarchical classification can be done in an urban street network, however this classification is prone to be inaccurate. A road that is classified as *important* from a map may not have many hosts passing. Measured host occurrences are a more reliable data source.

In the simulation, the recorded number of hosts per edge is obtained by executing a large number of host runs before the actual simulated trip of the
CHAPTER 3. DETERMINISTIC AND HEURISTIC PLANNING

client. This data is then used as a-priori knowledge during the client’s trip. The same spatial distribution of hierarchical labels is used as in the previous simulation runs. Therefore, hosts are expected to move in comparable patterns.

Approach

The passes of hosts for each directed arc are stored and used as weights. These weights \( w \) are then normalized \((w')\) by dividing them by the maximum weight \( w_{\text{max}} \) in the network, scaling \( w \) to a range of \([0, 1]\). The obtained values are treated as a measure of the likelihood of a host passing a certain edge \((u, v)\) compared to all other edges.

\[
    w'_{u,v} = \frac{w_{u,v}}{w_{\text{max}}} \quad (3.11)
\]

Algorithm 2 then computes a spanning tree from currently investigated node \( i \) to all other nodes, so that the product of weights \( w' \) along paths \( \pi \) from \( i \) to all other nodes is maximized. The value obtained for the path between currently investigated intermediate node \( i \) and destination \( d \) is then used as a reference for the expected structure of the transportation network between \( i \) and \( d \), because edges in close proximity are considered to influence each other. This means that if node \( i \) has a high value for a path between \( i \) and \( d \), the surrounding area of this path is expected to contribute positively to this result. Thus, the calculated value is not limited to the edges of the path alone, but can be regarded as a measure for the transportation network between \( i \) and \( d \).

If \((u, v)\) is an arc of the currently investigated path \( \pi(i, d) \), then the transport potential \( P_T \) of the path \( \pi_{i,d} \) within the spanning tree is
with $P_T(\pi_{i,d}) \in [0, 1]$. The calculated values for all nodes within the street network can be stored in a look-up table available for the client. This information can be introduced as another factor within the linear combination of Equation 3.4. We expand Equation 3.4 with a new weighted term $c(N_{T_{per}}, i, d)$. The resulting relevance function $r_P$ is therefore

$$r_P(N_S, N_T(t), N_{T_{per}}, l, i, d) = -\alpha \cdot a(N_T(t), l, i) - \beta \cdot b(N_S, i, d) + \gamma \cdot c(N_{T_{per}}, i, d)$$

(3.13)

with $\alpha, \beta, \gamma \geq 0$ and $\alpha + \beta + \gamma = const$. Here, $c(N_{T_{per}}, i, d)$ stands for the transportation potential between nodes $i$ and $d$, derived from Algorithm 2. $N_{T_{per}}$ is the a-priori information about the transportation network, recorded over a certain period of time. The sign of the new term is positive since $c(N_{T_{per}}, i, d)$ increases the potential for future travel of node $i$.

**Algorithmic Implementation**

The associated Algorithm 2 is based on BFS (Section 2.4.3), comparable in structure to Dijkstra’s algorithm (Section 2.4.4). As it is typical for these algorithms, the significant difference lies in edge relaxation [30]. Furthermore, Line 7 extracts the maximum value instead of the more common minimum value, as the algorithm aims at maximizing transport potential.

The algorithm stores the value for the transport potential $d[v]$ for each vertex $v$ found so far between $i$ and $v$. At start this value is initialized as 0 for all vertices, representing the fact that no path is known which leads to those vertices. When the algorithm terminates, $d[v]$ will contain the transport potential from currently investigated node $i$ to all nodes $v$. $d[v]$ remains 0, if no path between $i$ and $v$ was found.
Algorithm 2 Transport Potential

1: function TransPot\((G, w', i)\)
2:  
3:     for all \(v \in V[G]\) do     \(\triangleright\) Initialization
4:         \(d[v] := 0\)
5:     end for
6:     \(d[i] := 1; S := \{\}; Q := V[G]\)
7:  
8:     while \(Q \neq \{\}\) do     \(\triangleright\) Start of Computations
9:         \(u := extract_{maximum}(Q)\)
10:        \(S := S \cup u\)
11:        for all \((u, v)\) do
12:            if \(d[u] \cdot w'_{u,v} > d[v]\) then     \(\triangleright\) Relax Condition
13:                \(d[v] := d[u] \cdot w'_{u,v}\)
14:            end if
15:        end for
16:     end while
17: end function

