Agent Behavior in Peer-to-Peer
Shared Ride Systems

Yunhui WU

SUBMITTED IN TOTAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF MASTER
OF GEOMATIC ENGINEERING (BY RESEARCH).

Department of Geomatics
The University of Melbourne, Australia

January 2007

Supervisor:
Dr. Stephan Winter
Abstract

Shared ride systems match the travel demand of transport a client with the supply of vehicles, or hosts, so that the client find rides to their destinations. A peer-to-peer shared ride system allows a client to find rides in an ad-hoc manner, by negotiating directly with nearby hosts via radio-based communication. Such a peer-to-peer shared ride system has to deal with various types of hosts, such as private cars, taxicabs and mass transit vehicles. Agents, i.e. a client and hosts, have diverse behaviors in such systems. Their different behaviors affect the negotiation process, and consequently the travel choices. Preliminary research (Winter et al. 2005) has investigated peer-to-peer shared ride systems with homogeneous hosts and immobile client. This thesis extends their work to multiple types of agents. It focuses on what are typical agent behaviors in peer-to-peer shared ride systems, and how these behaviors affect negotiation processes in a dynamic transport environment.

This thesis presents and discusses a model of a peer-to-peer shared ride system with diverse types of hosts and one client with different levels of knowledge. Agents’ typical natures and behaviors contributing to peer-to-peer shared ride systems are identified and formalized in the model. This multi-agent model allows testing different communication and way-finding strategies in peer-to-peer shared ride systems. The inheritance design of the model makes it easy to extend and suitable for other related investigations and further research. A computer simulation is employed to implement this model.

Experiments are designed to test agent behaviors under different communication and way-finding strategies. The results of experiments demonstrate that different types of agents enrich the choices of the client, and lead to local solutions that are nearly optimal.
Declaration

This is to certify that

(i) the thesis comprises only my original work towards the Masters except where indicated in the Preface,

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

(iii) the thesis is 23,000 words in length, inclusive of footnotes, but exclusive of tables, maps, appendices and bibliography.

December 31st, 2006

Yunhui WU
Acknowledgments

I would like to express my gratitude to my supervisor, Dr. Stephan Winter, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and guidance to research skills (e.g., reading, writing, analysis and presentation), and his advices in writing the thesis. I am grateful for his efforts to make my time in the University of Melbourne rewarding and fruitful. I am also grateful to other (ever) members of the shared ride research group, their invaluable contribution and support to me: Linjie Guan working on trip quality, a good friend and research partner from the beginning, helping me a lot in programming and spending much time on discussing the research work with me; Christian Gaisbauer doing a wonderful work on mobility models, introducing me to the culture of German and sharing his research views; Dr. Caulos Vieira working on security issue and also a patient uncle talking to me about all aspects of life; Rupert Leigh providing a excellent work on mobility models as well, and making the simulation model in this thesis much attractive.

Special thanks to Mingzheng Shi, who shared his successful research and thesis writing experiences with me. Unlimited thanks go to Mark Burkitt and Serene Yeo for proof reading the thesis and spending their time on providing numerous invaluable comments especially the last days of thesis writing during Christmas and New Year.

Many thanks to Supawan Wongprayoon for being a patient, fun and supportive friend, a good cooking teacher and a great source of life advice. Numerous thanks also to Vimvipa Stitstian, Salaheddin Samir Mousa Al-Ajlouni, Cristian Gordinoto, Tao Chen and Erjiang Fu for being nice friends, who I met in Department of Geomatics. I would like also thank Prof. Ian Bishop for his suggestions on degree-related issues; admin officers Nghiem Tran, Sarah Stewart and Miharla Cpalmean for assistance of all aspects in office.
Deepest thanks to my families for the support they provided me through my entire life, and my housemates in Melbourne, without whose love, encouragement, company and assistance, I would not have finished this thesis.
Contents

Chapter 1 Introduction .................................................................................. 1

1.1 Motivation and Research Questions......................................................... 2
1.2 Target and Hypothesis ............................................................................. 3
1.3 Methodology ............................................................................................ 4
1.4 Organization of the Thesis ...................................................................... 4

Chapter 2 Literature Review ....................................................................... 5

