Westfälische-Wilhelms-Universität Münster
Fachbereich 14 Geowissenschaften
Institut für Geoinformatik
Sven Teßmann

Diplomarbeit

Time Geography for Efficient Shared-ride Trip Planning in Transportation Networks

Sven Teßmann

Mai 2006

Gutachter:
Prof. Dr. Martin Raubal
Dr.-Ing. habil. Stephan Winter

Abstract

Shared-ride trip planning is one application area of mobile geosensor networks. The challenge is to assign ad-hoc passengers to vehicles with free transportation capacity. Winter and Raubal have suggested to use time geography for solving the route planning task on these non-deterministic systems efficiently [WR06]. This thesis extends their work to the area of transportation network structures that have their own special properties in order to contribute to the development of a shared-ride trip planning service. Network-based algorithms and data structures are presented that assist clients and hosts in the search for optimal travel assignments. A multi-agent simulation software (`P2PSim`) is designed and implemented to test and verify the proposed strategies in a real street network. Simulation runs are recorded and analyzed with regard to the effectiveness and efficiency of the solution.

"Shared-ride trip planning" ist ein möglicher Anwendungsbereich mobiler Geosensornetzwerke. Die Herausforderung besteht hier in einer ad-hoc Zuordnung von Passagieren zu Fahrzeugen mit freier Transportkapazität. Winter und Raubal schlagen vor, die Konzepte der Zeitgeographie zu nutzen, um die Aufgabe der Routenplanung in solchen nicht deterministischen Systemen effizient zu lösen [WR06]. Diese Diplomarbeit erweitert den Ansatz um den Bereich der Verkehrsnetze. Ziel der Arbeit ist es, einen Beitrag zur Entwicklung eines "Shared-ride trip planning service" zu leisten. Dementsprechend werden netzwerkbasierte Algorithmen und Datenstrukturen vorgestellt, welche die Clients und Hosts bei der Suche nach optimaler Zuweisung von Fahrten unterstützen. Darüber hinaus wird der Entwurf und die Implementierung von `P2PSim`, einer Multi-Agenten Simulationssoftware, dokumentiert. Mit `P2PSim` werden die vorgestellten Strategien in einem realen Straßennetzwerk simuliert und verifiziert. Im Anschluss werden die aufgezeichneten Simulationsläufe in Hinblick auf Effektivität und Effizienz der eingesetzten Lösungen analysiert.

Contents

List of Figures

List of Tables

Mobility is fundamental to economic and social activities ...

(Jean-Paul Rodrigue [RCS06])

1. Introduction

The first chapter is an introduction to the thesis. At first the motivation of the work is presented. In the next sections the hypothesis and methodology are introduced, followed by a section on the relevance of the work. The chapter closes with a summary of the organization of the thesis.

1.1. Motivation

Location-based services provide mobile users with information with respect to their geographic location. These services are becoming more common and popular today, as mobility increases and mobile computers become more powerful and affordable. Combining mobile computing with location sensing technologies new realtime services are made possible, which use spatial data to assist people with their daily activities. Examples reach from consumer-oriented services, like "friend finder" and mobile games, to business-oriented services, such as fleet management and emergency services.

Trip planning is one application area where a location-based service offers new possibilities: traffic data, such as road congestion, schedule changes or weather information, which is relevant to a specific location can be dynamically incorporated into the planning process, thereby improving the assistance of mobile users. Even more interesting is the ability of such a service to keep track of the current travel demand and supply. Using mass transit and ride sharing this can be utilized to perform intelligent travel assignment and increase the efficiency of transportation networks [DWN02]. Most of the deployed systems today, however, use a central service provider with a client-server architecture that is not scalable as the number of mobile users increases.

In current research a shared-ride trip planning system is envisioned that works totally different. It uses mobile geosensor networks to negotiate between current travel demand and supply by using ad-hoc wireless communication between mobile users.

Finding optimal travel assignments in this environment is challenging, because knowledge for planning is limited both spatially and temporally. For trip planning in such an environment theoretically global network knowledge about all travel plans is needed, but not achievable realistically. Minimizing energy consumption is an important concern, therefore efficient communication strategies have to be devised. Winter and Raubal [WR06] propose a solution to those planning tasks using concepts from time geography. Originally developed by Hägerstrand, time geography uses various space-time concepts to describe people's ability to trade time for space and to participate in activities. The solution uses time geography to enable hosts in such a network to decide whether their travel plans are relevant to a travel request before communication starts. This way, as the authors show, it is theoretically possible to drastically reduce communication costs, while still finding optimal trips at a given point in time. In the proposed solution the increase in computational cost is minimal, because the necessary calculations are based on simple geometries of cones and frustrums. Towards a practical application some questions remain open, including the projection of these concepts to real transportation networks that have special space-time geometries and properties.

This thesis focuses on applying this approach to transportation networks, exploring the consequences and developing a simulation environment to confirm the theories. In particular the potential increase in computational cost is investigated.

1.2. Goal and hypothesis

The goal of this thesis is to find algorithms and data structures for efficient shared-ride trip planning by extending the proposed solution from [WR06] to real transportation networks. The results are used to implement an agent-based simulation environment, where specific travel scenarios can be tested.

The hypothesis is:

1. Time geography can be used to help solve the route planning task of a shared-ride trip planning system in transportation networks.

2. The solution, consisting of travel assignment strategies, algorithms and data structures, finds optimal trips, in terms of minimizing a cost function, using local knowledge of the network at a given point in time.

3. The solution is efficient, meaning it significantly reduces communication cost, while limiting computational cost to less than quadratic time complexity.

4. A simulation environment can be designed and implemented for testing the solution in a real street network.

1.3. Methodology

The methodology consists of the following steps: at first, based on previous work, the problem domain of shared-ride trip planning is examined. In a second step time geography is introduced and solutions to the trip assignment and planning problems are described. Then similar strategies that work on network structures are sought, as well as algorithms and data structures for implementing a network-based shared-ride trip planning system. Finally the design and implementation of a geosimulation environment is described and used to verify the network-based solution, by testing various scenarios.

1.4. Relevance of the work

The verification of the hypothesis would contribute to the implementation of a peer-to-peer shared-ride trip planning service deployed in an urban environment. Efficient network-based algorithms help to operate in a geosensor network with mobile nodes, by reducing the complexity of communication and computation. Direct benefits include lower battery consumption of mobile devices, lower bandwidth usage, higher tolerance to faulty network nodes, lower memory and computing power requirements to mobile devices as well as increased scalability of such a system.

1.5. Organization of the thesis

The second chapter introduces shared-ride trip planning and solutions to the ad-hoc travel demand problem. Time geography and its concepts are introduced in the third chapter, together with improved communication strategies. Chapter 4 describes the integration of the solution from previous work with transportation networks. Algorithms and data structures for a network-based solution are discussed. In chapter 5, after an introduction to geosimulation, the development of a simulation environment is described. The final chapters present and discuss results of the simulation runs and draw conclusions with regard to the hypothesis. Possible improvements are suggested and open questions are discussed.

2. Shared-ride trip planning

People use transportation networks to get from one location to another. In a modern urban environment, they have many options to choose from: public transportation, cars, bikes or walking. For most of these choices familiarity with the transportation network in terms of maps and schedules is required. Trip planning has to be done before the start of the trip and works well if people have some knowledge about the environment and if there are no major deviations from the travel plan. Trip planning can be assisted by computers systems, such as web-based travel planners, desktop applications and software running on mobile computers. These systems produce route instructions, which have to be followed by the user who whishes to reach the destination. This way less knowledge about the environment is needed, but the success depends on the quality of the generated route instructions.

What happens when trip planning, using the traditional approach, is not possible? Consider the following scenario:

A business traveler (Ann) arrives at the station in the morning and is scheduled for a meeting one hour later in a downtown office building. Ann decides to use public transportation to reach her destination. On the way she misses a bus and is now looking for a shared ride that allows her to still reach the destination on time. Using a handheld device she starts negotiation with devices in nearby vehicles. Soon her device receives an offer for a ride matching her request. After Ann books the ride, the driver picks her up and she reaches her meeting on time.

In this example, Ann is looking for alternative transportation methods because she missed a bus. This can be thought of as "ad-hoc travel demand" with travel supply, in this case other drivers offering a seat in their cars, depending on her location, time and desired destination. The problem in this case is to negotiate between travel demand and travel supply efficiently, in order to make trip planning possible. Beyond this example, a solution would contribute to a more efficient usage of today's transportation networks, because a large amount of available transportation supply remains unused. This chapter explores different solutions.

Figure 2.1.: Shared-ride trip planning scenario: missing a bus.

2.1. Hitchhiking

One solution to matching ad-hoc travel demand and supply has been around for a long time: hitchhiking. Hitchhiking is a transportation method where a traveler tries to get a ride from a car or truck driver by placing herself close to the road or at parking lots. The hitchhiker uses a thumbs-up sign to communicate her travel demand. The success rate depends on traffic density, road condition and also on factors like the weather or the gender of the hitchhiker [GFL04]. The most limiting factor, however, is communication range – the hitchhiker can only be spotted by drivers using one specific road and direction. The low success rate and the low reputation of hitchhiking make it a suboptimal solution to the ad-hoc travel demand problem.

2.2. Dial-A-Ride systems

With the advent of telecommunication, methods were developed to bring together transportation demand and supply of people and goods. One example are "Dial-a-Ride" systems where passengers call a control center, which re-routes buses or cars to service the requests [NOS$^+$03]. This represents one method of offering service to low demand regions or at low demand times. Calling a cab is another example, with a stronger emphasis on profit-making.

While this approach does offer some advantages compared to the hitchhiking approach, it also has several drawbacks. It relies on the telephone network and the control center is a natural bottleneck in the system. It has been demonstrated by simulation that the performance of these system decreases under high demand situations [NOS+03].

2.3. Location-based services

Location-based services (LBS) aim to provide information to mobile users with respect to their current location [Spi04, BCI+01]. Usually a subscriber of such a service uses a mobile phone or other handheld device to communicate with a central service, which performs tasks to provide users with relevant information. Communication relies on a cellphone network and location sensing techniques are used to retrieve the user's location [Lju01]. Services are implemented as *push services*, where the user receives information based on her location and *pull services*, where the user has to actively request information from a communication network. LBS have been successful with services like "friend-finder" [BD03], tourist guides [CDMF00] and can also be used to assist with trip planning and navigation.

Proposed trip planning and assignment systems such as the Intelligent Travel Assistant (ITA) offer services to the mobile user, such as ride-sharing offers, route planning and traffic information [DWN02]. This is achieved by using a central travel computer, which uses a location database, dynamic travel data and tracking information of mobile users to get a global knowledge of the transportation system. Planning is done by a central instance upon user request and information is transmitted back to the mobile device.

The question is, whether LBS can be used to solve the problem of ad-hoc travel demand efficiently. Having a central service provider and a communication network, users looking for transportation supply transmit their travel demand to the service provider, which in turn solves the task of assigning transportation offers to transportation requests. However, there are several drawbacks to this approach:

- cellphone network and device must be available
- communication fees to cellphone network provider must be paid
- central service provider represents a single point of failure
- frequent update of network data by the service provider to reflect changes

- solution does not scale well in high demand situations

A possibility to overcome the need of a central service provider is introduced in the next section. By considering peer-to-peer communication between mobile users, novel solutions to the travel assignment problem become available.

2.4. Mobile geosensor networks

With latest advances in microelectronics and embedded computing new applications utilizing spatial information are made possible. Small and inexpensive sensor devices can be deployed in large numbers almost anywhere on the surface of the earth [KKP99]. They can be used to record realtime data about phenomena such as weather or traffic density, enabling new monitoring applications [CEE+00]. Using short-range wireless communication, sensors can retrieve and exchange information with their immediate environment, thus forming sensor networks [ZG04]. Geosensors are sensor devices that are aware of their geographic location [NSC+03]. To acquire the current location different location sensing techniques can be used. Often the obvious choice of using a signal from the Global Positioning System (GPS) has to be discarded due to the small size, cost and power constraints of sensor nodes [BHE00]. There is an ongoing research on alternative location sensing methods, most of them using physical properties of the radio signal, such as propagation delay and signal strength [SBGP04, CHH01].

Geosensor networks can be formed by deploying a large number of geosensors in an area of interest, while keeping a small enough grid to allow for communication. In such a network geosensors communicate with other geosensors in their vicinity in an ad-hoc manner. These systems are not limited to static nodes, they often consist of moving peers.

Communication strategies in these networks differ from classic client-server based communication, because typically routing of messages is done via “hops” from one node to adjacent nodes, until the destination is reached. These types of networks are called mobile ad-hoc networks (MANET) [MLG03]. MANETs are characterized by their arbitrary, rapidly changing topology. Routing of messages in these networks represents a challenge, because routers move freely and can appear and disappear at any time. The basic idea is that network nodes announce their presence and listen to announcements from neighbors, thereby discovering the network topology over time [Tan03]. Many different ad-hoc routing protocols have been developed and can be categorized as table-driven, on-demand and hybrid [RT99]. Table-driven routing protocols require nodes to maintain and update a list of destinations and routes as they receive and forward messages[1]. On-demand routing protocols create a route only when requested by a source node through a *route discovery process*, which involves broadcasting a route request packet by the source node and waiting for a route reply packet to be sent back. In general, on-demand routing is better suited for mobile computing, because it uses less resources, such as bandwidth and battery power [RT99].

Nittel [NDK04] introduces the term MAGNET for a self-configuring mobile ad-hoc geosensor network, emphasizing the location-dependency of information in the network. MAGNETs require specific information dissemination strategies that differ from other communication networks [NDK04]. In section 2.4.2 some of these strategies are introduced in conjunction with shared-ride trip planning.

Geosensor networks will likely change the application domain of location-based services and the integration of both areas is a hot research topic today. Using this technology new trip planning and assignment systems are possible.

2.4.1. Ad-hoc shared-ride trip planning

Recently ad-hoc shared-ride trip planning was proposed [WN05]. This could solve the the ad-hoc travel demand problem described in the beginning of the chapter. The approach is based on peer-to-peer communication in mobile geosensor networks. Their envisioned shared-ride trip planning system consists of mobile network nodes: clients with a travel demand and hosts offering travel supply. Hosts and clients represent participants of transportation networks: pedestrians, cars or buses for example. Winter and Nittel have shown that a peer-to-peer based approach within these networks is possible and represents an effective and efficient alternative to traditional trip planning solutions.

The authors define three components for a trip planning and assignment system:

1. shared-ride trip planning
2. shared-ride trip revision

[1] Popular examples are the Destination-Sequenced Distance Vector Routing protocol (DSDV) and the Wireless Routing Protocol (WRP).