The basic operation is edge relaxation: if an edge exists between \(u\) and \(v\), then the path with maximum potential from \(s\) to \(u\) \((d[u])\) can be extended to a path from \(s\) to \(v\) by multiplying \(d[u]\) with \(w'_{u,v}\) at the end. This path will have potential \(d[u] \cdot w'_{u,v}\). If this value is higher than current \(d[v]\), \(d[v]\) is replaced by the new value. Edge relaxation is applied until all values \(d[v]\) represent the potential of the transportation network between nodes \(s\) and \(v\).

Two sets of vertices \(S\) and \(Q\) are maintained. Set \(S\) contains all vertices for which the value \(d[v]\) is already known, whereas set \(Q\) contains the remaining vertices. \(S\) starts as an empty set, in each step of the algorithm a vertex is moved from \(Q\) to \(S\). This vertex is the one with the highest value of \(d[u]\).
When $u$ is moved to $S$, every outgoing edge $(u, v)$ is relaxed.

The key factor for running time is the data structure for set $Q$, as well as the implementation of the $\text{extract}_{\text{maximum}}(Q)$ function. In a brute-force implementation of $Q$ as a linked list or array, and $\text{extract}_{\text{maximum}}(Q)$ simply using linear search through all vertices of $Q$, the complexity of the algorithm is $O(|V|^2)$. 
Chapter 4

Implementation

This chapter discusses the overall design and implementation of the agent-based computer simulation environment, as well as the user interface.

4.1 Tools for Development

The programming language of choice was Java™. It is an object-oriented language that compiles source code to byte code instead of machine code, thus enabling platform-independence. Java is currently being developed by Sun Microsystems and a very active community of voluntary contributors.

The simulation application described here was developed with Java compiler compliant to version 1.4. The software development environment for the simulation was the Java IDE of the Eclipse™ framework, in Version 3.1.0. Eclipse is an open-source, platform independent software framework for the development of so called rich-client applications, as opposed to thin-client browser based applications.
4.2 Design and Implementation

Developing a simulation environment with a high level of realism is a complex and time consuming endeavor, and goes beyond the scope of this thesis. The major objective of the simulation is to provide an environment that is sufficient for proving the hypothesis. Basic principles of the simulation environment from [5] were adopted for the simulation described here.

4.2.1 Design Parameters

The simulation shall be able to evaluate three strategies, the $HLR$-strategy, the $HLRP$-strategy and the pattern matching strategy for reasons of comparison. We use a rectangular grid world for performing the agent-based simulations, with time being divided into discrete time intervals. Host movement can be simulated either by a standard random walk mobility model (Section 2.3.3) or the mobility model developed in Section 3.3.1.

Criteria for Host Mobility

Hosts follow a random walk mobility model with certain modifications—as described in Section 2.3.3—for comparing the pattern matching strategy with the $HLR$-strategy. The mobility model from Section 3.3.1 is used for comparing the $HLRP$-strategy to both the pattern matching strategy and the $HLR$-strategy.

All hosts move with the same constant speed, i.e., one street segment per time unit. Hosts are moving constantly and are not allowed to take u-turns (i.e. a memorizing strategy). This means that hosts do not travel back along the street segment they came from. At each time instant $t$, all hosts are positioned at some street intersection, and move to one of the neighboring
intersection nodes (± one row or ± one column) in \( t+1 \). Locations reachable by the client are thus limited to street intersections only.

**Agent Communication**

The communication part is generalized with the transmission range being limited to the 4-neighborhood. This means that all four of the adjacent intersections can be reached by an agent (three at the border of the network, two in a corner). The message is then relayed to all nodes that are reachable by the client via multi-hop communication. As a general approach we chose the flooding strategy for broadcasting (Section 2.2.3). The flooding strategy allows the client to communicate with all peers that are currently within the connected communication component. For reasons of simplicity, we disregard communication delay and let communication happen instantly.