2.1 Agents and Shared Ride Systems ............................................................... 5

2.1.1 Carpooling/Vanpooling ................................................................. 6
2.1.2 Hitchhiking ....................................................................................... 8
2.1.3 Mass Transit Systems ................................................................. 9
2.1.4 Dial-A-Ride .................................................................................... 10
2.1.4 Web-based Shared Ride Applications .................................... 10

2.2 Agent-based Transportation Simulation ................................................. 11

2.2.1 Multi-agent Systems for Transportation Simulation ....................... 12
2.2.2 Cellular Automata and Geographic Automata Systems ................. 13
2.2.3 Agents in Transportation Systems ............................................. 14
2.2.4 Simulation Toolkits for Transportation Systems ....................... 15
2.2.5 Dynamic Transportation Applications .................................. 16
2.3 Shared Ride Trip Planning ................................................................ 18
4.5 Trip Planning Algorithms ................................................................. 63
  4.5.1 LPA* Algorithm ................................................................. 63
  4.5.2 $K$ Shortest Paths Algorithm ........................................... 69
  4.5.3 Multi-criteria Optimization Algorithm ............................. 71
4.6 Simulation Assessment ................................................................. 72

Chapter 5 Experiments ................................................................. 74
  5.1 Simulation Results ................................................................. 75
    5.1.1 Boundary Effect with Homogeneous Agents .................... 75
    5.1.2 A Client with Various Motilities and Level of Knowledge .... 78
    5.1.3 Types of Hosts ................................................................. 81
    5.1.4 A Mixed Case with All Types of Agents ......................... 84
    5.1.5 Agent Mobility Model with Street Centrality ................. 86
    5.1.6 Multi-Criteria Trip Planning ........................................... 88
  5.2 Discussion ................................................................................. 90

Chapter 6 Conclusions and Future Work ....................... 94
  6.1 Conclusions ................................................................. 94
  6.2 Evaluation of the Simulation Model ..................................... 96
  6.3 Limitations and Future Work ............................................. 98

Bibliography ............................................................................. 100
List of Tables

Table 4-1: Node features........................................................................................................... 35
Table 4-2: Simulation environmental features ........................................................................... 36
Table 4-3: Message features ..................................................................................................... 38
Table 4-4: Agent properties ..................................................................................................... 43
Table 4-5: Agent methods ....................................................................................................... 44
Table 4-6: A client properties................................................................................................... 48
Table 4-7: A client methods ..................................................................................................... 49
Table 4-8: Host properties ....................................................................................................... 54
Table 4-9: Host methods ......................................................................................................... 54
Table 4-10: Car properties ....................................................................................................... 56
Table 4-11: Car methods ......................................................................................................... 56
Table 4-12: Taxi properties ..................................................................................................... 57
Table 4-13: Taxi methods ....................................................................................................... 58
Table 4-14: Structure of BusLine ............................................................................................ 59
Table 4-15: Bus properties ....................................................................................................... 60
Table 4-16: Bus methods ......................................................................................................... 61
Table 4-17: BusStop properties .............................................................................................. 63
Table 4-18: BusStop methods ................................................................................................. 63
Table 4-19: LPASearch properties .......................................................................................... 67
Table 4-20: LPASearch methods ............................................................................................ 68
Table 4-21: LPANode (inherited from Node class) properties ........................................ 68
Table 4-22: LPAEdge properties .................................................................................. 69
Table 5-1 Parameter settings of transportation hosts .................................................. 74
List of Figures

Figure 4-1: The structure of simulation components...................................................... 34
Figure 4-2: Structure of shared ride negotiation process............................................... 40
Figure 4-3: Hierarchy of agent classes ........................................................................... 41
Figure 4-4: Two mass transit lines in a grid network ..................................................... 59
Figure 4-5: Using an offer to update weights in LPA* algorithm .................................. 65
Figure 5-1: Positioning of a client’ route........................................................................ 76
Figure 5-2 (a): Travel time along 3 routes................................................................. 77
Figure 5-2 (b): The number of messages along 3 routes .............................................. 77
Figure 5-3(a): Travel time of a client with different mobility and knowledge............... 80
Figure 5-3(b): The message numbers of a client with different mobility and knowledge .......................................................... 80
Figure 5-4: Comparison of hosts as mass transit running on two lines......................... 84
Figure 5-5(a): Travel time of different host composition and a mobile client ............... 85
Figure 5-5(b): Number of messages of different host composition and a mobile client 85
Figure 5-6: Test network containing named streets (Leigh 2006)............................... 87
Figure 5-7: Comparison of agents having different level of knowledge..................... 88
Figure 5-8: Comparison of multi-criteria trip planning with single criterion................. 90
Chapter 1
Introduction