3. shared-ride trip traveling

The first step contains the matchmaking process, that is the formulation of the client's request, the search for an optimal travel opportunity and the assignment of the client to a host's offer. The second step describes the continuous revision of the travel plan, allowing for the unpredictable nature of a geosensor network. At any given point in time only local knowledge can be used for trip planning. The search must be an iterative process, as new hosts and clients enter and leave the network. The last component involves the physical agents, mainly focusing on social aspects of the system. This involves issues regarding privacy, trust and safety. One central question is how people can be brought to share resources, such as passenger capacity. Since this work is concerned with the subject of optimizing the communication between hosts and clients in such a system, the first step is explored in detail below.

Communication and negotiation are described to work as follows:

1. A client formulates a request for a shared-ride trip containing its current location and the desired destination. The request is then transmitted to nearby network nodes and rebroadcasted further in a peer-to-peer manner. It is the client's responsibility to compute an optimal trip, for instance using a shortest-path algorithm, before communication starts. Hosts are reactive only. The request contains the complete route with timestamp information.

2. Hosts have their own independent travel plan and are assumed to be willing to pick up passengers on their way without detours. Hosts receiving a request decide if they are able to offer a ride to the client based on their future travel plan and current passenger capacity.

3. The client, having received all offers, chooses the best offer and books the trip. This decision can be based, for instance, on preferring earliest departure or arrival times.

2.4.2. Communication strategies

For shared-ride trip planning, as one application area for MAGNETs, efficient communication strategies are necessary to exchange information about travel supply and demand. Besides routing protocols, the issue of opportunistic message exchange is important in this context: in the beginning, clients and hosts have no knowledge of other nodes in the network, they discover other peers by broadcasting messages to their vicinity, anticipating other nodes to be in communication range of them. This applies to request messages and the rebroadcast of messages (offer and booking messages are routed on a defined path between the client and a host). Nittel et. al. discuss information dissemination in MAGNETs by opportunistic ("routing free") message exchange [NDK04].

The authors introduce the following strategies:

- *Flooding*: If a geosensor receives a message it rebroadcasts it to every other node within its communication range. The receiving nodes repeat this process.
- *Epidemic*: Each node receiving a message passes on the information to only n other nodes.
- *Location-constrained*: Information is only passed on within a spatial buffer around the origin of the message.

These strategies control the distance information is passed on from the message origin and the time the information persists in the network. These factors have a direct influence on the performance of communication in MAGNETs, because unrestricted rebroadcasting can lead to exponential increase of message exchange as the number of geosensor nodes increases. Shortly after the initial broadcast most of these messages are redundant, as has been demonstrated by simulation for all three strategies [NDK04]. The location-constrained strategy offers a good overall performance.

The latter communication strategies focus on one-way communication, for applications such as a hazard-warning system for vehicles equipped with geosensors, where information about a hazardous road condition spreads outwards from the detected location. For an ad-hoc shared-ride trip planning system two-way communication strategies are needed. Winter [WNed] suggests adjusting the above communication strategies for the negotiation cycle in shared-ride trip scenarios, resulting in the following strategies: (1) *Unconstrained*: Each node rebroadcasts a message if it has not already done so. Eventually clients will get offers from all reachable hosts. (2) *short-range proximity*: Clients only communicate with nodes within their communication range and no messages are forwarded. (3) *mid-range proximity*: Client communicates with hosts within a spatial buffer defined by a fixed number of hops. Only by using the first strategy the client can expect to find a global optimum offer. The other strategies represent a trade-off between reduced network traffic and finding relevant hosts. It was demonstrated by simulation that the mid-range proximity strategy is an effective and efficient communication strategy for shared-ride trip planning scenarios [WNed].

2.5. Comparison

For ad-hoc travel demand scenarios different approaches have been presented. For usage in dynamic transportation networks, a decentralized approach based on autonomous communication between network nodes seems to be the most promising solution. The proposed shared-ride trip planning system has several advantages: (i) it does not depend on a central service provider, (ii) it does not rely on a telecommunication network and (iii) it finds optimal solutions given local knowledge at a time t.

In order to realize such a system, there are several things that have to be considered: The use of geosensors and radio-based shortrange communication requires that communication and computation costs must be kept at a minimum. Usually geosensors are battery-powered and only "wake up" for short communication windows, within which all communication and computation tasks must be completed. For a large number of nodes in the network bandwidth also becomes a problem. Communication includes the following tasks: (i) request messages sent by clients, (ii) host's offer messages and (iii) rebroadcasting of messages. Most likely the offer messages and the rebroadcast have the greatest potential of reducing communication cost, as only a few hosts will have a travel plan that can contribute to the client's request. The computation tasks consist of: (i) host's matchmaking of clients requests and (ii) filtering of received offers and requests. Another barrier is the assumption from the initial approach that the client has knowledge of the street network and needs to request with a precalculated route. This can be a problem for mobile devices, as memory and computing power is limited. Also requesting with a predefined route may lead to suboptimal routes, as the best route might not be the shortest one [WNed]. It is desirable to find methods to relax these preconditions, for instance enabling the client to request with its current and destination location only. In the next chapter methods are introduced to solve some of the problems mentioned.

3. Time geography

This section introduces time geography and describes concepts that help to solve the trip planning and assignment tasks discussed in the previous chapter.

In his 1970 paper “What about people in Regional Science” Hägerstrand investigates relationships and constraints between people and resources in space and time [Hä70]. He realizes that people can only be present at a particular location in space at a given point in time. Hägerstrand defines space-time concepts and mechanisms that describe people’s ability to trade time for space. These concepts form the basic elements of time geography. A space-time framework describes constraints that limit people’s ability to participate in events in space and time, at the same time defining necessary conditions for human interaction [Mil91]. Time geography can be seen as a framework to aid in the understanding of human spatial behaviour [Mil05a].

3.1. Time geography concepts

This section gives a short overview of the terminology used in time geography.

3.1.1. Space-time paths

Space-time paths represent the movement of individuals in space plotted over time. Figure 3.1 shows a space-time path of a person’s daily activities. Paths can be visualized at different spatial and temporal resolutions, depending on the area of application. The slope of the path represents the travel velocity. Locations at which individuals remain stationary, to undertake activities that require them to stay at one place for instance, are called *space-time stations*. These are visualized as tubes with the space-time path going upward and usually refer to activities like sleeping and eating.

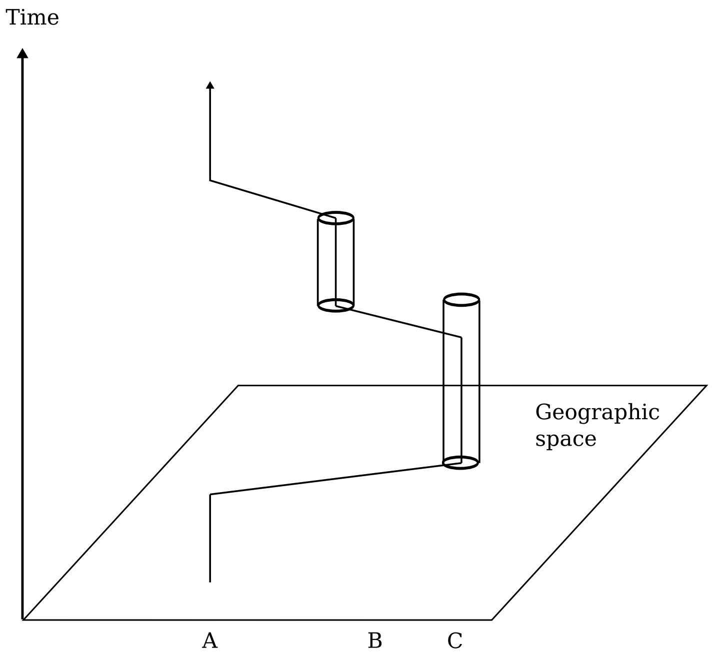


Figure 3.1.: Space-time path of a person's daily activities.

3.1.2. Space-time constraints

Time geography defines different constraints that limit the activities an individual can participate in. Fundamental physical restrictions on abilities and resources are summarized as *capability constraints*. Not having access to a car in order to trade time for space efficiently is one example for a capability constraint. Another class of constraints is described as *coupling constraints*, referring to the requirement for a person to be at a specific location at a certain time or for a fixed time duration. For instance in order for a family to have dinner together requires every family member to be at home in the evening. *Authority constraints*, a third class of constraints defined by time geography, refers to restrictions on activities implemented by authorities, such as the government or private ownership. For instance one cannot cash in a cheque if the bank is closed.

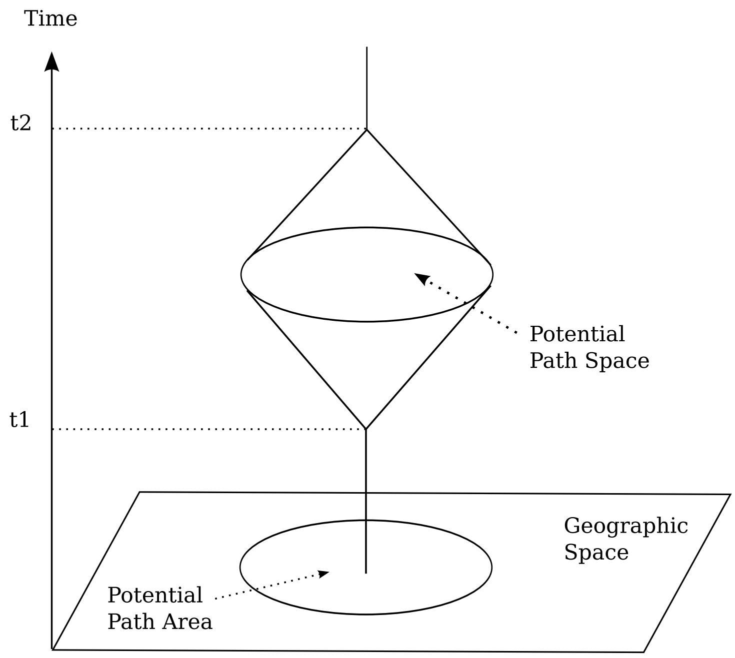


Figure 3.2.: Example of a space-prism.

3.1.3. Space-time prisms

A *space-time prism* (STP) can be defined as an extension of the space-time path, as it describes the locations in space-time that an individual can reach between fixed activity locations [Mil05a]. The space-time prism is the geometric expression for the possible locations a person can move to between two fixed activities given a certain time budget [Hä70]. Figure 3.2 depicts a space-time prism for a scenario where the origin and destination location are coincident in space. The time budget is defined by $\Delta t = t_2 - t_1$ in which a person can move away from the start location, limited only by the maximum travel velocity. In figure 3.2 this is a constant and finite velocity in all directions. In a real street network the movement is limited by the network geometry and by the maximum travel velocity, which can vary in different directions and at different times. This fact and its implications are discussed in Chapter 4.

Given a constant maximum travel velocity the space-time prim can be constructed by two cones: one originating in the location at time t_1 and going forward in time and another originating in the location at time t_2 and going backward in time. The slope of the cones is defined by the maximum travel velocity. The interior of the prism defines a *potential path space* (PPS), which represents all locations in space and time that an individual can occupy during Δt [Mil05a]. The projection of the PPS onto geographical space is called *potential path area* (PPA).

One important conclusion that can be drawn from space-time prisms is the fact that an activity or person is only accessible, if its space-time station or its space-time path intersects the space-time prism of another individual for a sufficient time period [Mil05b]. This way space-time prisms can be used to answer space-time queries, for example: Which locations are reachable within a given time limit? or How long can a person remain at one flexible activity location?

3.2. Shared-ride trip planning strategies using time geography

In [WN05] and [WNNC05] strategies for finding optimal trips in shared-ride trip scenarios are discussed. In order to find an optimal trip from the current location to the destination a client can request travel plans from all hosts in the network and extract from this list the ones that can contribute. An optimal trip is defined as the route that minimizes the cost function, for instance the quickest route for the client at a given time t. If the peer-to-peer network is sufficiently dense, theoretically the client's request can reach all hosts in the network via inter-node hops. This method, however, represents a huge communication effort (see chapter 2). It has been stated that it is desirable for a request message to only contain the start and destination location. Then the only way a host could decide whether to send back an offer or not is to apply a shortest-path network algorithm and then run a matchmaking on its own route. In a geosensor environment this is not a practicable solution.

Winter and Raubal [WR06] suggest filtering techniques based on space-time prisms, constructed with information from a client request that require a minimum of communication and computation effort. Using an arbitrary[1] latest arrival time t_1 and a space-time diagram, a cone centered on the destination location going backward in time down to the start time t_0 can be constructed. This can be thought of as one half of a space-time prism as introduced in the previous section. The slope of the cone is defined by walking speed in this case. The base circle of this so called *dl*-cone (destination, latest) contains all hosts that can potentially contribute to the client's

[1] Walking time as a reasonable upper time limit is suggested.

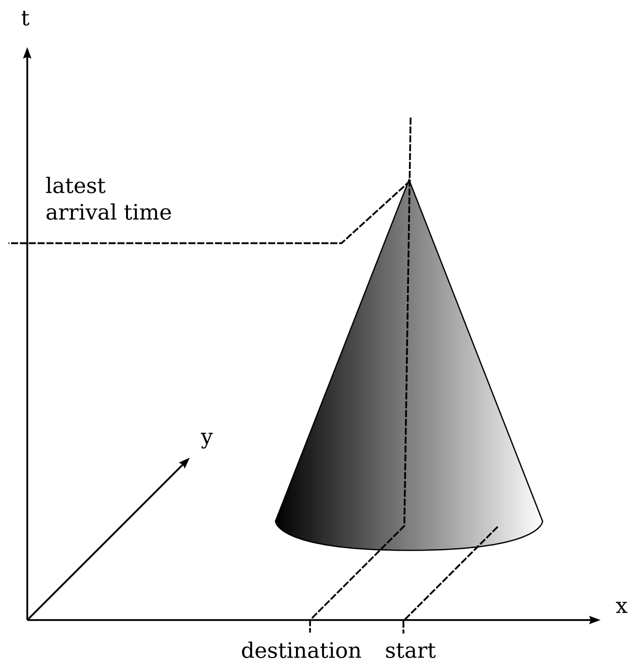


Figure 3.3.: Example of a *dl*-cone.

request at time t_0, because they can reach the destination within the given time interval $\Delta t = t_1 - t_0$. Figure 3.3 shows an example of a *dl*-cone. All hosts outside the *dl*-cone cannot contribute to the client's travel plan, thus this method already reduces the communication effort by eliminating these messages.