The flooding strategy gives the best knowledge of the transportation network, but communication is extensive and likely to exceed the bandwidth of the communication infrastructure. Therefore, other, more selective communication paradigms need to be considered in a real world implementation [35].

**Client Behavior and Planning**

In the simulation, clients are not motorized and thus depending on hosts for reaching their destination. The client sends a request for the routes of all reachable hosts at each time instant \( t \). The client then receives the routes, incorporates the data into a space-time network, and employs one of the available route-choice strategies on the dataset. The potentially booked trip at time instant \( t - 1 \) is incorporated into the currently evaluated model at \( t \). The route planned at time instant \( t - 1 \) cannot be computed at \( t \), if
the current communication network topology does not allow that. I.e., the connection to hosts involved in the currently booked trip may get lost. It is very well possible that this route calculated at $t - 1$ is more advantageous than the one at $t$, hence this route should not be lost. By incorporating the planned route from $t - 1$ into the space-time network at time instant $t$, we avoid a loss in quality.

4.2.2 Software Structure

The implemented simulation consists of 12 classes and one interface, the class structure is shown in Figure 4.1. The whole simulation is organized in one package labeled simulator, with the executable main-class situated in MainClass.

MainClass

This class is the core of the simulation. Constants in the head of the class make up the user interface, and allow for a variety of simulation parameters to be set. It also offers boolean variables for adjusting the data output. Aside from the interface and methods of the class, it contains the executable main-class main.

A single client trip simulation may be performed several times per program execution, so that average values can be derived from a broader sample. For the HLR- and the $HLR_P$-strategy, the determination of weights is done using a trial-and-error approach. The value-resolution for $\alpha$, $\beta$ and $\gamma$ is preset, and the simulation then performs the predefined number of individual trip simulations for each combination of weights. Table 4.1 shows the weights for a weight resolution of 20 for the HLR-strategy, i.e., 5% steps.

The simulation then returns the results for the one weight configuration
Figure 4.1: UML class diagram of the simulation environment
that yields the lowest average trip duration.

Class Listing

In the following, all remaining classes are listed in alphabetical order and their roles and actions are briefly described.

- **AdjList** is a small interface embedded in all classes that search in a graph.

- **Client** is the object that represents the current position of the traveling client, as well as the desired destination. It contains its current plans as an attribute of type **ClientRoute**.

- **ClientRoute** stores current travel plans in the form of a linked list.

- **Edge** is the multi-purpose class of edges for all graphs of the simulation.

- **Graph** is the class for the graph representation of the regular grid world, in which all agent simulations are performed. This graph may be hierarchical, in case the $HLR_P$-strategy is evaluated. After constructing a hierarchical graph, this class also runs a specified number of host simulations using the mobility model from Section 3.3.1.

- **Histogram** formats the results of average travel time for easier representation and analysis.

- **HostIndex** is an auxiliary data type which is needed for several search operations and calculations, e.g. in determining all reachable hosts.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>$\cdots$</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>$\cdots$</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Weight configurations tested for a resolution of 20.
CHAPTER 4. IMPLEMENTATION

- **HostState** is the model of hosts. A host is identified by an ID, and includes the current position and its travel plans.

- **Probabilities** initiates and stores the measure of likelihood for a passing host, outlined in Section 3.3.2. The actual calculations are carried out by the class **ProbSPT**. Its values are at the disposal of the client using the $HLR_P$-strategy.

- **ProbSPT** performs the calculations for the likelihood of a passing host, according to Section 3.3.2.

- **STGraph** constructs the space-time graph according to Section 3.1.2.

- **TimeDependentSP** calculates the shortest path on the space-time network, represented by **STGraph**.

4.2.3 Testing

The simulation was tested in a *black-box* manner. All crucial subsystems of the simulation were individually tested with different configurations. In case the outcome was incorrect, further testing and correction was done directly on the code.

4.3 User Interface

As a purely scientific application, the simulation targets a very limited audience only, hence usability is not a priority. The simulation relinquishes a graphic user interface, and enables user interaction directly through manipulating source code and the Java output console. The input interface
is situated in the head of MainClass, from which all parameters can be adjusted. Simulation results are communicated through the Java console, where they are shown in a textual manner.