Movement of people in a city forms a complex system. It includes the street network and other ways of traveling, traffic rules, traffic infrastructure (e.g., traffic lights, signs) as well as the cognition, decisions and actions of intelligent, autonomous agents such as pedestrians and vehicle drivers. This complex system is burdened by more and more traffic and expanding cities. It is estimated that travel in privately owned vehicles accounts for 81.3% of all local trips in the United States. High density of vehicles on roads causes traffic congestion and air pollution. According to the research of (Harford 2006), by reducing the use of private vehicles, it is expected to save $20 billion for congestion reduction, and decrease 25% greenhouse gases and pollution per year. In this situation a peer-to-peer shared ride system can contribute to a relief of the critical situation: it enables people to negotiate in an ad-hoc manner for ride sharing, and thus, helps reducing the traffic, increases urban access, and improves the integration of different modes of transport. In such a system, pedestrian are the agents with transport demand, called clients; drivers in vehicles are the agents providing the transport supply, called hosts.

Peer-to-peer shared ride systems depend on the negotiation between peer users to arrange shared rides. Compared to traditional shared ride systems, the proposed peer-to-peer systems are more economical, flexible and scalable for the following reasons. First of all, there is no central management server needed to implement such systems, and no additional devices needed for peer users, thereby lowering installation cost. Secondly, peer users can ad hoc to existing systems and are able to look for shared ride in a real-time transport environment. Sequentially, peer-to-peer shared ride systems are not limited in the capacity of central management server and complexity of ride arrangement to suit every user’s requirements, therefore such systems are able to
implement in a large area, e.g. country wide. Extended from the previous research on peer-to-peer shared ride systems, this thesis focuses on agent behaviors in the circumstance. By modeling in a simulation, this thesis investigates the various agent behaviors under different communication and trip planning strategies, and demonstrates the importance of involving types of agents in analyzing peer-to-peer shared ride systems.

1.1 Motivation and Research Questions

The motivation for this work comes from two directions: the effect of agent behaviors to shared rides trip planning, and the requirement from peer-to-peer design.

Although peer-to-peer shared ride systems have many advantages as addressed above, there is no operating system realized. In this situation to understand such systems, a method is needed to build a vital traffic environment, where agents can move and negotiate with each other. A peer-to-peer shared ride system has to deal with various types of agents, such as private cars and mass transport vehicles, or mobile and immobile client, to cope adequately with the complexity of urban movements. The agents’ different interests, capacities and behaviors affect the negotiation process, and consequently, the trips made. For example, hosts can be distinguished by their travel speed, their passenger capacity and their fare structure, and clients can be distinguished by mobility. Additionally, transportation information keeps changing in a dynamic traffic environment. Without central management service, agents make decision totally depends on local knowledge. For this reason, how agents communicate efficiently and effectively, and how agents perceive environment, collect and deal with real-time transport information to plan their routes are key points to secure shared rides. Those aspects make agents behavior study necessary.

This thesis concerns on what are typical agents and their behaviors in peer-to-peer
shared ride systems, how different communication and way-finding strategies affect negotiation processes, and how agents’ various behaviors affect the result of shared rides.

1.2 Target and Hypothesis

In peer-to-peer shared-ride systems, agents, i.e. a client and hosts, have knowledge of their environment. They can collect and transmit information from/to their neighbors. Frequently a client has choices among multiple travel opportunities in the neighborhood. A client has preferences and various optimization criteria, such as travel time, fares or both, and is able to make current optimal decisions based on their knowledge. However, for practical reasons agents have only local and current knowledge of their environment. Previous research (Winter and Nittel 2006) investigates the ability to make trip plans from different levels of local knowledge. They define two criteria of mobile geosensor networks: *effective* resulting in trips close to the optimal trip according to a cost function; and *efficient* significantly less communication effort for an effective trip in terms of the number of broadcasted messages in negotiations than that for collecting complete transportation knowledge. This previous investigation is based on a simulation with homogeneous hosts and an immobile client, and proves that a mid-range communication schema is both efficient and effective. This thesis extends the preliminary work to multiple types of hosts, and one more realistic client with different mobility and level of knowledge. By using a redesigned simulation model, communication and way-finding strategies are tested and compared in the multi-agent circumstance. Considering the complicate issues raised by client competition, multi-client will not be involved in this thesis.