In a second step not only locations of hosts are considered, but also their future travel plans [WR06]. By constructing a second so called *se*-cone (start, earliest) centered on the start position of the client at time t_0 and going upward until time t_1 the travel possibilities of the client from the start position are considered. The slope of the *se*-cone is defined by the maximum travel velocity in the network. Intersecting the *dl*-cone and the *se*-cone results in a space-time prism that defines the possible movements of the client between time t_0 and t_1 (see figure 3.4). This can be used to further limit the number of hosts that have to be considered, because a host's travel plan must intersect the space-time prism of the client in order to be valuable.

Comparing the original strategy with the strategy involving the client's space-time prism the authors found a theoretical reduction of communication cost (hosts sending an offer to the client) down to $\frac{1}{28}$ of the original communication cost. This

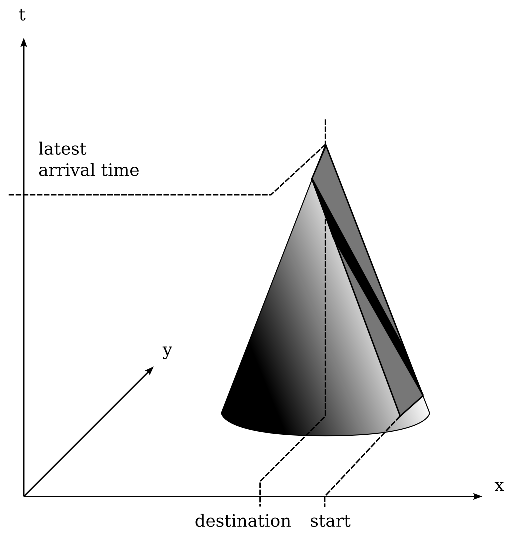


Figure 3.4.: Example of a client's space-time prism.

was supported by an example using cellular automata [WR06].

Admittedly in this research the application of concepts from time geography was accomplished with the assumption of a homogeneous street network and constant travel velocities in all directions and at all times. This results in perfect cones and space-time prisms with simple geometries. In most urban settings a large amount of space is not available for participation in activities or travel [Mil91]. The question comes up, whether the projection of the cones and prisms to a street network and the varying travel velocities in a real street network influence this result. Therefore, in the following, the method of reducing computation and communication effort by using time geography elements are applied to a real street network and the consequences are tested in a simulation environment.

4. Efficient shared-ride trip planning in transportation networks

This section first introduces transportation systems and how the modeling of transportation networks can be accomplished. In a second part network algorithms are discussed, which can be used for trip planning. Then, following the methodology, shared-ride trip planning strategies using time geography are applied to networks and algorithms are described which are needed for an implementation of such a system. Algorithm complexity is discussed where appropriate.

4.1. Transportation systems

The purpose of transportation is to overcome space, which is a basic human ability limited by distance, time and topography [RCS06]. With the ever-growing demand for mobility the modeling, analysis and simulation of real transportation systems and networks has become more and more important. Societies and economies are heavily dependent on transportation systems today, as they are the basis for travel, resource allocation and trade. Intelligent transportation systems (ITS) use network monitoring and optimization techniques in order to improve efficiency of transportation systems [Nor00]. ITS have a high economic importance, for example providing localization and distribution of resources.

Conceptually a transportation system contains

- transportation nodes, serving as primary access points of a transportation system,
- transportation networks, defining the spatial structure, support and organization of transport and
- transportation demand, which is served by transportation supply.

All three parts are related and dependent on each other [RCS06]. Accessibility plays a prominent role in transportation systems. It describes the potential of a location to reach, and be reached by, other locations. Finding solutions to accessibility problems is among the popular usages of GIS tools, which model transportation networks. Examples for applications include

- Optimal route selection.
- Defining catchment areas.
- Optimal resource placement.

Early interest in transportation systems was dominated by mathematics and research focused on operations and topology [Nor00].

4.2. Network models

Transportation systems are modeled as networks. Locations are represented as nodes and routes are defined as links between nodes. Depending on the type of application, network models can differ in their level of abstraction. For instance links between nodes can be represented with a precise geometry (road map) or as straight lines (airways). Networks can have different structures depending on the number of nodes and the layout of links between nodes. One possible distinction is that of *centripetal* networks (strong centrality, one node more connected than others) and *centrifugal* networks (no centrality, grid)[RCS06]. The former favors some locations over others. An example are many large airports, which serve as transport hubs. Figure 4.1 shows an example of the two network structures.

The structure of a network also directly influences its efficiency or network “flow”, that is how it can answer growing mobility demand.

4.2.1. Network topologies

A more formal view on the structure of networks is the one on network topology. It describes the relationship of network nodes in terms of connectivity, thus the general layout of the network. Again different types of networks can be distinguished, examples include linear, tree, mesh or hub structures (see figure 4.2).

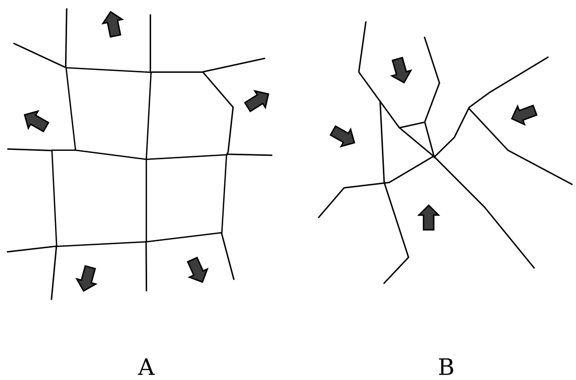


Figure 4.1.: Example of centrifugal (A) and centripetal (B) network structures.

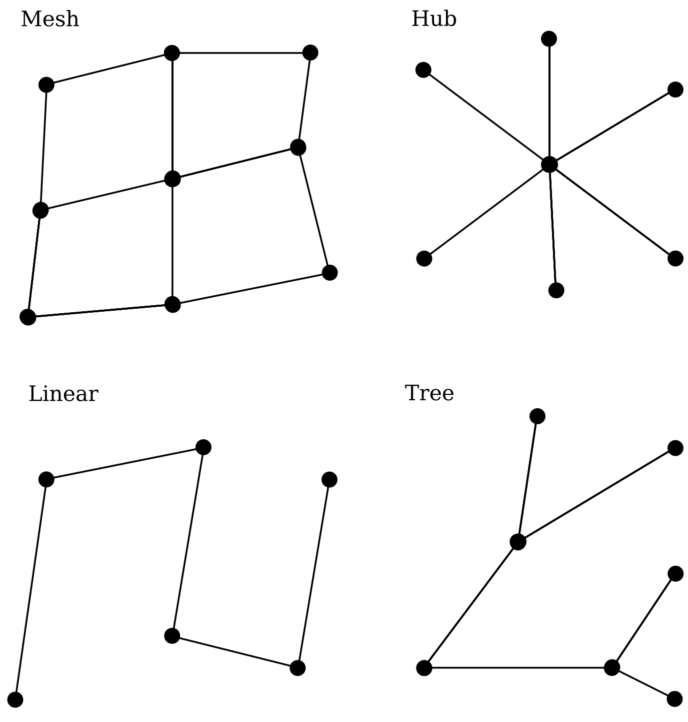


Figure 4.2.: Example of different network topologies.

In order to represent these networks in GIS, network data models are needed. Often graph theory is used for the modeling and analysis of transportation networks. The following section gives a short overview of its concepts.

4.2.2. Graph theory

Graph theory allows for the mathematical abstraction of networks with a set of properties, measures and indices [RCS06]. It is also the branch of mathematics and computer science concerned with research on trees, connectivity and graph algorithms.

In the following fundamental elements from graph theory are introduced, which are needed for an abstraction of transportation networks. This is a requirement for the formalization, representation and implementation of such a network.

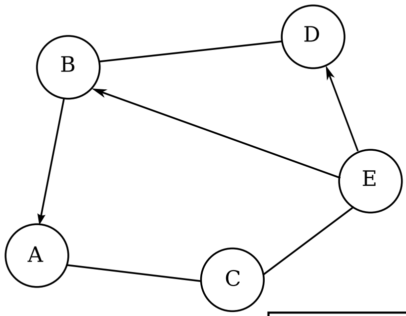
A *graph* is defined as a pair $G = (V, E)$ with elements of V called *vertices*(nodes) of G, and elements of E called *edges*(lines) of G [Die05]. Edges and vertices can be populated with attributes, for example addresses, classification or traffic volume in order to model different phenomena [Nor00]. A *weighted* graph is a graph where edge or vertice attributes are used as inputs to cost functions for graph traversals. A *path* is an ordered sequence of vertices written as $P = x_0 x_1 ... x_{k-1}$. If P is a graph then $C := P \cup x_k + x_k x_0$ is called a *cycle*. Graphs can be divided into *planar* and *non-planar* graphs. In the former every intersection of two edges is a node, whereas in the latter at least one intersection of two edges can be found, which is not a node. Non-planar graphs imply a higher dimension and can model more complex networks. A subset of a graph G is called a *subgraph*.

In order to model accessibility, possible movement over edges and vertices has to be described. Graph theory uses the term *connectivity*: a graph G is called connected, if any two of its vertices are linked by a path in G. A connected graph without a cycle is called a *tree* with a root node r. It is possible to specify the direction in which an edge can be traveled. Such a graph is called a *directed* graph. In order for two nodes to be accessible from each other the model must contain two directed edges originating at either of the two nodes. The term *network* is defined as a directed acyclic graph. Usually, however, the term is used more broadly.

4.2.3. Network data models

For encoding network models in a computer system, data models are needed that offer efficient storage and retrieval mechanisms. Transportation network models can contain large numbers of edges and vertices, therefore efficiency is important for applications that allow querying and displaying of network data.

One basic method is using nodes and links tables. The nodes table contains one entry for each node with a unique identifier and unique attributes in a chosen reference system. The links table contains: (i) a unique identifier for each edge, (ii) the node of origin and (iii) the node of destination. Figure 4.3 shows an example.



Nodes		
ID	X	Y
A	2	1
B	3	4
C	4	1
D	5	5
E	6	3

Links			
ID	From	To	$\longrightarrow$
1	A	C	0
2	B	A	1
3	B	D	0
4	E	B	1
5	E	D	1
6	C	E	0

Figure 4.3.: Example of a graph encoded with nodes and links tables.

This method stores the connectivity of the graph including the direction of its edges. Since this method is similar to the relational data model, it can be easily implemented in a relational database. Topological queries can be answered with a relational join over the two tables.

Another basic network model is using matrices, which can be encoded in computer systems using array structures. For a network with n nodes an adjacency matrix of size $n \times n$ can be constructed. If there is an edge going from n_i to n_j then $n_{ij} = 1$, otherwise $n_{ij} = 0$. This way the connectivity of a network is captured. The matrix model can be changed to a weighted graph version easily, by substituting the entry values of 1 or 0 with the actual weights.

For cartography, geocoding and census applications the layered-based approach, known from conventional GIS, is common. Using this approach, transportation network models are based on geographical elements such as lines, points and polygons. The most challenging aspect here is to find a common standard for the attributes and reference information, to make datasets interchangeable. One example is the TIGER (Topologically Integrated Geographic Encoding and Referencing) standard, developed by the U.S. Census Bureau to store nationwide street data for the United States.

4.3. Network algorithms

Network algorithms are used for storage and query operations on networks. This section briefly introduces some of the more common algorithms that will be used later.

4.3.1. Graph traversal

Graph traversal algorithms systematically visit all nodes of a graph and perform computations at each node [Pre01] along the way. Two categories are usually identified: *Breadth-first* traversal and *depth-first* traversal. The former refers to algorithms that systematically visit all nodes at one depth level, beginning at the root level and continuing downward. The latter algorithms visit nodes along the depth of the graph. In a tree the traversal starts with the root node – in a graph one node has to be chosen as the start node.

4.3.2. Shortest-path algorithms

Shortest-path algorithms find paths in graphs, based on weights attached to edges and nodes. Weights represent the cost of traveling over an edge or node. The algorithm tries to minimize the associated cost function in order to find an optimal trip. The cost metric depends on the application, one popular choice is absolute travel time, resulting in a path that has the shortest global travel time. For instance for a train travel information system an edge weighted graph $G = (V, E)$ can be constructed with the elements of V representing train stations and the elements of E representing train connections between stations. The weight of an edge $W(v_i, v_j)$ is then defined as the travel time to move from v_i to v_j. A shortest-path algorithm on G would return a path P with the smallest weighted path length defined by

$$\sum_{i=1}^{k-1} C(v_i, v_{i+1})$$

for a specified origin node v_1 to a destination node v_k.

Single-source shortest path problems

Most common in this context are single-source shortest path problems. The task is to find a shortest path, in terms of minimizing the cost function, from a given source node v_0 to a destination node v_j. Given an edge-weighted graph G the algorithm must calculate the shortest weighted path from v_0 to every other node in the network, aborting when the destination node is reached. The reason is that if the algorithm skips a node v_i in the computation process all paths from v_0 to the destination going over v_i are left out. These paths, however, might contain the optimal solution.

Dijktra's algorithm

Dijktra's algorithm is one example for a single-source shortest path algorithm [Dij59]. It is part of the class of *greedy* algorithms, which use the best currently known result to get from one state to another, also called local optimum [Wei97]. Algorithm 1 depicts Dijktra's algorithm in pseudocode notation. Given a weighted graph G the parameters of the algorithm are: a set of nodes V, a set of edges E, a cost function d and a start node s. The algorithm stores at each vertex v the cost of traveling from s to v along the shortest path as well as the vertex from which v was entered (parent). In the initialization phase the cost is set to infinity, reflecting the fact that no route is currently known between s and v. In every iteration the relaxation phase adds from all outgoing edges the edge with the lowest cost to the currently known shortest path. The algorithm maintains two lists of vertices: S containing vertices for which the cost along the shortest path from s is known and Q containing vertices for which that cost is unknown. In each iteration one vertex from Q is moved to S.

If iteration reaches a given destination node z the algorithms stops[1].

[1] This step is optional, but usually implemented, if we are only interested in the shortest path between s and z.

Algorithm 1 Dijktra's algorithm

function DIJKSTRA(V, E, w, s)
  for all $v \in V$ **do**
    $d(v) := \infty$, $parent(v) := null$ ▷ Initialization
    $d(s) := 0$, $S := \emptyset$, $Q := V$
  while Q *not empty* **do** ▷ Core algorithm
    $u := extract_m in(Q)$
    if $u = z$ **then** STOP ▷ Optional.
      $S := S \cup \{u\}$
      for all $(u, v) \in E$ outgoing from u **do**
        if $d(v) > d(u) + w(u, v)$ **then** ▷ Relax (u,v)
          $d(v) := d(u) + w(u, v)$
          $parent(v) := u$

Extensions and improvements

The original Dijktra's algorithm has been modified and extended for different applications areas. Sometimes it is desirable to calculate a ranked list of the next best solutions. K-shortest path algorithms have been designed to address this problem [Epp94]. The A* ("A star") algorithm introduces a heuristic estimate, which is used as part of the cost function in order to find the optimal solution [HNR72]. For instance in addition to the cost of traveling over an edge (u, v) the euclidean distance from v to the destination can be used as a decision criterion in the relax phase of the algorithm. If negative edge weights are required, for instance to model penalties for traveling over an edge, the Bellman-Ford algorithm represents a solution [Bel58].