4.3.1 Data Input

Data input happens directly in the code, changing a number of constants of defining characteristics of the simulation environment. Figure 4.2 shows the input interface, where the first part configures the simulation environment. In the second part, the user may turn on/off a number of output features which are controlled by boolean variables. In principle, it is possible to back-trace all client/host movements. However, the amount of data gathered during an extensive simulation run would produce a very large data file. The variables and its actions are further explained directly in the source code.
Figure 4.3: Console output of the simulation.

4.3.2 Data Output

Simulation results are communicated in a textual manner via the Java console. Figure 4.3 shows the output file that results from the configuration in Figure 4.2. With all boolean variables set to false, Figure 4.3 shows only the input configuration and the results of the simulation run. The file commences with the date and time of the simulation start. In the evaluation, the initial configuration is described, followed by the results of the simulation run for the pattern matching strategy, showing mean travel time, standard deviation, 95%-confidence interval, and best- and worst-case trip duration.
Chapter 5

Evaluation

This chapter contains information on how the evaluation process was designed and executed. The parameter convention is given that describes the setting of the simulation runs. Finally, the obtained results are depicted and discussed.

5.1 Evaluation Methodology

As stated in the hypothesis (Section 1.2), the goal of this thesis is to develop a heuristic approach for intermediate location choice, and thus free route choice. For this purpose, we run simulations for each of the two strategies, the $HLR$-strategy and the $HLR_p$-strategy, as well as for the pattern matching strategy.

The parameter that we set in the focus of simulation runs is the number of hosts that are active in the system. We start with the lowest possible number of hosts, and increase this parameter until we reach saturation with regard to trip duration for all investigated strategies. I.e., adding more hosts to the system has no further effect on trip durations. The duration of trips
is expected to decrease as the number of hosts increases. Other parameters of the simulation environment are outlined in Section 5.2.

5.1.1 Dispersion of Trip Durations

We expect the simulated trip durations of the new strategies to be shorter than the ones pattern matching strategy. Hence, our main optimization parameter is average trip duration. Average trip duration, however, does not state anything about the distribution of values. For example, one configuration might yield low average travel time, but at the same time have a bad worst-case performance. As a result, we also present standard deviation and best- and worst-case trip duration.

5.1.2 Methodology for the $HLR$-Strategy

In the evaluation of the $HLR$-strategy, host density shall be defined as the ratio of the number of hosts and the number of network nodes of the simulation environment\footnote{E.g., a grid world of 10 by 10 has 100 nodes, for 50 hosts active in the system we have a host density of $50/100 = 0.5$.}. We run simulations for the pattern matching strategy with the same simulation parameters and host densities. The results shall be compared and visualized, thus drawing conclusions about the effectiveness of the $HLR$-strategy compared to the pattern matching strategy.

5.1.3 Methodology for the $HLR_P$-Strategy

For the evaluation of the $HLR_P$-strategy, the mobility model devised in Section 3.3.1 is configured in a way that there is a regular distribution of hierarchical labels, meaning that rows and columns are alternatingly labeled...
CHAPTER 5. EVALUATION

$H$ and $L$. During the parameter determination of $\alpha$, $\beta$ and $\gamma$, the special case of $\gamma = 0$ occurs as well. This case is essentially the $HLR$-strategy, which allows us to directly compare the results obtained from the two strategies within one single simulation run. The pattern matching strategy is also evaluated in this environment.

5.2 Simulation Parameters

The simulation environment allows manipulation of a variety of parameters. Table 5.2 shows the constant parameters that are the same for all simulation runs and strategies$^2$.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>World length</td>
<td>20</td>
</tr>
<tr>
<td>World width</td>
<td>10</td>
</tr>
<tr>
<td>Start node</td>
<td>91</td>
</tr>
<tr>
<td>Destination node</td>
<td>98</td>
</tr>
<tr>
<td>Host lifespan</td>
<td>25</td>
</tr>
<tr>
<td>Convergence criterion</td>
<td>1000</td>
</tr>
<tr>
<td>Weight resolution</td>
<td>20</td>
</tr>
<tr>
<td>Client runs</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5.1: Constant simulation parameters

We use a rectangular $20 \times 10$ grid world, with start- and destination node in a central, horizontal line. Hosts have a lifespan that allows them to travel 10 street segments until they withdraw. Trips with a certain parameter configuration are deemed unsuccessful and aborted if they exceed the

$^2$Weight resolution is of course not necessary for the pattern matching strategy.
convergence criterion of 1000 time intervals per trip. A client’s trip is run a 100 times. This means that each weight configuration\(^3\) is simulated a 100 times in order to get a reasonably large sample for each configuration.