The hypothesis of this thesis is that upon involving other types of agents, the trips will change significantly, but mid-range communication is still both efficient and effective compared to other communication strategies. With local knowledge limited to
mid-range neighborhoods, agents are able to make decisions close to optimal ones.

1.3 Methodology

This thesis employs a computer simulation to investigate agent behaviors in peer-to-peer shared ride systems. The natures and behaviors of agents, which affect negotiation processes are identified and formalized in this simulation. The simulation is realized as a multi-agent system, which allows us to model and understand individual behavior of different agents. A negotiation protocol and mechanism is designed for communicating between agents. The approach requires identifying and specifying the essential aspects of an urban shared ride system, implementing them in a multi-agent system, and then running large numbers of random experiments to collect required data for analysis. Results of experiments are typical according to the setting of parameters and simulation environment. All agents are designed in an inherited architecture in this model, which are also suitable for other experiments (e.g. trip quality analysis) in peer-to-peer shared ride system, and able to extend for further investigation.

1.4 Organization of the Thesis

Chapter 2 reviews related researches on shared ride systems, agent-based transportation simulation and trip planning algorithms. Chapter 3 discusses the types of agents, their knowledge and economic modes in shared ride systems. Chapter 4 introduces the design of a multi-agent simulation to model the proposed peer-to-peer shared ride system in chapter 3. Chapter 5 designs experiments on agent behaviors under different communication and way-finding strategies. Results of experiments are also examined in this chapter. Chapter 6 comes with the conclusions, limitation and future works.
Chapter 2

Literature Review

At the time of writing the thesis, there is no peer-to-peer shared ride system in practice. Current applied shared ride systems often experience problems in efficiency, convenience, organization and size of service area/number of users. To solve these problems, a peer-to-peer shared ride mode is proposed.

In lieu of actual practice in transportation systems, some researchers use computer simulation methods to study issues in this field. However understanding complex negotiation processes and realistic agent modeling still needs further investigation. As motivated in Section 1.1, this thesis focus on the agent behaviors in peer-to-peer shared ride systems and how negotiation processes vary under different communication and way-finding scenarios. For this purpose, this chapter reviews a number of related works on shared ride systems, transportation simulation, and trip planning strategies.

This chapter starts from an overview of shared ride systems, from user self-organized to centrally managed, from dial-in to web-based, followed by an introduction of agent-based transportation simulation, the approach used to investigate agent behaviors in this thesis. This review also addresses the algorithms and techniques of trip planning in a dynamic environment. Previous researches on peer-to-peer shared ride systems are summarized in the last section. It is expected that reviewed works contribute to extending knowledge of developing an efficient simulation for peer-to-peer shared ride systems to analyze agent behaviors.

2.1 Agents and Shared Ride Systems

Consider the following case:
Out of his office in north-west corner of the city, John is going to meet his customer at a south-east point of the city. He has no car and needs to arrive at the appointment in half hour. What he can do in this case? Walk to a station one block away to catch the city-circle tram, or flag a taxi passing by? What if he misses the tram or is confined by budget to take a taxi? How can John utilize surrounding transportation to get to the appointment on time?

Shared ride systems provide an alternate model of traditional transportation, in which two or more people with common destination or destinations, share the use of a motor vehicle for trip to their destinations (Winter and Nittel 2005). The one sharing a vehicle with others can save the cost of fuel, tolls, parking, vehicle maintenance and insurance. Fewer vehicles are needed in shared ride systems compared to traditional transportation, therefore fewer cars are in traffic flow, and consequently fewer traffic congestions and accidents happen. Besides saving travel cost and time, high occupancy rates also reduce the consumption of fuel and the corresponding pollution of air. Share ride systems are not popular for several reasons, such as security and privacy issues, but there are potential economic and environmental reasons to implement such systems.

To introduce peer-to-peer mode, this section reviews typical kinds of shared ride systems in the real world. These systems are diverse in shared ride service providers, e.g. non-profit organizations or government agencies, and operation models, such as manual coordinated, dial-in and web-based.