4.4. Time geography concepts and networks

In order to apply the shared-ride trip planning strategies presented in the last two chapters to transportation networks, the network equivalents of time geography concepts are defined and then computational methods are investigated.

Space-time paths of individuals in networks are limited to movement along edges and nodes. Visualizations of these space-time paths look similar to paths, where movement in space is unrestricted. The projection of the space-time path onto geographic space maps to edges in the network. Space-time stations will coincide with

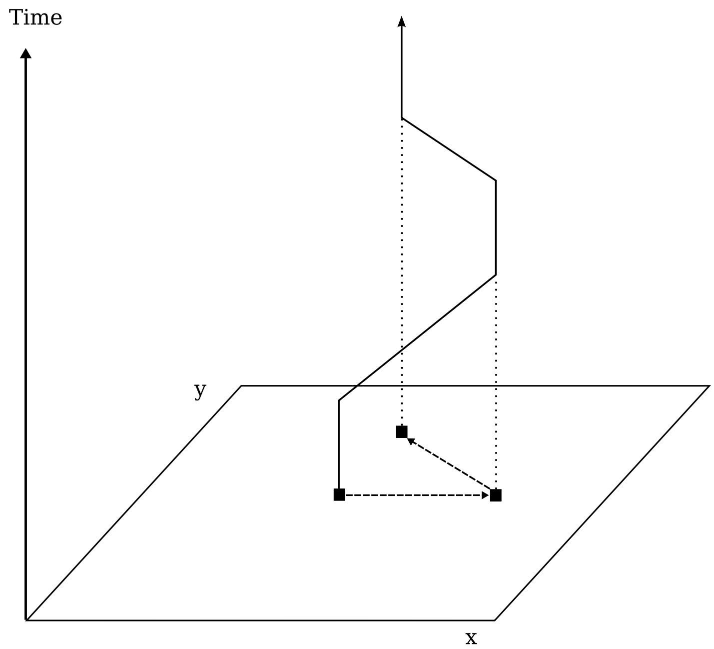


Figure 4.4.: Space-time path in a network.

locations along edges or node locations. Figure 4.4 shows an example of a network space-time path. Note that the geometry of the transportation network represents a space-time constraint for individuals using the network.

The impact of the network structure on the geometry of the STP is rather large. Since movement within a transportation network is limited by its geometry and the travel velocity might be different along each edge, even worse might change over time, the geometry of the STP in a network is not a cone, but rather irregularly shaped.

Miller [Mil91] developed methods and procedures for implementing network-based space-time prisms. The *network time prism* (NTP) consists of edges and nodes in transportation networks rather than a geometric set of locations. The *potential path tree* (PPT) is a subgraph of the network, which consists of edges and nodes reachable by an individual, given fixed activity locations and a time budget [YHW01]. For networks this represents a more realistic geometry of accessibility for individuals. Miller also developed methods for computation of the PPT and discusses implementation issues for query and visualization applications. Research was also done on incorporating cognitive constraints into the network-based space time prisms [KH98].

Capturing the dynamic aspects of transportation networks, such as time dependent travel times, temporal peaks and congestion, is an ongoing research effort, leading to increasingly realistic space-time accessibility measures.

For the calculation of the PPT, given a travel origin and a fixed destination location, Miller introduces a generic procedure containing two steps:

1. Calculation of shortest paths from the travel origin up to a cumulative impedance (travel time) along each path.

2. Testing, for each edge, whether traveling from the origin over the edge and to the destination is possible within the cumulative impedance.

This approach covers all properties of a network, including edge directions and turn costs. Relaxing the constraints on these properties, a simpler approach is introduced in section 4.5.3.2 for calculating the PPT.

4.5. Shared-ride trip planning in transportation networks

In the following, based on the previous work introduced in the last chapters, methods are suggested on how to solve ad-hoc shared-ride trip planning in a transportation network environment. This includes client and host communication and computation, as well as trip planning and assignment tasks. Algorithmic aspects are discussed in order to ensure the efficiency of the proposed network-based solutions.

For the following considerations imagine a shared-ride trip setting within an urban street network with cars and pedestrians. Car drivers are moving along the streets on a travel route from a source to a destination location. They are willing to offer free seats in their cars[2] to people looking for a ride, if they are not required to make a detour. Cars are equipped with geosensor devices. Travelers are pedestrians that look for a ride from their current location to a destination. The task of looking for corresponding offers has to be solved by their mobile devices.

[2]For means of simplicity unlimited seat capacity is assumed.

4.5.1. Communication

Wireless communication is used to exchange request, offer and booking messages. The network structure has no influence on communication[3] compared to a spatially unrestricted approach, therefore any of the communication strategies from section 2.4.2 can be used. For simplicity the unrestricted communication method is chosen.

4.5.2. Trip assignment strategies

Travelers are called client agents, which look for, plan and book a shared-ride. Car drivers are called host agents, which move according to their own travel plan that is not predictable. The task is to find the host with the best offer to a client's request. There are numerous trip assignment strategies possible, the following three strategies are investigated further.

4.5.2.1. Shortest, complete route with hosts inside dl-subtree (DL_d-strategy)

The client has to formulate a travel request. This request contains three parameters: a start location, a destination location and a latest arrival time. Since the client makes use of a geosensor the first parameter can be directly obtained from a location detecting device, such as a GPS sensor. The destination location must be identified by user input, thus a model of the transportation network must be known to the client. Once the three parameters are known, the request message can be encoded and broadcasted by the client. Eventually, the request message reaches all hosts in the network, because an unrestricted communication strategy is implemented. Hosts receiving the request have to decide if their future travel plan contributes to the client. At first the host decides whether the current client position is part of its future travel route. If this is not the case, no offer is created since hosts are not willing to make detours. In a second step the DL_d-strategy applies the concept of the dl-cone (see section 3.2) as a criterion for the host's decision process. In the following paragraph the network equivalent of the dl-cone is defined.

Subtree of latest arrival time Similar to the concept of the cone of latest arrival time a subtree of latest arrival time can be defined for networks, as the PPT con-

[3] Since the network structure has an influence on the distribution of hosts (for instance clustering at junctions), it is assumed that this factor has an impact of some sort on communication. However, this is not discussed here and left to future work.

taining all nodes in the network from which the client destination can be reached, within the time budget defined by the latest arrival time and movement along the shortest path towards the destination. This subtree is called *dl*-subtree.

Using the client's destination location and the latest arrival time, the host can calculate the *dl*-subtree and decide whether its current location is part of the *dl*-subtree or not. Only hosts with locations inside the *dl*-subtree will make an offer, because they have a chance to pick up the client and reach the destination within the time budget. The client maintains a list of received offers and chooses the optimal one. In this strategy it is defined as the offer that reaches the destination location earliest. Once the client received a matching offer it creates and broadcasts a booking message.

4.5.2.2. Shortest, complete route, hosts with travel path inside client's NTP (NTP_d-strategy)

The second strategy introduced in this section is similar to the first, except that the concept of a space-time prism of the client's travel possibilities as defined by time geography is used. Hosts use this as a criterion whether to make an offer or not.

Subtree of earliest arrival time In analogy to the *se*-cone, the subtree of earliest arrival time can be defined, as the PPT containing all nodes that can be reached from the start location and within the time budget by moving along the shortest-path at maximum travel speed.

Network time prism of client's travel possibilities The spatially unrestricted approach results in a space-time prism defined by the intersection of the *se*-cone and the *dl*-cone. The network equivalent is a network time prism, defined by the subgraph that contains all nodes that, at the same time, are part of the *dl*-subtree and the *se*-subtree.

Using the client's current and destination location, as well as the latest arrival time, the host can calculate the client's NTP and decide whether its future travel route is completely or partially part of the NTP. Using the NTP_d-strategy only hosts that have a continuous travel route that is part of the NTP and reach the client destination make an offer. The client chooses the best offer similar to the first strategy.

4.5.2.3. Shortest, partial route, hosts with travel path inside client's NTP (NTP_s-strategy)

In [WR06] a strategy was suggested, where the client only looks for parts of the route ahead. The third strategy applies this concept by using the NTP. A host calculates the client's NTP similar to the second strategy. However, hosts also make an offer if their future travel route is only partially part of the NTP. Also it is not demanded that this route includes the client's travel destination.

The client collects all offers and decides to book a trip even if there is no trip that reaches the destination. Among these offers the client chooses the one that gets closest to the final destination, by looking for the node in the host's route with the smallest euclidean distance to the final destination. At this node the client chooses to be dropped off and continues with the search for the next optimal offer.

Using this strategy is a risky decision for the client: She might get a ride quickly, and the ride might also get her closer to the final destination, but chances are that she will reach the destination late or never.

4.5.3. Algorithms and data structures

This section defines algorithms and data structures needed for implementing the introduced trip assignment strategies for shared-ride trip planning scenarios. Proposals on data types are based on widely used *abstract data types* (ADTs).[4] Algorithmic complexity is discussed using common notations from computational complexity theory. The following section introduces some of the concepts briefly.

4.5.3.1. Computational complexity

When dealing with algorithms and data structures the question of efficiency is important: How many resources are needed to solve a given problem? Computational resources are time (number of steps or iterations an algorithm requires) and space (memory consumption). Whether a problem can be solved by computation at all is a different issue, covered by computability theory. Decision problems can be divided into classes of similar complexity. For instance the complexity class P contains problems that can be solved in polynomial time by using a deterministic machine.

The time complexity for a given algorithm is a function of the size of its input. The

[4] Details can be found in [EM85].

value not only depends on the number of input elements but also on the language or machine used to implement the algorithm. For algorithmic complexity to be comparable, asymptotic notations are used that abstract from concrete measurements. These mathematical notations define an asymptotic upper bound to functions with increasing size of input.

Big O notation The "Big O notation" (order of) is an example of an asymptotic notation used frequently for describing the efficiency of algorithms [CLRS01]. It is formally defined as:

$$O(g) := \{f : N_0 \rightarrow R^+ \mid \exists c \in R^+ \, \exists n_0 \in N_0 \, \forall n \geq n_0 : f(n) \leq c \cdot g(n)\}$$

The set $O(g)$ contains all functions that grow at most as fast as g. If a given function f satisfies the definition, it is written as $f \in O(g)$. Breaking algorithms down to the basic functions they perform, common classes of algorithm complexity can be defined. Examples are constant $O(1)$, logarithmic $O(\log n)$, linear $O(n)$ and quadratic complexity $O(n^2)$.

4.5.3.2. Calculation of the NTP

For all strategies the *dl*-subtree, *se*-subtree or the network time prism have to calculated. Using the same parameters as for the *dl*-cone, a *dl*-subtree can be constructed by running a shortest-path algorithm originating at the destination location that stores the cumulative travel times along the shortest path in each node. The algorithm discards subtrees from nodes where the cumulative travel time exceeds the upper time limit (latest arrival time). This solution assumes that (i) travel along an edge must be possible in both directions for the same cost and (ii) turn costs are not modeled. The reason is that the client will travel to the destination in the opposite direction.

For the second and third strategy, the *se*-cone must be constructed. This can be done by using a shortest-path algorithm originating at the current client location, iterating up to the travel time limit set by the client's request.

Using both subtrees the network time prism of the client's travel possibilities can be constructed by calculating the intersection of both trees, defined by all nodes that are part of the *dl*-subtree and the *se*-subtree. Figure 4.5 summarizes the construction process: (1) a *dl*-subtree originating at the destination location starts to grow, (2) a growing *se*-subtree is added and (3) the two subtrees overlap creating

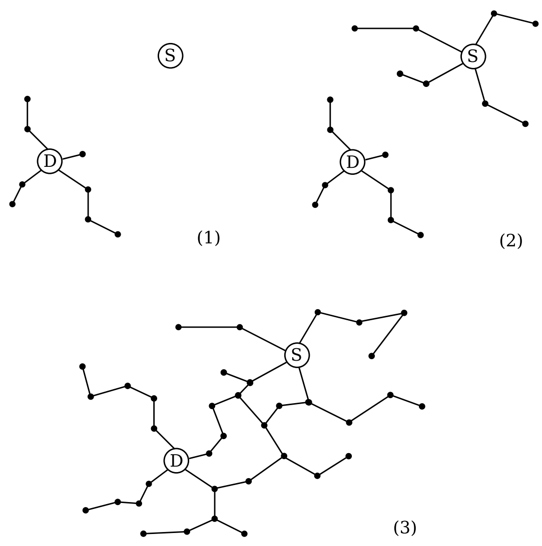


Figure 4.5.: NTP construction process.

the network time prism. Only if the chosen upper time limit is sufficient, an intersection can be found. Construction of the network time prism consists of running an impedance-aware shortest-path algorithm twice. The time complexity of building the NTP depends on the shortest-path implementation. For a graph with n nodes and m edges Dijktra's algorithm using a Fibonacci heap implementation has a logarithmic average time complexity of $O(n \cdot logn + m)$. Since the constant factor of 2 has no influence on the complexity class, the average time complexity of the NTP buildup algorithm is logarithmic. The NTP can be stored using a heap-based structure, because it is used for individual node lookup only, therefore the topology of the network must not be preserved.

4.5.3.3. Host decision algorithm

Hosts have to store their own travel route. This can be done with a stack ADT that has entries for each node to be visited. At the beginning of the trip the host start a shortest-path algorithm with a given start and destination location, thereby defining the travel route. Successive nodes can be obtained from the stack within constant time complexity of $O(1)$. Time complexity of the initial buildup of the route stack depends on the shortest-path implementation.

Upon receiving a client's request hosts have to decide if they can make an offer or not. Algorithm 2 shows the algorithm based on the previous considerations in pseudocode notation for the NTP_d- and the NTP_s-strategy. The input parameters are: the client request (*request*), a cache of requests already received (*cache*), the host's travel route stack (*route*) and the calculated network time prism (*ntp*). At first duplicate requests are eliminated in order to avoid re-sending offers (1). Assuming a hash-based implementation of the request cache, lookup can be done within constant time complexity of $O(1)$. In (2) the offer is initialized as an empty stack together with two boolean helper variables. The *ntp* variable is initialized with a calculation of the network time prism based on the request parameters. Then iteration starts by looking at one node of the future route at a time (3). If the client start or destination location are found, the helper variables are set accordingly. Once the start location has been found the currently visited node is added to the offer stack if it is part of *ntp*. An empty stack indicates that the host's travel plan does not include the client start location, thus no offer is created (4). For the NTP_d-strategy the optional statement (5) is used to exclude hosts with travel routes not containing the destination location.