### 5.3 Results and Discussion

This section shows the results for both the \(HLR\)- and the \(HLR_P\)-strategy. Graphs in the following are given with logarithmic x-axis, in order to put more emphasis on the lower host densities where more simulations were performed.

#### 5.3.1 Performance of the \(HLR\)-Strategy

For the simulation runs, host density was increased by 0.125-steps in the beginning, then in 1.25-steps. This means in essence that at first 25 hosts\(^4\) are added for each simulation run, to get an insight into the behavior for low host densities. Then steps are increased to 250 additional hosts per simulation run in order to get an overview of the behavior until travel times converge to the theoretically shortest trip, which is 8 time cycles. Figure 5.1 depicts the comparison of mean travel time, standard deviation, best-case and worst-case travel time, and the dispersion of weights \(\alpha\) and \(\beta\). This figure also shows the advantage of the \(HLR\)-strategy over the pattern matching strategy in percent, calculated by

\[
100 \cdot \left( 1 - \frac{HLR\text{-strategy}}{\text{pattern matching strategy}} \right)
\]

\((5.1)\)

\(^3\)This not the case for the pattern matching strategy, since there are no weights involved.

\(^4\)This is due to the implementation. Host numbers may be increased by manifolds of the host lifespan.
The results show an overall strong performance of the HLR-strategy. It outperforms the pattern matching strategy on all scales used, with the biggest advantages in the areas of lower host density. That is to say, for the smallest number of hosts, the HLR-strategy is superior on all used scales by a factor of more than two. This advantage of the HLR-strategy persists for all tested host densities, but the effectiveness for both strategies converges at very high numbers of hosts. One can see how the HLR-strategy starts with an advantage of more than 50% on all measures except best case travel
time. In this configuration, best case performance is an exception since both
tactics achieve more or less instantly the theoretically shortest trip during
one of the trip simulations.

The behavior described above makes sense, since for lower densities of
hosts a flexible path can compensate significantly for a lack of transporta-
tion along the predefined path. This advantage diminishes with growing host
densities, as there is a good chance to get a complete or almost complete ride
along the predefined path from the start. Standard deviation and worst case
show the strongest benefit from the HLR-strategy, which indicates signifi-
cantly higher reliability compared to using the pattern matching strategy.

Regarding the behavior of weights it can be seen that by increasing host
numbers, the influence of $\beta$ is reduced while $\alpha$ gains in importance. This shift
in sensitivity from $a(N_T(t), l, i)$ to $b(N_S, i, d)$ can be interpreted as follows:
With small numbers of hosts, it is more crucial that the client closes in to
the destination as quickly as possible, with lesser regard to the time it takes
to get to the intermediate location $i$. The reason for this is that the overall
waiting time for the client is relatively long compared to the time it takes to
reach $i$. However, when the transportation situation improves, the relevance
function becomes more sensitive to $a(N_T(t), l, i)$. This means that there is a
lot of choice for traveling, and the client does not have to be that eager to
reduce just the distance to the destination $d$. The client can now choose with
more regard to $a(N_T(t), l, i)$—and therefore further optimize trip durations—
since the client can rely on a dense transportation network that will provide
a continuing ride soon.

The simulations showed that the HLR-strategy converges for $0 \leq \alpha < \beta$.
However, for $\alpha \geq \beta$ the client would strive to minimize the distance to $i$ and
neglect the need to close in to $d$. This results in the client not to reach the
destination.