2.1.1 Carpooling/Vanpooling

Carpooling, also known as ride-sharing, lift-sharing and car sharing, is defined as a prearranged shared ride system, in which a group of people each have a car but travel together regularly, particularly between home and work place, to save costs (Miller and Green 1977). An additional advantage is that carpooling provides a social connection
among commuters. Vanpooling is similar to carpooling but on a larger scale of travel group and concurrent savings in fuel and vehicle operating costs. Van/car vehicles can be provided by individuals, or rented from government or private organizations.

There are a number of carpooling/vanpooling programs over the world, such as *Courtesy Ride* program (Blumstein and Miller 1983) comprised of 15 major arteries and 17 pick-up stations to reduce the number of highway vehicles; *Upass* transportation program (Willams and Petratt 1993) offering commuting options to University of Washington students, faculty and staff at a reduced price; *StattAuto* in Berlin, and Mobility CarSharing in Switzerland (Shaheen et al. 1998) are among the largest carpooling organizations in Europe; *Carlink II* Program (Shaheen et al. 2004) cooperating by the government and local business partners in California area; and Commute Trip Reduction Rideshare Programs (Victoria Transport Policy 2005) provided to residents of a suburban community in Southern California.

Ferguson (1997) reviews the history of carpooling in America, analyzes census data and concludes that the lack of necessary policy development and theory research on carpooling formation have become more important impact factors on recent declines in carpooling than others, such as age, gender, family income, and urban form. In carpooling/vanpooling programs, people often need to meet at a common pick-up point at a specific time. Participants need to arrange the use of cars and commuting routes for single/multiple rides. New users can participate in existing pooling routes; alternatively they can create a new group to travel together. The main disadvantages of carpooling are lack of privacy and convenience issues (e.g. not a door-to-door service, limited service covering area). The driver also carries the additional burden of the safety of passengers. Finally, people may have difficulties with organizing and keeping together with other commuters. Carpooling/vanpooling requires users to plan trip in advance, thereby this kind of shared ride system is not suitable for frequent change of route demands.
2.1.2 Hitchhiking

Hitchhiking is a form of transport, where travelers try to get a ride from a car or trunk by placing themselves close to the road or at parking sites. Hitchhiking is also called lifting, thumbing, or thumb up a ride, because an up thumb are used as a sign of looking for a ride by hitchhikers. There are several reasons for hitchhiking: no funds or no transportation, broken vehicles, physical inability for driving and accident. Hitchhiking is also a method to meet people. From the research of Rinvolucri (1974), 60% people give lifts because they want company, and 30% people do so because they feel lonely and want to talk. The advantage of hitchhiking is obvious: it is the cheap way to getting around. The disadvantage of hitchhiking is also obvious: it is a dangerous way to travel with strangers.

Hitchhiking is a popular shared ride model. Due to its potential risk, different countries treat it in distinct ways. Hitchhiking was much safer in Poland during communist regime period, where hitchhikers were required to use a formal document for recording travel and drivers were encouraged by completing ride confirmation, which could be used as a shopping discount; in Russia, hitchhiking becomes an adventure sport; in Australia, hitchhiking is not recommended but still popular among backpackers; in Canada, hitchhiking is conditionally banned on highways; in the United States, hitchhiking happens in areas where there is no public transportation and hikers are welcomed by local communities as they often contribute to local economies, but hitchhiking is forbidden in some areas for security reason.

There are several factors affecting hitchhiking, such as traffic density, traffic speed, the number, gender and presentation of hikers, weather, time of the day and hiking location. To save the time of drivers to stop and enquire, hikers can hold a sign with their destination. The communication range of hikers is another factor (Tessmann 2006), as hikers’ view is limited by one specific road and direction. Because drivers normally will
not change their route to meet the requirement of hikers, a long distance travel can involve many rides and walks. Generally, low success rate and high potential risk make hitchhiking a sub-option for shared ride seekers.