For the DL_d-strategy the host decision algorithm works slightly different. Algorithm 3 shows the adapted algorithm. For this strategy the parameter *ntp* is substituted with *ppt*, which contains only the *dl*-subtree. The host route does not have to include the final destination, filtering is done on the client side, therefore no checking on the destination is done. If the host's current location is not part of the *dl*-subtree, no offer is created (1), otherwise the host offer contains a route starting in the client's current location.

The time complexity of the host decision algorithms can be found by considering the following parts: (1) calculation of the NTP and PPT have logarithmic average time complexity of $O(n \cdot \log n + m)$, (2) cache lookup and stack operations have constant time complexity of $O(1)$ and (3) search in the NTP and PPT have logarithmic average time complexity of $O(\log n)$ when using a heap-based implementation. The cost of iterating over the route has linear time complexity of $O(r)$. With growing network size, however, the input size r (number of nodes in the host route) will be much smaller on average than the input size n and m (nodes and edges in the graph) and can therefore be neglected in the analysis of the overall time complexity. Since

$$O(n \cdot \log n + m) + O(1) + O(\log n) \in O(n \cdot \log n + m)$$

is true, the host decision algorithms have total linearithmic average time complexity of $O(n + \log n + m)$.

Algorithm 2 Host decision algorithm (for strategies NTP_d and NTP_s)

input: client request, request cache, host route

output: host offer

function DECIDE_REQUEST($request, cache, route$)
  if $cache$ contains $request$ **then** STOP ▷ 1
  $offer$ = empty stack; ▷ 2
  $foundStart$ = false;
  $foundDestination$ = false;
  ntp = CALCULATE_NTP($request$)
  while $route$ not empty **do** ▷ 3
    $node$ = pop from $route$;
    if $node$ == client start location **then** $foundStart$ = true;
    if $node$ == destination location **then** $foundDestination$ = true;
    if $foundStart$ == true **then**
      if ntp contains $node$ **then** $offer$ push($node$);
      else break;
  if $offer$ == empty stack **then** STOP ▷ 4
  if $foundDestination$ == false **then** STOP ▷ 5
  return $offer$

Algorithm 3 Host decision algorithm (for strategy DL_d)

input: client request, request cache, host route
output: host offer

```
function DECIDE_REQUEST(request, cache, route)
    if cache contains request then STOP
    offer = empty stack;
    foundStart = false;
    ppt = CALCULATE_PPT(request)
    if host current location is not inside ppt then STOP          ▷ 1
    while route not empty do
        node = pop from route;
        if node == client start location then foundStart = true;
        if foundStart == true then offer push(node);
    if offer == empty stack then STOP
    return offer
```

4.5.3.4. Client choice algorithm

The client has to be able to receive, store and evaluate offers received from nearby hosts. Storage is necessary in order to identify duplicate offers and to select, at any point in time, the best of all current offers. The client is interested in a trip minimizing the total cost of getting from the start to the destination location. This decision depends on the strategy chosen, for instance preferring the trip that will reach the destination earliest. An offer contains a route (list of nodes) and arrival timestamps for each node. This way the client can easily identify the most valuable trip. Algorithm 4 shows the client choice algorithm in pseudocode notation for the strategies NTP_d and DL_d.

For the list of known offers a priority queue ADT is used with priority based on hosts minimizing the cost function. Retrieval of the best host offer (2) can then be done with constant time complexity of $O(1)$. For the DL_d-strategy the optional step (1) checks whether the destination location is part of the offer. The newly received offer is compared to the best currently known offer (3) and $offerQueue$ is updated if necessary. Time complexity for the update operation depends on the implementation of the priority queue – using a balanced tree the insert, update and remove operations have an average and worst-case time complexity of $O(\log n)$ [Knu97].

Algorithm 4 Client choice algorithm (for strategies NTP_d and DL_d)

input: host offer, current offer queue
output: -

```
function RECEIVE_OFFER(newOffer, offerQueue)
    if newOffer not includes destination then STOP                ▷ 1
    bestOffer = dequeue from offerQueue;                          ▷ 2
    if newOffer is quicker than bestOffer then
        enqueue(newOffer) in offerQueue at top;                   ▷ 3
```

The NTP_s–strategy requires the client to choose from the received offers, the one that contains a node closest to the final destination. For this the euclidean distance to the destination location is considered. Algorithm 5 (page 38) shows the algorithm in pseudocode notation for this strategy.

Two priority queues are used: one for offers that contain the destination location ($cOfferQueue$) and one for offers that contain only partial routes ($pOfferQueue$). The client chooses the best offer from the complete offer queue – if the queue is empty the best from the partial offer queue is chosen. At first newly received complete offers are handled and compared to the currently known best offer from $cOfferQueue$ (1). If the received offer contains only a partial route, the drop off location from the currently known best partial offer is extracted (2). The algorithm then iterates over all nodes from the received partial offer's route (3) and compares the distance of the node to the destination location with the best known drop off location (4). If a closer location was found $pQueue$ is updated accordingly (5). Two new operations have to be considered for evaluating time complexity: the iteration over all nodes from the offer and the distance calculation. The latter has a time complexity of $O(1)$, the former of $O(n)$. Algorithm 5 has logarithmic total average time complexity of $O(\log n)$ since

$$O(1) + O(\log n) \in O(\log n)$$

is true. Total average time complexity of Algorithm 5 is linear, because with the two additional operations

$$O(1) + O(\log n) + O(n) \in O(n)$$

is true. Because the input size n of the client choice algorithms refers to the number of nodes in the offer, which will be much smaller on average compared to the total

Algorithm 5 Client choice algorithm (for strategy NTP_s)

input: host offer, complete offer queue (cOfferQueue) and
partial offer queue (pOfferQueue)
output: -

```
function RECEIVE_OFFER(newOffer, cOfferQueue, pOfferQueue)
    if newOffer includes destination then
        bestOffer = dequeue from cOfferQueue;
        if newOffer is quicker than o then                                ▷ 1
            enqueue(newOffer) in cOfferQueue at top;
        STOP;
    bestOffer = dequeue from pQueue;
    currentDistance = DISTANCE(bestOffer, destination)                    ▷ 2
    dropOff = null;
    for all nodes n ∈ newOffer do                                         ▷ 3
        if DISTANCE(n, destination)< currentDistance then                 ▷ 4
            dropOff = n;
            currentDistance = DISTANCE(n, destination);
    if dropOff is not null then enqueue(newOffer) in pQueue;              ▷ 5
```

number of nodes in the network, the average time complexity of the client's calculations is less relevant than the host decision algorithms. These respond to increasing network size, whereas the former respond to increasing trip length of the offer.

4.6. Summary

In this chapter concepts from time geography and transportation networks have been applied to shared-ride trip planning scenarios in networks. Trip assignment strategies have been proposed together with algorithms and data structures for implementing these strategies.

5. Application: Design and implementation of a simulation environment

This chapter describes the design and implementation of a simulation environment, which is used to apply the network-based strategies for ad-hoc shared-ride trip planning to a real street network. At first geosimulation and automata are introduced, followed by a description of the design and implementation of the `P2PSim` application.

5.1. Geosimulation

Geosimulation is a term for simulation modeling of spatial phenomena using concepts from computer science and geographic information science. Geosimulation can be used to represent complex, adaptive and dynamic systems, for instance urban environments [BT05]. While having much in common with traditional simulation approaches (microsimulation, cellular automata, agent-based simulation) geosimulation tries to explicitly capture the characteristics of individual spatial units and spatial relationships. Geosimulation defines concepts for realtime, high-scale simulations of complex geospatial phenomena.

Historically there has been a disappointment with classical urban simulation from the 60's and 70's that resulted in a dismissal of these technologies in urban planning. Recent geosimulation models are based on spatially related automata and implemented using object-oriented paradigms. Elements of urban systems, such as humans or infrastructure, are modeled as objects with properties and behavior.

The advance of high resolution GIS and data mining techniques made the success of geosimulation possible and its application popular [Atk99]. Spatial analysis and remote sensing data can be used as an input for geosimulation environments. Geosimulation has its foundations in mathematics, going back to the idea of automata developed by Alan Turing, as well as universal computation theories, and in computer science, with its object oriented paradigms, which can be used to represent entities together with their properties and behavior.

Geosimulation can be defined as the modeling of “adaptive collectives of interacting entities” [BT05]. This is called the *generative* (or bottom-up) approach, compared to the *reduction* approach of dissolving phenomena into entities in a top-down manner. It is suitable for representing self-organizing and emergent systems such as urban environments or transportation systems.

In order to describe the behavior of entities, automata can be used. This is a common concept in many simulation solutions. The following section introduces the two most popular types: cellular automata and agent-based systems. A new approach of geographic automata systems is described and the section finishes with a comparison of the three approaches.

5.1.1. Cellular automata

Cellular automata (CA) are based on a discrete model. This model contains a regular grid of cells and each cell has an internal state [Wol84]. The grid can be of any finite number of dimensions. For spatial simulations a two-dimensional version with finite boundaries is often chosen for means of simplicity. For simulation runs each cell's state is updated in discrete time intervals, depending on the state of its neighborhood cells. Different neighborhood configurations are possible. Figure 5.1 shows two examples: a 5 x 5 ”von Neumann”-neighborhood (A) and a 3 x 3 ”Moore”-neighborhood (B). Cellular automata can be used to simulate systems, in which the behavior of elements can be described by rules.

One popular example of the use of cellular automata is John Horton Conway's model of ”Life” that was used to study the spatial growth of populations [Gar71]. Using a simple set of rules, a cell's state is set to alive, dead or reincarnated, depending on the number of living neighbors within a 3x3 Moore neighborhood:

1. *Survival*: If a cell has exactly two or three live neighbors it stays alive.
2. *Birth*: Once a dead cell has exactly three live neighbors it becomes alive again.
3. *Death*: In all other cases, a cell dies due to overcrowding or loneliness.

Using these simple rules numerous patterns and variations emerge that inspired the development of many applications to study these patterns and made the the

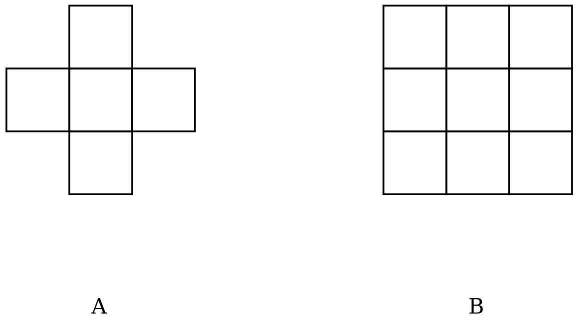


Figure 5.1.: Examples of CA neighborhoods.

concept of cellular automata popular. In a geographic context CA are used in the study of land-use dynamics, planning and urban systems.

5.1.2. Multi-agent based simulation

Multi-agent systems (MAS) describe systems as a combination of multiple autonomous and independent *agents*. Agents extend the representation of automata by adding *behavior* that governs the internal state change [Mae95]. In computer science the term agent is used as an abstraction, describing a complex entity that is able to act autonomously, without supervision, in order to perform tasks on behalf of the user [Woo02]. Categories like intelligent, distributed and mobile agents exist. The agent-based modeling approach is popular in the areas of economics, sociology and ecology [Tes06, QS03].

MAS are of interest to geosimulation due to their ability to reflect human behavior [BT05]. For instance a household, car drivers or individuals using public transportation can be modeled as agents with an internal state and transition rules that drive their actions. Using this concept self-organizing and complex urban systems can be studied well with MAS. Examples include the study of pedestrian's walking paths, cars in urban traffic and city development.

5.1.3. Geographic automata systems

Benenson and Torrens [BT05] suggest a new approach of formalizing urban systems using automata. Their approach states the following requirements for a geographic modeling framework that captures the spatial properties of objects:

- Capturing a typology of entities.
- Capturing of space and spatial relationships.
- The change of spatial properties over time.

This distinguishes their approach from non-spatial simulation systems like the former two and "spatially enables" geosimulation. By formalizing these requirements a geographic automata system is formally defined as

$$G \sim (K; S, T_S; L, M_L; N, R_N)$$

K is a set of types or ontologies. The authors distinguish between *fixed* types, for example buildings, and *non-fixed* types, such as moving objects. S is a set of automata states with T_S representing the set of transition rules (as for traditional automata). L defines a set of georeferencing rules that define the geographic location of automata within the system. M_L represents the movement transition rules in the system. The type of georeferencing depends on the application: direct representation (exact geometry), indirect representation (contours, nodes) and also representation as links to other automata are possible. Finally N represents the neighborhood of automata, with R_N describing neighborhood rules. In contrast to cellular automata neighborhood relationships can vary in space and over time. The concept of neighborhood in geographic automata systems covers concepts known from traditional GIS, such as connectivity or proximity.

5.1.4. Comparison

The relatively new approach of geographic automata systems has a high potential for geosimulation as it offers methods to incorporate the spatial component with all its consequences. Since this represents a relatively novel approach and implementations or frameworks using these concepts have not yet been developed, this thesis focuses on using established frameworks to simulate an urban environment. It will be attempted, however, to incorporate as many of these concepts as possible. The cellular automata approach could be easily implemented, because of its simple structure and rules. However, the approach is too static for the modeling of moving nodes and network structures. The MAS approach seems to be the most promising solution for modeling the dynamics of shared-ride trip planning scenarios. Clients and hosts can be modeled as agents with their own identity and behavior. It is therefore preferred for implementing the simulation environment.

5.2. P2PSim: A software for simulating shared-ride trip planning scenarios

`P2PSim` is a software for simulating shared-ride trip planning scenarios in transportation networks. It is developed to assist in proving the hypothesis of this thesis by simulation. The following section illustrates the development process by describing the requirement analysis, design and implementation.

For the development of a software product a *process model* is chosen, which defines development phases. Usually the phases contain iterative processes, while more functionality is added and overall quality is improved [Bal96]. There are various established process models available, most of them tailored to commercial software development[1] `P2PSim`, however, is designed to be a "proof-of-concept" simulation, used in a scientific environment with a very focused purpose and limited target audience. Therefore a much simpler process model can be used, similar to that of a *prototype* application. Many of the non-functional requirements can be relaxed and a formal analysis step is skipped.

5.2.1. Design

The software is used to simulate shared-ride trip planning scenarios with one client requesting a ride and multiple hosts moving in a transportation network, possibly offering transportation supply. The following requirements can be directly inferred from shared-ride trip scenarios:

- Communication and computation of hosts and the client within a network must be simulated by the software.

[1]Examples include the Rational Unified Process(RUP), the V-Model and Extreme Programming (XP).