5.3.2 Performance of the $HLR_P$-Strategy

Here the same steps in host density have been taken as in Section 5.3.1. The parameter $h_r$—which steers the hierarchical difference between street segments—has been set to $h_r = 6$ in order to distinguish strongly between the two hierarchies. As in the example situation in Section 3.3.1, only intersections with $deg^-_L = deg^-_H = 2$ need to be considered. Since we use a memorizing strategy, these intersections either have $deg^-_L = 1, deg^-_H = 2$ or $deg^-_L = 2, deg^-_H = 1$, in case the host approached the intersection via an $L$-arc or an $H$-arc. The probabilities are therefore:

<table>
<thead>
<tr>
<th>Inc. Arc</th>
<th>$deg^-_L$</th>
<th>$deg^-_H$</th>
<th>$P_L$</th>
<th>$P_H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>1</td>
<td>2</td>
<td>8%</td>
<td>46%</td>
</tr>
<tr>
<td>$H$</td>
<td>2</td>
<td>1</td>
<td>13%</td>
<td>75%</td>
</tr>
</tbody>
</table>

Table 5.2: Probabilities for host choice using $h_r = 6$.

In addition to the $HLR_P$-strategy, both $HLR$ and the pattern matching strategy ($PMS$) are evaluated in this environment. First, the behavior of Algorithm 2 is discussed, as well as minor adjustments to the simulation environment. Then the results are analyzed.

Discussion of Algorithm 2: Transport Potential

Figure 5.2 shows the values derived by Algorithm 2. The values, originally between zero and one, are multiplied with 100 for means of visualization. The values peak close to the destination, as the likelihood for a quick ride is increasing in the destination’s proximity. The destination itself has trans-
Figure 5.2: The results of Algorithm 2, values multiplied with 100.

port potential 1, or 100 respectively. This is not shown in the figure for visualization reasons.

It can be seen that transport potential is non-linearly increasing towards the destination. The different hierarchical constellations of street intersections can be easily identified: for example, sinks in Figure 5.2 are intersections where only $L$-arcs meet.

The results of Algorithm 2 for $c(N_{\text{per}}, i, d)$ are small, and would be significantly smaller than frequently appearing values for the other two functions $a(N_T(t), l, i)$ and $b(N_S, i, d)$. To overcome this obstacle in the given simulation environment, we change the third term of Equation 3.13 to

$$\ldots + \gamma \cdot \log_{1.047}(100 \cdot c(N_{\text{per}}, i, d))$$ (5.2)

We leave the values that indicate transport potential multiplied with 100, as this value fits better to the values derived by $a(N_T(t), l, i)$ and $b(N_S, i, d)$. We then logarithmize with base 1.047, thus scaling the values to reach a
maximum of 100 at the destination. These two actions give more emphasis to the regions that are far away from the destination. The logarithmic function has a linearizing effect on the values, as observable in Figure 5.3. This causes the derived values to be better integrated and weighted within the ranking function.

Results of the $HLP$-Strategy

The evaluation of the three strategies shows once again a large disadvantage of the pattern matching strategy compared to heuristic location ranking. Figure 5.4 shows mean travel time, standard deviation, best case and worst case travel time for the $HLP$-strategy. In Figure 5.5 we find that the $HLP$-strategy brings an advantage over the $HLR$-strategy for the lower host densities. In the beginning, this advantage is about 25% for mean trip duration, and around 40% for the standard deviation. Best case travel time does not profit as both strategies achieve results that are close to the
Figure 5.4: Mean travel time, standard deviation, best case and worst case travel time for the HLR-, the HLR$_P$- and the pattern matching strategy in a hierarchical environment.

Theoretically shortest trip. Worst case travel time, however, maintains an advantage of about 30% for the lower densities. These advantages though converge quickly with increasing host densities.

Weight dispersion for the HLR-strategy exhibits similar patterns as in an environment without hierarchies(Figure 5.5). For the HLR$_P$-strategy, however, weight behavior is not that explicit (Figure 5.5). A closer look at the output files of the simulation runs indicates that this is due to the fact that different weight configurations returned similar results in terms of travel time. Focusing too strongly on $a(N_T(t), l, i)$ in the beginning, however, is detrimental to average trip durations, according to output files.

The simulation results indicate that the HLR$_P$-strategy is a further im-
Figure 5.5: Weight dispersions and advantages for the HLR- and the HLR\textsubscript{P}-strategy in a hierarchical environment.