### 2.1.2 Mass Transit Systems

Mass transit, also called public transport, is a well-known transportation form to share vehicles. Compared to other shared ride systems, mass transit usually runs on a fixed route and often under a timetable. Generally, mass transit includes rail and bus service, and wider definitions would include scheduled airline, ferries, and any system that provide transport service to the public. Social, environmental and economic benefit from US mass transit systems have been demonstrated by Harford (2006) according to transportation statistics on 81 US urban areas. As a form of shared ride system, the traditional mass transit system is a cheap way to travel, especially for the long distance trip, but it is far less flexible and comfortable than other paratransits, such as carpooling and vanpooling, because of the fixed routes and time-schedules and is also challenged by massive infrastructure cost in lower density markets according to the research of Edner and Weiner (1982) and Colorni and Righini (2001).

Another relative flexible mass transit mode is shared taxis, such as Jitney in the United States and Canada, which have fixed routes but are able to stop anywhere to pick or drop passengers. Different to general taxi service, shared taxis provide many-to-many services but are usually restricted to downtown cabstands and major transportation terminals, like airports and intercity train stations. Cervero (1997) studies several shared taxi cases in the United States, and concludes that shared taxis always need an informal arrangement between unrelated parties waiting at cabstands heading to the same destination. Cervero (1997) also points out that shared taxis are rarely marketed where customers are highly time sensitive, or zonal fares or flat fares are allowed.
2.1.3 Dial-A-Ride

Dial-a-ride is an alternative to traditional public transportation systems, in which users call a control centre to request a ride. Compared to traditional fixed-route mass transit systems, dial-a-ride systems are more flexible and comfortable, as they can provide door-to-door services by commercial vehicles and taxis (Colorni and Righini 2001). To utilize the vehicles’ passenger capacity, drivers can pick up other passengers before reaching the destination of the first customer. There are three basic schema of dial-a-ride systems (Colorni and Righini 2001): 1) group taxi (do not need previous calls, the driver negotiates with the customers and plans the route accordingly); 2) static dial-a-ride (customers ask for service in advance, and trip plans are made before the route starts); 3) dynamic dial-a-ride (customers call in during services, and the current plan is re-optimized). Colorni and Righini (2001) and Noda et al.(2003) concern on the modeling of dynamic dial-a-ride systems that support a many-to-many service: customers have different departures and destinations. However, the coordination of vehicles and customers becomes difficult if the number of dynamic demands is large. Therefore, dial-a-ride systems are not able to cover an unconstrained area.

2.1.4 Web-based Shared Ride Applications

Shared ride services are also available on-line. Google Ridefinder (Google 2005) provides a real-time approach to individual users to find a ride in local areas. Users have wide choices from taxis, limousines and shuttles, which are contract companies with Google. The locations of vehicles in this service, observed by GPS and collected in a central database, are said to be less than 5 minutes old, which practically means the locations are correct within 2-3 km. Currently, this service only works in a few metropolitan areas in the United States. Using Google Map, users can view the potential host vehicles by entering city names or addresses and call selected service providers to request a ride. But because only locations of these vehicles are provided in this interface,
users do not know whether the shown host vehicles have free passenger capacity for them unless called.

Other shared ride applications provide textual web interfaces to attract registrations of shared ride a client and ride hosts, such as Ride Now! (Wash et al. 2005), RidePro3 (Trapeze Software Group 2006) and eRideShare (eRideShare 2006). The applications are maintained by local and regional agencies with central databases. Mediated trips are usually regional or national travels, with inner urban travels generally not catered for. To request or offer a ride, users (a client and hosts) need to provide their home addresses, cell phone number, email addresses and requested trip details. Then the databases match requests and offers immediately, and feedbacks a contact list of potential shared ride hosts or a client. The choice is left to the users who can email or call their selections. Agencies need high-powered workstations, database servers and internet connectivity to run such an application. Personal computers or mobile devices with Internet connectivity are necessary as data terminals for the users.

2.2 Agent-based Transportation Simulation

This section introduces simulation methodology of peer-to-peer shared ride systems. Almost all transportation simulation models describe dynamic systems, where transportation agents change continuously. Discrete simulation models are a method to present either continuous or discrete changes of state by tracking changes at points in time. Lieberman and Rathi (1997) classify transportation simulation into two models: discrete time, where changes of state are represented in a succession of known time intervals; discrete event, where agents are recorded when their states are changed. Their comments on choosing discrete simulation models are that for systems where most agents are continuously changing state and where detailed description of these changes is required, the discrete time model is likely to be the better choice. Each agent in shared ride systems changes location, and details of negotiation processes are concerned with
the peer-to-peer circumstance, therefore simulation of a peer-to-peer shared ride system in this thesis is designed as a discrete time model.