- The application must support an import of GIS data (e.g. ESRI shapefile format), creation and internal representation of this data as a consistent network structure and representation of client and host movement along the network structure.
- Realtime visualization of the network structure and moving peers must be possible, including the ability to change the field of view.
- Parameters, such as the number of hosts or the travel assignment strategy, must be adjustable for different simulation runs.
- Recording and analysis of measurement variables over time must be possible.

Since a prototype process model is chosen for development, the requirements will be directly mapped into a product model. This will not start from scratch, because for basic functionality frameworks will be used. The goal of the design phase is to develop a software architecture and a specification of system components [Oes01].

5.2.1.1. Loading, displaying and querying of spatial data

Basic GIS functionality is needed in order to work with real spatial data sets and create networks from these data. An optimal solution offers the ability to import GIS data in a standard format (e.g. ESRI shapefile) and allows for conversion of this data into an in-memory network structure. There is a selection of open-source frameworks available that offer this functionality, most of them are designed in an object-oriented manner and implemented using the Java programming language. The geotools framework is chosen as a popular open-source Java-based GIS toolkit[2]. The geotools project focuses on standards compliant[3] implementations of a wide range of GIS features, including data stores, feature manipulation, as well as transformation and visualization of spatial data. The framework is actively developed, strictly object-oriented and for most parts well documented.

5.2.1.2. RepastJ

Since the MAS approach was chosen for simulating shared-ride trip planning scenarios it is necessary to find a multi-agent based framework that meets the requirements

[2] http://www.geotools.org
[3] Most of these standards are defined by the Open Geospatial Consortium (OGC).

of P2PSim. Again many open-source toolkits are available. The Recursive Porous Agent Simulation Toolkit (Repast) is an agent-based simulation toolkit. It has been written from scratch in 2003 at the University of Chicago, based on established concepts from the SWARM toolkit [MBLA96] and is still actively developed [NCV06, CHN03]. It differs from SWARM, because it is implemented in several programming languages and has distinctive features like genetic algorithms and regression. The SWARM framework uses Objective-C as its primary programming language. A Java port is available, however, documentation is sparse. Repast has an architecture that makes it easy to develop new simulations environments. More complex functionality can be added step-by-step and many example applications are available.

Therefore the choice is made to use the geotools framework for basic GIS data handling and the RepastJ framework for multi-agent modeling. This demands for the product model of P2PSim to be object-oriented and adapted as much as possible to the established interfaces of these frameworks. The Unified Modeling Language (UML), an industry standard for software design, is used to capture the model. Figure 5.2 shows the P2PSim model as a UML class diagram.

The model is divided into two packages. Classes from the model package represent a mapping of the entities and operations involved in shared-ride trip planning, namely hosts, clients and communication methods. Two interfaces are defined: P2PClient and P2PHost. They specify methods for exchanging offers and requests and for providing information on the current position and travel route. Both interfaces inherit from the P2PNode interface, which contains selected common methods. The main package is a collection of classes that handle network data, map display and simulation control. The NetworkHandler class provides methods for building the network representation from an input file, creating random client and host routes and calculating potential path trees. The responsibilities of the NetworkDisplay class include the display and update of various map layers. P2PNetworkAgentsModel is the main class and called upon start of the application. It implements simulation control methods inherited from the SimModelImpl class and data recording methods. SimModelImpl is a template class provided by the RepastJ framework for creating new simulation applications. Deriving a fully specified architecture from this point on is straightforward in this case. Detailed information can be found in the appendix that contains the source code documentation of P2PSim.

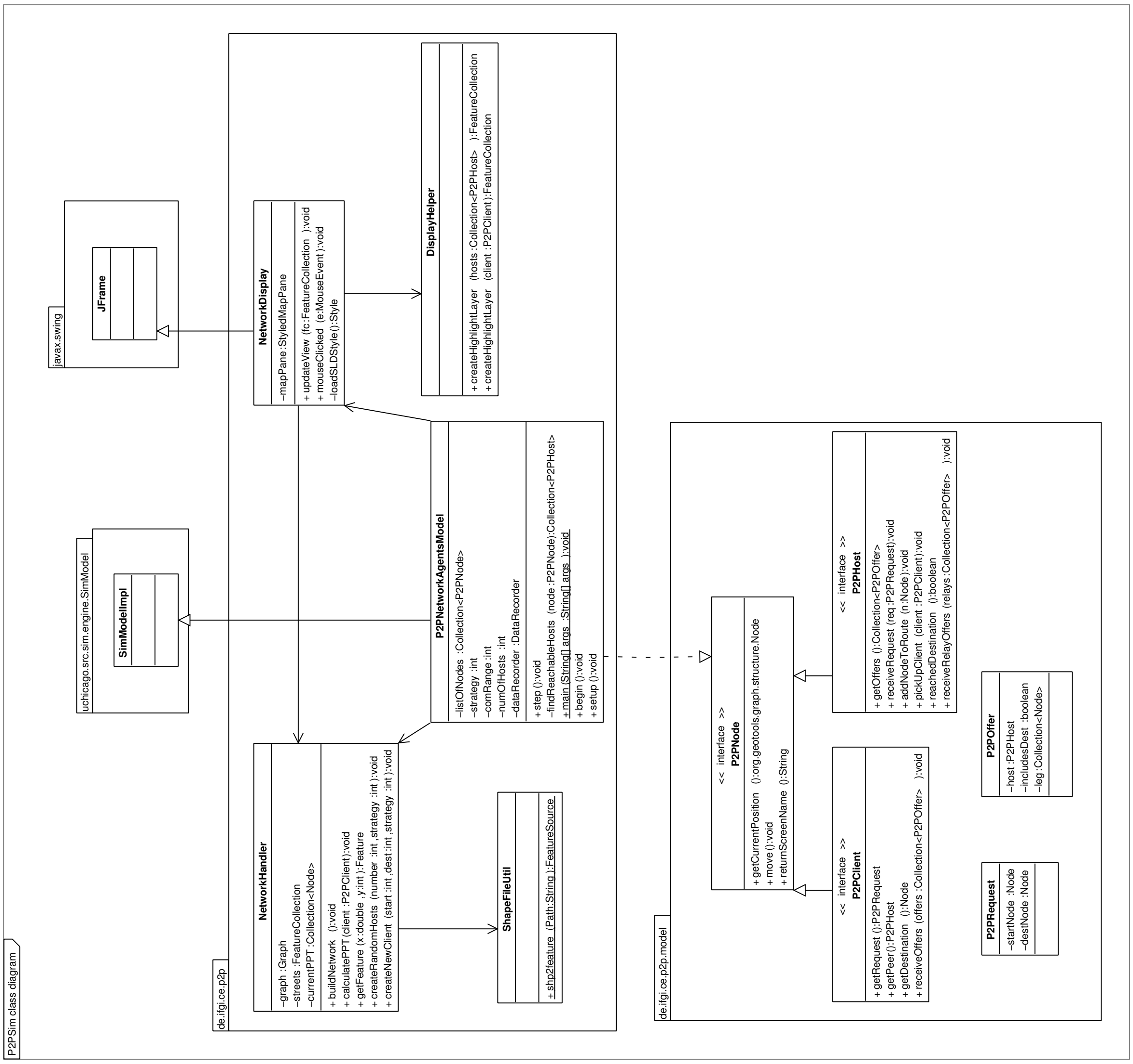


Figure 5.2.: UML class diagram of the `P2PSim` model.

5.2.2. Implementation

Implementation means realizing the system architecture with a programming language. This was done using `Java`, an object-oriented programming language developed by `SUN`, and `IDEA`, a commercial integrated development environment (IDE). The most challenging aspects of the implementation work were the consistent representation of the network structure, the integration with the existing frameworks and performance optimization of the network visualization. Testing was done by using the open-source testing framework `JUnit` and implementing test cases for the main functionality.

5.2.3. Simulation runs

In total 400 simulation runs with varying parameters were recorded. As a data basis a street network of the city of Münster (Germany) was used. The data was available from a `ESRI` shapefile containing polylines, that were manually digitized, using a TK-50 airphoto of the city. `P2PSim` was used to load the shapefile and convert it into an in-memory network representation, to be used for simulation. The network contains 127,264 nodes and 160,803 edges. Figure 5.3 shows a screenshot of a simulation run. Parameters that can be adjusted in the parameters toolbar window are: number of hosts, travel assignment strategy and minimum communication range. It is possible to start multiple simulation runs for unsupervised data collection. The simulation finishes when the client reaches its destination or when all hosts have reached their destination. A trip was successful if the client reaches the destination within a specified time budget. Afterwards collected data is made available as text or graphics for analysis. More information and a manual to the `P2PSim` software can be found in the appendix.

5.2.4. Travel assignment strategies

For the simulation runs, the client is placed at a location in the city center with a destination location on the inner street circle. This choice represents a realistic medium-length trip offering two main alternatives to reach the destination: one going northward, the other going southward. Different numbers of hosts are created with randomly chosen start and destination locations. The trips are created such that host routes fall into one of three possible categories:

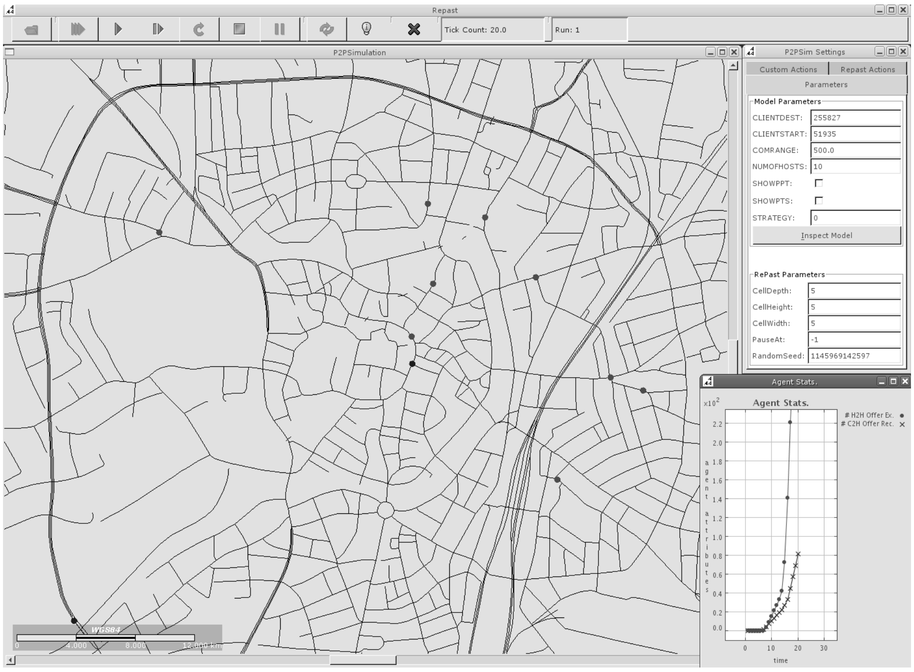


Figure 5.3.: Screenshot of a simulation run.

- *All random route*: a route is chosen randomly.
- *Random start and destination location*: start and destination location are chosen randomly, but the route includes the current position of the client.
- *Random start location*: starting from a random location, the host route includes the current client position and the final destination of the client.

These choices were made in order to give the client a chance to be picked up by a host, as well as keeping the simulation realistic. The time budget was set to walking time to the destination plus an arbitrary constant value. For the test runs, the theoretically derived strategies from section 4.5.2 have been implemented, as well as a *Base*-strategy for comparison:

1. *Base*-strategy: Hosts implement no special decision algorithm. If a host receives an offer from the client an offer is made and sent back to the client. Messages are rebroadcasted without restrictions. The client collects all travel plans and prefers the quickest route to the destination. Once a matching offer is received the trip is booked. This strategy is implemented for comparing the results of a trivial assignment strategy to the ones proposed in section 4.5.2. For this

strategy the following probabilities for creating host routes are used: 20% all random route, 60% random start and destination location and 20% random start location, but at least one from the latter category. It is expected that the *Base*-strategy leads to a high number of offers being created, as well as a quick increase in the exchanged messages between the client and hosts and between hosts, because no filtering techniques are applied.

2. NTP_d: This strategy uses the NTP of the client's travel possibilities as a criterion for hosts whether to make an offer or not (4.5.2.2). The same probabilities as in (1) are used for creating host routes. Compared to the *Base*-strategy this strategy is expected to greatly reduce the number of offers created by hosts and overall exchange of messages, because most of the hosts (except those with a travel plan inside the NTP including the final destination) will not create an offer.

3. DL_d: As a criterion for hosts the *dl*-subtree is used in this strategy (4.5.2.1). The probabilities for host route creation are the same as in (1). Filtering of host offers is reduced compared to (2), because only those hosts with positions outside the *dl*-subtree are excluded. The level of the number of offers created and overall communication are expected to lie between (1) and (2).

4. NTP_s: This strategy differs from the NTP_d-strategy because hosts also make offers for partial routes. The client chooses the one that gets closest to the destination (4.5.2.3). For simulation runs using this strategy the probabilities of host creation are adjusted slightly. Since the client uses multiple stops on her travel to the final destination, a new set of random hosts is generated after the client completes a partial trip. After n runs the probability of a host route to reach the destination is increased with each run. This way the client has a chance to eventually reach the destination after a few runs. It is expected that the overall trip length using this strategy is increased since the client most likely does not travel on the shortest-path route, but rather takes detours. Therefore the overall number of messages exchanged is expected to be higher. Since more hosts have routes that are valuable to the client, the number of offers created is also expected to be higher compared to (2) and (3).

The results and discussion of the simulation test runs are presented in the following chapter.

6. Results

In this chapter the results of the simulation runs are presented and discussed.

6.1. Simulation results

For the 400 simulation runs the *Münster* dataset was used with the following parameters:

- Communication range: 500 meters.
- Trip assignment strategies: $Base$, NTP_d, NTP_s and DL_d (100 runs with each).
- Number of hosts: 10, 20, 30, 40 and 50 (20 runs with each).

In order to evaluate communication and computation effort the following variables have been recorded:

- Client-to-Host(C2H) messages: offers and requests that were exchanged directly between the client and hosts.
- Host-to-Host(H2H) messages: all messages exchanged between hosts, including messages that have been rebroadcasted.
- Unique Host-to-Host messages: count of all unique messages received by hosts from other hosts. Unique refers to messages that had not been received by the host before.
- Offers: count of all newly created offers by hosts.
- Computation time: total time hosts spent running the respective host decision algorithm in milliseconds (ms).
- Time: the time for each run in seconds.

Table 6.1.: Average number of exchanged messages.

Strategy	C2H (Avg).	H2H (Avg.)	H2H Unique (Avg.)
Base_d	14,575	757,768	692
NTP_d	2,395	114,154	119
NTP_s	22,686	437,834	366
DL_d	2,678	129,409	139

- Success: a boolean variable indicating, whether the run was successful or not.