Improvement over the HLR-strategy in a hierarchical world, however the advantage diminishes earlier compared to HLR-strategy versus pattern matching strategy. To have a closer look at that, we also show results for a host lifespan of 12 instead of 25\textsuperscript{5} (Figures 5.6 and 5.7).

Here the figures show a similar pattern as with a host lifespan of 25, but with a focus on the lower densities. The dispersion of weights for the HLR\textsubscript{P}-strategy shows here that $\gamma$ is very influential in these densities and then decreases, while $\alpha$ gains with rising densities. Weight $\beta$ also contributes positively, but its influence remains rather constant.

\textsuperscript{5}By doing that we can investigate areas of even lower host densities. This needs to be done this way for implementation reasons.
CHAPTER 5. EVALUATION

Figure 5.6: Mean travel time, standard deviation, best case and worst case travel time using a host lifespan of 12.

Figure 5.7: Weight dispersions and advantages for the HLR- and the HLR\(_P\)-strategy using a host lifespan of 12.
Chapter 6

Conclusion and Outlook

The final chapter first gives a summary of what has been done in this thesis. Then, we draw conclusions with regard to the research hypothesis and finish with an outlook on future research regarding the problem domain treated in this thesis.

6.1 Summary

The hypothesis of this thesis is that heuristic route choice strategies can be efficient planning strategies with regard to average trip duration. For reasons of comparison, the planning strategy from previous work was used as well (pattern matching strategy). After evaluating relevant literature, a concept called heuristic location ranking (HLR-strategy) was developed, as well as an extension to it (HLRP-strategy). The HLR-strategy required the introduction of space-time networks, in order to model the complexities of transportation networks in shared-ride tip planning. For simulating the HLRP-strategy, a mobility model was needed that goes beyond the common random walk mobility model. Therefore, a hierarchical extension to the
random walk mobility model was developed and introduced. The developed theory was then implemented in an agent-based simulation environment, followed by an evaluation of the results.

6.2 Conclusion with Regard to the Hypothesis

The hypothesis consists of two parts. The first part reads:

*Heuristic route-choice strategies can provide a solution to the problem of intermediate location choice, thus enabling free route choice compared to the strategy employed before: the pattern matching strategy.*

The first part of the hypothesis could be verified. Heuristic location ranking evaluates all reachable locations on a regular basis, thus allowing the client to move without an a-priori restriction to a predefined path, i.e. the pattern matching strategy. The second part of the hypothesis reads:

*Heuristic route-choice strategies achieve shorter trip durations on average, compared to the route-choice technique currently employed in shared-ride trip planning: the pattern matching strategy.*

This point is strongly supported by the simulation results. Using a standard random walk mobility model, the HLR-strategy outperforms the pattern matching strategy roughly by a factor of two on all scales used. The results for both strategies converge when host densities are further increased.

In a hierarchical street network, the pattern matching strategy is vastly outperformed by the developed alternatives. The $HLRP$-strategy brings an advantage over the $HLR$-strategy for the lowest host densities. This advantage is not as pronounced as between $HLR$-strategy versus pattern matching strategy without hierarchies, and the advantage diminishes more quickly. For
the lowest host densities, however, the $HLR_P$-strategy again provides a substantial improvement, roughly 25% for mean trip duration.

## 6.3 Outlook

This section gives an outlook on a variety of topics which have yet to be addressed in research.

### 6.3.1 Further Modeling and Evaluation

The developed strategies need to be evaluated within a more realistic transportation network and simulation environment. Models for more realistic agent behavior have already been proposed in [38].

The role of transfer stops with regard to overall trip durations has not yet been investigated. Its impact on the relative performance of the discussed route planning strategies remains open for further research.

The strategies have so far been evaluated within a regular grid world, but urban street networks in general do not follow this structure. The influence of these varying street network topologies for trip planning can be analyzed. Heuristic location ranking is expected to perform well within a realistic street network. E.g., a client is not in danger of getting into a dead-end street, the client would stop at the beginning of such a street. Traveling further along such a street is not proposed by the device. Traveling to the dead end would increase both booked travel time, as well as the estimate to reach the destination since the client would have to travel all the way back.