Simulation was recorded in *runs*, consisting of constant time intervals (*ticks*) of 1 second. Data was recorded using two `Repast DataRecorder` instances: one for summarizing the variables for each run, the other for recording timeline data for each tick per run. The measurements have then been aggregated using a custom `Perl` script.

6.1.1. Number of messages exchanged

The number of messages exchanged between client and hosts have been recorded and aggregated. Table 6.1 shows the average number of client-to-host (C2H), host-to-host (H2H) and unique host-to-host (H2H Unique) messages measured over all runs. The majority of messages fall into the category H2H. Less than 1% of these messages were unique host-to-host messages. Only a small part of the messages exchanged fall into the C2H category. The use of the *Base*-strategy resulted in the exchange of more messages than in any other strategy, followed by the NTP_s-strategy with a reduction to 59,62% of the communication generated by the *Base*-strategy. Noticeably less communication was produced by the NTP_d-strategy and the DL_d-strategy, with a reduction down to 15% and 17% respectively.

6.1.2. Number of offers created

The total number of offers created by the hosts for each run was recorded. In figure 6.1 the average number of offers for all runs is visualized as a boxplot.

Most offers were created using the *Base*-strategy (21), followed by the NTP_s-strategy (15) and the DL_d-strategy with 6 offers on average. Only 4 offers were created on average using the NTP_d-strategy. Compared to the *Base*-strategy this means on average 81% less offers were created.

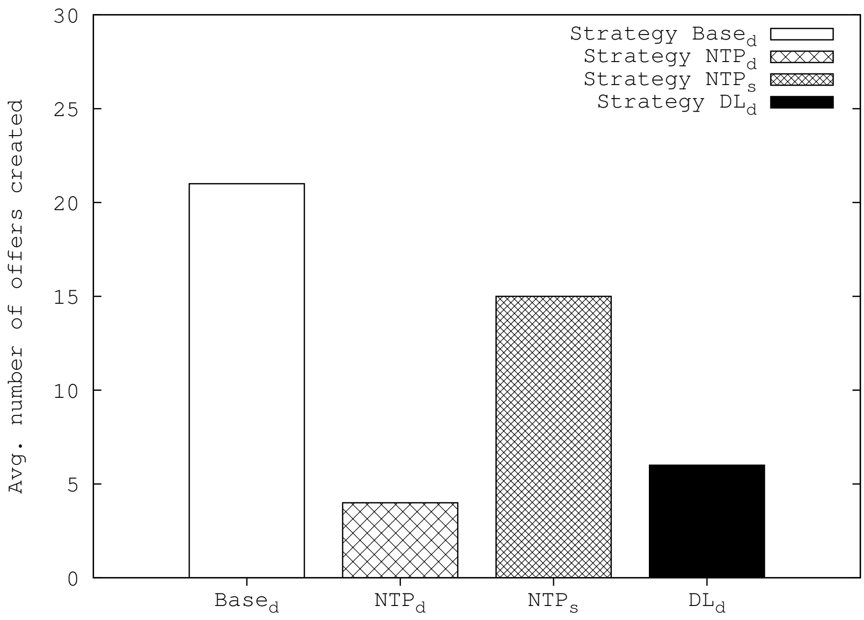


Figure 6.1.: Average number of offers created.

6.1.3. Effect of increasing network density

For the simulation runs different numbers of hosts were used to study the influence of increased network density on the communication and computation in all strategies. Figure 6.2 shows the average number of messages exchanged with an increasing number of hosts (10 to 50). For the message count the sum of host-to-host (H2H) and client-to-host (C2H) messages was used.

Using the NTP_d- and DL_d-strategy communication increases slightly with more than 30 hosts populate the network. A much higher increase in communication at this point is documented for the *Base*-strategy. The number of exchanged messages increases almost linear using the NTP_s-strategy.

6.1.4. Increase of communication over time

Using the second `DataRecorder` variables for each tick have been recorded. Figure 6.3 shows a timeline of the increase in average number of messages exchanged for all strategies.

The *Base*-strategy has the highest increase in communication, with quadratic growth between 0 and 45 ticks, settling after about 60 ticks to a logarithmic increase. With an about logarithmic increase the NTP_s-strategy has the second highest in-

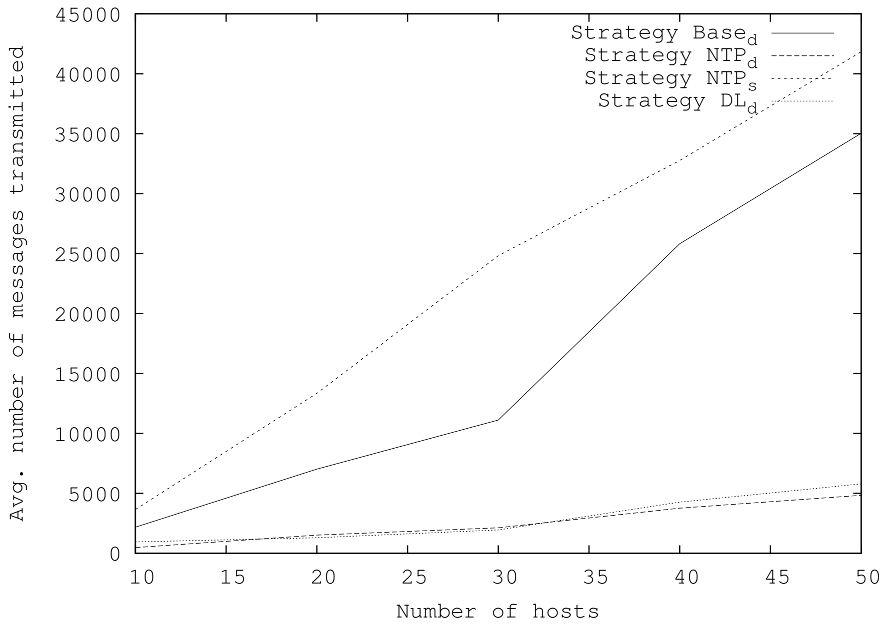


Figure 6.2.: Communication with increasing network density.

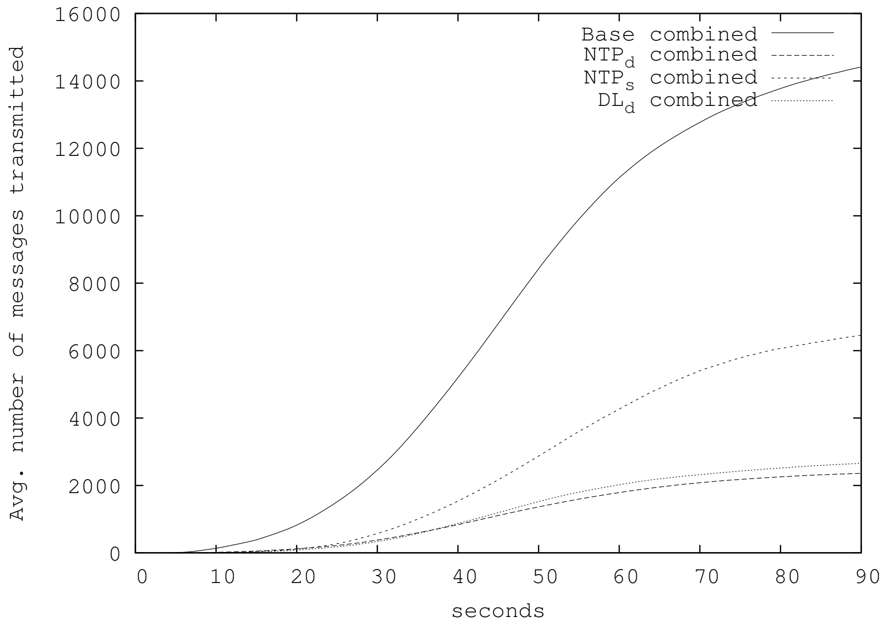


Figure 6.3.: Increase of communication over time.

Table 6.2.: Average computation times for host decision algorithms.

Strategy	Avg. Number of Hosts	Avg. Computation time (ms)
Base	30	64.84
NTP_d	30	1775.24
NTP_s	30	4832.69
DL_d	30	814.67

crease in the average number of messages exchanged. The DL_d– and NTP_d-strategy have a comparably low increase rate that is about constant after 60 ticks.

6.1.5. Comparison of computation time

Computation time was measured also for the host decision algorithms performed after receiving a request. Table 6.2 shows the average computation times[1] for a single run for all strategies. These values only give a rough estimate with regard to the different strategies. There can be a huge difference in these values on different systems, particularly on mobile devices. Most valuable for comparing the computational effort are the considerations on time complexity in section 4.5.3.

The NTP_s-strategy has the highest computation times, more than 70 times higher compared to the $Base$-strategy, which has the lowest computation time of all strategies.

6.1.6. Success rate

Whether a run was successful or not, in a sense that the client was able to find a shared-ride and reach the destination on time, was recorded. The euclidean distance between the client start position and the destination was 2067,2 meters, the distance along the shortest-path was 2389,8 meters. Assuming walking speed of about 1.5 m/s a walk takes roughly half an hour (28 minutes). For the $Base$-, NTP_d and DL_d-strategy all runs were successful. Out of all recorded for the NTP_s-strategy 81% were successful. All of these runs were finished on time, but took roughly twice as long compared to the $Base$-strategy.

[1] Times measured on system with the following specifications:
Pentium-M 725 1.8Ghz, 1GB RAM, Debian Sarge Linux (kernel 2.6.12-10-686),
JRE: build 1.5.0_06-b05, mixed mode, sharing.

6.2. Discussion

This section discusses the results with regard to the expectations from section 5.2.4. At first the results show a significant reduction in overall communication for the strategies using the network time prism or *dl*-subtree as a filter criterion. This is true for client-to-host and host-to-host communication, with reduction of communication down to 15%(NTP_d) compared to the unfiltered approach. The results reveal a large gap between the number of raw host-to-host messages and the unique messages received. This shows convincingly that the number of messages that is rebroadcasted has a great influence on the overall message exchange and that reducing the number of messages, which is produced in the first place represents efficient way to reduce overall communication.

For the number of offers created the results show a similar reduction for the strategies using filtering techniques based on time geography. The reduction of offers using the NTP_d-strategy lies within the expected amount: 80% of the generated hosts most likely did not have a route valuable for the client, yet using the $Base$-strategy they still made an offer. With an average host count of 30 the results show that using the $Base$-strategy 70% of the hosts made an offer, whereas this number is only 13% when applying the NTP_d-strategy. The slightly higher number of 20% for the DL_d-strategy fulfills the expectations from chapter 5. The number of offers created using the NTP_s-strategy is a little higher than expected with an average of 50% of all hosts creating an offer. In section 5.2.4 it was argued that the overall number of messages is to be higher, because the trip duration will be longer using this strategy. This is proven by the results (trip duration is about twice as long compared to the $Base$-strategy), but cannot solely explain the difference of 37% between the NTP_s- and NTP_d-strategy. The recorded data sets indicate that using the NTP_s-strategy more than half of the hosts generated in the "random start and destination location" category had a trip valuable for the client, because there was a node in their future travel route that brings the client closer to the final destination. Half of the hosts in this category means 30% – which is close to 37% and thus explains the difference.

Analyzing the communication with increasing network density and over time undermines previous considerations about message generation and rebroadcast. The $Base$-strategy shows a much steeper increase in overall communication compared to the other strategies as the number of hosts in network increases.

The measured computation times reflect the increase of calculating the network time prism and subtrees while using the NTP_d-, NTP_s- and DL_d-strategies.

Overall it can be stated that the results meet the expected outcome. They show that using the proposed strategies communication effort is reduced significantly, by reducing the number of offers created and messages transmitted. The results prove that the *Base*-strategy represents a very inefficient trip assignment strategy. The most efficient strategy is the NTP_d-strategy with the lowest level of overall communication and small increase of message exchange over time and with increasing network density. At the same time this strategy has the highest computation demand. The Dl_d-strategy represents a decent trade-off between the number of offers created and the relatively low computation times. The NTP_s-strategy was confirmed to be a risky strategy for the client: almost 20% of the shared-rides have been unsuccessful. Even with increased trip duration, however, this method is still superior to the *Base*-strategy in terms of communication load.

7. Conclusions and future work

The final chapter evaluates the simulation environment `P2PSim` and the simulation results with regard to the hypothesis and the requirements of the application from Chapter 5. Then conclusions are drawn and the chapter closes with an outlook and ideas for future work.

7.1. Evaluation of the simulation environment and results

7.1.1. Regarding the hypothesis

Time geography can be used to help solve the route planning task of a shared-ride trip planning system in transportation networks.

The first part of the hypothesis was verified completely. In chapter 2 the challenges of assigning travel supply to travel demand in ad-hoc shared-ride trip planning scenarios were described. In chapter 4 concepts from time geography have been applied to transportation network structures. Using these concepts, such as the network time prism or the potential path tree, in section 4.5.2 three trip assignment strategies were presented that help in assigning free host capacity to client demand in transportation networks, thus assisting in solving the route planning task. It has been shown by simulation that successful route planning is possible in a street network using implementations of the presented strategies.

The solution, consisting of travel assignment strategies, algorithms and data structures, finds optimal trips, in terms of minimizing a cost function, using local knowledge of the network at a given point in time.

In chapter 4 travel assignment strategies for ad-hoc shared-ride trip planning system have been defined, which considered the special properties of transportation networks. As stated in the previous chapters a shared-ride trip planning system has a non-deterministic, highly dynamic nature, therefore an optimal solution that minimizes the client's cost function is temporally limited. In section 4.5.3 algorithms and data structures were presented that enable hosts and clients in networks to decide, whether an request or offer is relevant to them. Using these methods only irrelevant offers were excluded, the remaining relevant offers included the optimal solution. This is underlined by the simulation results that document successful trips for almost all recorded runs. The small percentage of unsuccessful runs did not include any feasible host route, therefore no trip could be assigned (see section 6.2). Hence, this part of the hypothesis was verified.

The solution is efficient, meaning it significantly reduces communication cost, while limiting computational cost to less than quadratic time complexity.

This part of the hypothesis was verified. As stated in chapter 2 efficient communication and computation in shared-ride trip planning scenarios is very important, because the resources of mobile geosensor devices are limited. The network-based algorithms and data structures presented in section 4.5.3 have logarithmic average time complexity (for the hosts) and linear average time complexity (for the client). Since the input size n of the client algorithm is much smaller than the input size of the host algorithm (see section 4.5.3.4) the presented solution qualifies as efficient in terms of the hypothesis' requirement. The results presented in chapter 6 document significant decrease in communication cost for all three proposed strategies. This includes client-to-host and host-to-host communication.