For a more realistic simulation, communication needs to be revised. This goes both for communication protocols as well as for physical phenomena (latency, etc.). In this thesis, communication happens instantly, i.e. zero
CHAPTER 6. CONCLUSION AND OUTLOOK

latency, and with no consideration of bandwidth. For the latter, communication can be limited to constant ranges [33], or even be controlled in a smart way by concepts from time geography [35].

The quality of planned trips needs to be compared to trips calculated from global knowledge. This can be done in simulation as well, e.g. compare the results from a simulated trip to the a-posteriori possible shortest trip. An approach to this problem is described in [12].

6.3.2 Trip Quality and Communication

An improvement of trip quality through the communication strategy and infrastructure should be tried. Finding ways to provide more information to the client increases the reliability, and hence decreases travel time.

Also the situation when there is a communication connection to disconnected subsets of the transportation layer $N_T(t)$ has to be considered. This would mean that there are certain hosts within the network that can not yet contribute, since they cannot be reached at a particular time instant. This might change however, and these hosts might pose a strong contribution. Presently, clients consider only currently reachable locations.

6.3.3 Competition

The impact of competition onto trip planning strategies needs to be investigated. Heuristic location ranking is expected to outperform pattern matching with respect to competition. Since predefined paths are inflexible if multiple clients have overlapping routes, long waiting times for clients in regions with low host densities are to be expected. Heuristic location ranking promises to be less vulnerable to this problem, since bottlenecks in transportation can be easier avoided.
6.3.4 Feasibility and Data Structures

The goal of this thesis was to provide a solution to a recently arisen problem. Shared-ride trip planning is in its earlier design stages, and implementation issues are yet to become more prominent in research.

The computational effort and memory needs for heuristic location ranking can be analyzed with regard to realistic transport situations. The deterministic part of heuristic location ranking (function $a(N_T(t), l, i)$) requires replanning on a continuously changing transportation network ($N_T(t)$). Life-long planning $A^*$ [17] promises to be an efficient technique for this part of the ranking function.
Bibliography


Avenue of the Americas, NY 10020.

Problem Solving. Artificial Intelligence. Addison–Wesley, Reading, MA.

CarTALK 2000: safe and comfortable driving based upon inter-vehicle-
545–550.
URL www.cartalk2000.net


IPTPS’04.


[31] Turing, A., 1936. On computable numbers, with an application to the
Entscheidungsproblem. Proceedings of the London Mathematical Soci-
ety 2 (42), 230–265.


List of Figures

1.1 Car capacities are often inefficiently used (from [7]). 2
1.2 Inter-vehicle communication (from http://www.car-2-car.org/). 4

2.1 Traveling pattern of an agent using the random walk mobility model (from [4]). 16
2.2 Example situation of the grid world described in [5] (5x5 world). 18

3.1 A space-time network (from [22]). 34
3.2 Evaluation of locations $i_1$ and $i_2$ using the HLR-Strategy, with $\alpha = 6, \beta = 14$. 36
3.3 A street network with hierarchical labels. 40

4.1 UML class diagram of the simulation environment. 51
4.2 The input interface of MainClass. 54
4.3 Console output of the simulation. 55

5.1 Comparison of the HLR-strategy (HLR) and the pattern matching strategy (PMS). 60
5.2 The results of Algorithm 2, values multiplied with 100. 63
5.3 Modified values for transport potential. 64
5.4 Mean travel time, standard deviation, best case and worst case travel time for the $HLR$-, the $HLR_P$- and the pattern matching strategy in a hierarchical environment. . . . . . . . . . 65

5.5 Weight dispersions and advantages for the $HLR$- and the $HLR_P$-strategy in a hierarchical environment. . . . . . . . . . 66

5.6 Mean travel time, standard deviation, best case and worst case travel time using a host lifespan of 12. . . . . . . . . . . . . . 67

5.7 Weight dispersions and advantages for the $HLR$- and the $HLR_P$-strategy using a host lifespan of 12. . . . . . . . . . . . . . 67
List of Tables

4.1 Weight configurations tested for a resolution of 20. . . . . . . . 52

5.1 Constant simulation parameters . . . . . . . . . . . . . . . . . . . . . . . 58

5.2 Probabilities for host choice using \( h_r = 6 \). . . . . . . . . . . . . . . . . . 62