A simulation environment can be designed and implemented for testing the solution in a real street network.

This thesis documents the design and implementation of the `P2PSim` simulation environment, which offers methods for running simulated shared-ride trip planning scenarios using one of the defined trip assignment strategies. Networks can be generated from input files and used for recording simulation runs including measurement data on communication and computation. More than 400 simulation runs with different parameters have been successfully conducted. Thus, the last part of the hypothesis was also verified.

7.1.2. Regarding the requirements of the simulation environment

In section 5.2 several functional requirements are defined that were considered in the design phase of the `P2PSim` software. This section evaluates the implementation with regard to these requirements.

The first requirement was to model clients and hosts as moving nodes in a shared-ride trip planning scenario. This requirement is met by `P2PSim` through the object-oriented modeling of these entities and their behavior in the `model` package. Communication is simulated by offering methods for receiving and sending offer and request messages by clients and hosts. Thus, this requirement is also met. Another important requirement was the import of GIS data and graph representation of the network. `P2PSim` supports the import of a shapefile and converts it to a network representation. However, this functionality was only tested with one specific dataset and must be tested further to guarantee compatibility with other datasets. This was sufficient for the purpose of verifying the hypothesis, but this functional requirement was only partially met. Realtime visualization of the network and the moving peers, another initial requirement, is possible using `P2PSim`. The decision to use a georeferenced polyline-based shapefile was not optimal: the exact coordinates of the network nodes make the simulation a realistic one, but at the price of lower display performance. Many float value calculations have to be executed by the `geotools` framework to update the display, which is counterproductive to smooth animation. This is part of the non-functional requirements, which had a low priority, because a prototype process model was used. The last two requirements, adjustable parameters, and recording and analysis functions were fully met by the implementation.

7.2. Conclusions

This thesis applied concepts from time geography and shared-ride trip planning to network structures. It was demonstrated that the solutions presented in [WR06] can be applied to transportation networks. This could not be intuitively deducted, because travel in transportation networks is spatially and temporally constrained. The shortest route is not necessarily the quickest, thus the route planning task becomes more difficult. The algorithms and data structures defined in this thesis use graph traversal techniques, together with concepts from time geography, such as the network time prism. Considering the simulation results the increase in computational cost is low compared to the gain in efficient communication. The three proposed trip assignment strategies and their implementations solved the trip planning for the client in a street network scenario and reduced communication between hosts and clients. This holds true for increasing host density and increasing trip length.

However, this thesis mainly explored filtering techniques that excluded irrelevant and redundant messages: the unpredictable nature of a shared-ride trip planning system remains of course. In a street network scenario the client will still be faced with decisions on partial routes, waiting times and route directions. As the simulation results document, using the NTP_s-strategy leads to longer trip durations and increased overall communication in the network. Since routes leading from the client to the destination will be uncommon in an urban street environment the heuristics for selecting an offer become very important. Here more than using the euclidean distance to the destination is necessary. For instance the client could decide to reach the destination later or accept trip destinations in a proximity to the original location.

The successful implementation of the simulation environment proved that multi-agent systems are a good choice for simulating and analyzing shared-ride trip planning scenarios. Modeling entities in an object-oriented manner enables the study of communication and movement of clients and hosts in these systems. `P2PSim` allows for further exploration of these scenarios. For instance different behavior or trip planning strategies can be added as well as different network structures.

7.3. Future work

This section gives an outlook on future work regarding shared-ride trip planning in transportation networks and further development of `P2PSim`.

7.3.1. User preferences

Considering user preferences [RMB04] would improve the strategies for ad-hoc shared-ride trip planning. For instance a host could prefer to provide rides only to non-smoking passengers or limit the baggage size to fit the car's capacity. This way, also clients could define route planning preferences. If methods are found to formalize and communicate such preferences, personalized decision strategies [RR04] could be used to improve the algorithms defined in this thesis.

7.3.2. Seat competition

In this thesis it was assumed that all hosts have sufficient passenger capacity. With only one client requesting a ride, competition for seats is not problematic. A more realistic approach is to track the number of available seats for each host and to add more than one client to the scenario. Then questions arise, such as: Which out of a set of feasible hosts is chosen in order to use the seat capacity most efficiently? If a client accepts partial routes to its destination, which order of hosts services the request best? If a host's route is feasible for more than one client, which client shall be preferred? Answers to these questions become more important as the number of clients and hosts in the network increases and more travel assignments become available.

7.3.3. Multimodal networks

Modern urban transportation networks are multimodal, consisting of multiple modes or types of transportation. Examples include subway, car, tram and train networks. By modeling these different modes, the network becomes more complex and trip planning more difficult. In addition to considering node stops, switching between different modes of transport must be accounted for. It might be interesting to explore the influence of multimodal transportation to strategies in shared-ride trip planning scenarios.

7.3.4. Flexible clients and hosts

In the conducted simulation runs the behavior of hosts and the client were not flexible: the client was looking only for host routes that contained its current position and hosts were not willing to make detours to pick up the client. More travel opportunities arise if these restrictions are relaxed. The client, for instance, could be willing to walk a short distance within a predefined radius in order to get a ride from a host. This way the chances of finding an optimal trip can be increased. This would require a recalculation of the network time prism, because the client position is changing. The challenge is to find an efficient algorithm that supports these flexible strategies.

7.3.5. Optimizations

Calculation of the network time prism and the potential path trees is expensive and becomes the dominating factor in computational complexity as the number of nodes and edges in the network increases. Recalculation is necessary when the client's current or destination location changes. It might be worthwhile to reduce the recalculation to a minimum by using precalculated potential path trees and lookup tables. Another idea would be for hosts to calculate only local parts of the client's potential path tree and to use peer-to-peer communication to acquire other parts when necessary.

References

[Atk99] Peter M. Atkinson. *Advances in Remote Sensing and GIS Analysis*. John Wiley & Sons, 1999.

[Bal96] Helmut Balzert. *Lehrbuch der Software-Technik*, volume 2. Spektrum, Heidelberg, 1996.

[BCI+01] Chatschik Bisdikian, Jim Christensen, John Davis II., Maria R. Ebling, Guerney Hunt, William Jerome, Hui Lei, Stephane Maes, and Daby Sow. Enabling location-based applications. In *WMC '01: Proceedings of the 1st international workshop on Mobile commerce*, pages 38–42, New York, NY, USA, 2001. ACM Press.

[BD03] Louise Barkuus and Anind Dey. Location-based services for mobile telephony: a study of users privacy concerns. In *Proceedings of the INTERACT 2003, 9th International Conference on Human-Computer Interaction*, 2003.

[Bel58] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[BHE00] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[BT05] Itzhak Benenson and Paul M. Torrens. *Geosimulation*. John Wiley and Sons, Ltd., 2005.

[CDMF00] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: The guide project. In *Proceedings of CHI 2000, Netherlands*, pages 17–24, 2000.

[CEE+00] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: Application driver for wireless communications technology. Technical report, UCLA Computer Science Department, Los Angeles, California, 2000.

[CHH01] Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, volume 9, page 9008. IEEE Computer Society, 2001.

[CHN03] N. Collier, T. Howe, and M. North. Onward and upward: The transition to repast 2.0. In *Proceedings of the First Annual North American Association for Computational Social and Organizational Science Conference, Electronic Proceedings*, 2003.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[Die05] Reinhard Diestel. *Graph Theory*. Springer, Heidelberg, 2005.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graph. *Numerische Mathematik*, 1:269–271, 1959.

[DWN02] J. F. Dillenburg, O. Wolfson, and P. C. Nelson. The intelligent travel assistant. In *5th International Conference on Intelligent Transportation Systems*, pages 691–696. IEEE, Singapore, 2002.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1: Equations und Initial Semantics*, volume 6 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1985.

[Epp94] D. Eppstein. Finding the k shortest paths. In *Proc. 35 th Symp. Foundations of Computer Science*, pages 154–165. IEEE, 1994.

[Gar71] M. Gardner. Mathematical games: On cellular automata, self-reproduction, the garden of eden, and the game 'life'. *Scientific American*, 224:112–117, 1971.

[GFL04] N. Gueguen and J. Fischer-Lokou. Hitchhikers smile and receipt of help. *Psychological Report*, XCIV(3):756–760, 2004.

[HNR72] P. E. Hart, N. J. Nilsson, and B. Raphael. Correction to "
a formal basis for the heuristic determination of minimum cost paths"
. *SIGART Newsletter*, 37:28–29, 1972.

[Hä70] Torsten Hägerstrand. What about people in regional science ? *Papers of the Regional Science Association*, 24:7–21, 1970.

[KH98] M.-P. Kwan and X.-D. Hong. Network-based constraint-oriented choice set formation using gis. *Geographical Information*, 5:139–62, 1998.

[KKP99] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *International Conference on Mobile Computing and Networking (MOBICOM)*, pages 271–278, 1999.

[Knu97] Donald Knuth. *The Art of Computer Programming*, volume 3 : Sorting and Searching, chapter 6.2.3, pages 458–475. Addison-Wesley, 1997.

[Lju01] Peter Ljungstrand. Context awareness and mobile phones. *Personal and Ubiquitous Computing*, 5(1):58–61, 2001.

[Mae95] P. Maes. *Artificial Life, An Overview*, chapter Modeling adaptive autonomous agents., pages 135–162. The MIT Press, 1995.

[MBLA96] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system: a toolkit for building multi-agent simulations. Technical report, Santa Fe Institute, 1996.

[Mil91] Harvey J. Miller. Modelling accessibility using space-time prism concepts within geographical information systems. *International Geographical Information Systems*, 5:287–301, 1991.

[Mil05a] Harvey J. Miller. A measurement theory for time geography. *Geographical Analysis*, 37:17–45, 2005.

[Mil05b] Harvey J. Miller. Necessary space-time conditions for human interaction. *Environment and Planning B: Planning and Design*, 32:381–401, 2005.

[MLG03] P. Mohapatra, J. Li, and C. Gui. Qos in mobile ad hoc networks. *IEEE Wireless Communications*, pages 45–52, 2003.

[NCV06] M.J. North, N.T. Collier, and J.R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1–25, 2006.

[NDK04] Silvia Nittel, Matt Duckham, and Lars Kulik. Information dissemination in mobile ad-hoc geosensor networks. In Max J. Egenhofer, Christian Freksa, and Harvey J. Miller, editors, *GIScience*, volume 3234 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2004.

[Nor00] Val Noronha. Transportation networks. Technical report, NCGIA Core Curriculum in GIScience, 2000.

[NOS+03] Itsuki Noda, Masayuki Ohta, Kousuke Shinoda, Yoichiro Kumada, and Hideyuki Nakashima. Evaluation of usability of dial-a-ride systems by social simulation. In David Hales, Bruce Edmonds, Emma Norling, and Juliette Rouchier, editors, *MABS*, volume 2927 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2003.

[NSC+03] S. Nittel, A. Stefanidis, I. Cruz, M. Egenhofer, D. Goldin, A. Howard, A. Labrinidis, S. Madden, A. Voisard, and M. Worboys. Report from the first workshop on geosensor networks. Technical report, University of Maine, USA, 2003.

[Oes01] B. Oestereich. *Objektorientierte Softwareentwicklung*. Oldenbourg, 2001.

[Pre01] Bruno R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*. John Wiley & Sons, 2001.

[QS03] D. Qi and R. Sun. A multi-agent system integrating reinforcement learning, bidding and genetic algorithms. *Web Intelligence and Agent Systems*, 1:187–202, 2003.

[RCS06] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. *The Geography of Transport Systems*. Routledge, 1 edition, January 2006.

[RMB04] Martin Raubal, Harvey J. Miller, and Scott Bridwell. User centered time geography for location-based services. *Geografiska Annaler B*, 86(4):245–265, 2004.

[RR04] Claus Rinner and Martin Raubal. Personalized multi-criteria decision strategies in location-based decision support. *Journal of Geographic Information Sciences*, 10(2):149–156, 2004.

[RT99] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks, 1999.

[SBGP04] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Bodhi Priyantha. Tracking Moving Devices with the Cricket Location System. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.

[Spi04] S. Spiekermann. *Location Based Services*, chapter General Aspects of Location Based Services, pages 10–24. Morgan Kaufmann, 2004.

[Tan03] Andrew S. Tanenbaum. *Computernetzwerke*. Pearson, 4., überarb. aufl. edition, 2003.

[Tes06] Leigh S. Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. Staff General Research Papers 12514, Iowa State University, Department of Economics, March 2006. available at http://ideas.repec.org/p/isu/genres/12514.html.

[Wei97] Mark Ellen Weiss. *Data Structures and Algorithm Analysis in C*. Addison-Wesley, 1997.

[WN05] Stephan Winter and Silvia Nittel. Shared ride trip planning with geosensor networks. 2005.

[WNNC05] Stephan Winter, Silvia Nittel, Arda Nural, and Trang Cao. Shared ride trip planning in large transportation networks. In *Proceedings Symposium on Societies and Cities in the Age of Instant Access*, 2005.

[WNed] Stephan Winter and Silvia Nittel. Ad-hoc shared-ride trip planning by mobile geosensor networks. *International Journal of Geographical Information Science*, accepted.

[Wol84] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311:419–424, 1984.

[Woo02] Michael Wooldrige. *Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[WR06] Stephan Winter and Martin Raubal. Time geography for ad-hoc shared-ride trip planning. 7th International Conference on Mobile Data Management 2006, 05 2006.

[YHW01] Harvey J. Miller Yi-Hwa Wu. Computational tools for measuring space-time accessibility within dynamic flow transportation networks. *Jorunal of Transportation and Statistics*, 4, 2001.

[ZG04] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.

A. P2PSim User's Manual

This section contains the `P2PSim` User's Manual, which provides information on installing the application, as well as running and recording simulation runs.

B. CD-ROM

The enclosed CD-ROM has the following directory structure:

- `/docs`: `P2PSim` manual and `Javadoc` source code documentation.
- `/thesis`: Copy of this thesis in `PDF` format.
- `/src`: `P2PSim` source files.
- `/dist`: Binary distribution of `P2PSim`.

Independent Study Declaration

The thesis entitled "Time Geography for Shared-ride Trip Planning in Transportation Networks" is based on my own work and was entirely written by myself. Where work of others has been used it is fully acknowledged in the text. The thesis has not been submitted to obtain an other educational qualifications.

Die Diplomarbeit "Time Geography for Shared-ride Trip Planning in Transportation Networks" basiert auf meiner eigenen Arbeit und wurde von mir selbständig angefertigt. Arbeiten Dritter sind im Text als solche kenntlich gemacht. Die Diplomarbeit wurde nirgends eingereicht um andere wissenschaftliche Qualifikationen zu erlangen.

Münster, April 30th, 2006

Sven Teßmann