Simulation models can be also classified into microscopic, mesoscopic and macroscopic according to the level of detail about the systems to be presented (Lieberman and Rathi 1997). In microscopic models both agents and their interactions are described in the most high fidelity; in mesoscopic models, agents are described in a high level of detail, but with their interactions at a lower level of detail than microscopic models; macroscopic models describe either agents or their interactions in the lowest fidelity. Low-fidelity models are less costly to develop and maintain, but they have a risk of representing the real world less accurately. Due to the sensitivity requirement of testing different communication and way-finding strategies, it is decided that the simulation model in this thesis is a microscopic model.

To develop an efficient model, the following sections review the content in agent-based simulation, particularly in the transportation circumstance. Some toolkits for transportation modeling and simulation applications are also addressed in later sections.

2.2.1 Multi-agent Systems for Transportation Simulation

In computer science, a multi-agent system (MAS) is a system comprising multiple autonomous and intelligent agents, which are capable of perceiving their environment and acting upon that environment to achieve their goals (Ferber 1999; Russell and Norvig 2003). The main applications of MAS at the moment are problem solving, multi-agent simulation, construction of synthetic worlds and collective robotics. According to their study, a good behavior is measured by how successfully the agent’s action hits its goal or how happy the agent feels its performance. Russell and Norvig (2003) give out four factors involved in design a rational agent: the performance measure defining the success criterion; the agent’s prior knowledge of the environment;
the available actions of the agent; and the agent’s percept sequence to date. Raubal (2001) indicates in his article that identification of the agents’ knowledge and beliefs is important during design the simulation. And to recognize what actual information is needed before modeling is also essential. Due to its ability of reflecting human behavior, Burmeister et al. (1997) propose the application of MAS approach in traffic and transportation systems, which naturally characterize “geographically and functionally distributed” (p. 52) subsystems, such as traffic management, traffic guide and control, and capacity and resource management. Particularly when traffic congestion becomes a world-wide problem, how to find an effective way to model and predict traffic flow has found the attraction of researchers (Bazzan et al. 1999).

2.2.2 Cellular Automata and Geographic Automata Systems

Cellular automata (CA) are popularly applied in agent-based modeling. CA arrange individual automata in a cellular space, where each cell has its state. Automata can collect information from their neighbors, and change their states according to their neighbors’ states and transition rules. Due to its simple structure, CA have proven their success in land use and urban planning. However, CA are inefficient in representing mobile agents, because cells themselves cannot move (Benenson and Torrens 2004). Additionally, state transition is too simplistic for implementing the negotiation processes in shared ride systems.

Benenson and Torrens (2004) combine CA and MAS concepts and extend them as geographic automata systems. Geographic automata systems are multi-agent systems in which agents are distributed in space, are able to exhibit autonomous behavior, in particular to re-locate, and interact with each other. States of geographic automata, including their location, are influenced by their neighbors, and the behavior of automata is specified by state transition rules. A peer-to-peer shared ride system can be seen as a geographic automata system, because it has states, and state transitions, in particular
finding a ride, depend on the neighbors.

2.2.3 Agents in Transportation Systems

Transportation systems are complex dynamic systems where particles, or agents, in traffic simulation are intelligent: they have strategic goals and they have an internal environment perspective around which to pursue these goals (Nagel 2003). As this thesis intends to model a microscopic transportation simulation, the internal intelligence and adaptability of agents and interaction between them are discussed in this section. In wider context of traffic simulation, agents include travellers, traffic signals, traffic management centres etc, while under the concern of peer-to-peer shared ride systems, agents in this thesis are defined as transportation demanders (a client) and providers (hosts) only. Nagel (2003) discusses several aspects in the design of intelligent agents such as: route generation, activity generation, learning framework, route replanning and private knowledge, while no transportation simulation integrates all the aspects. The following research shows the interest on agent identification, formalization and agents’ route generation in transportation systems.

Mataric (1994) addresses interaction and intelligent agent behaviors in complex domains, such as traffic. She defines behavior as “a control law that clusters a set of constraints in order to achieve and maintain a goal” (p. 18). A process of selecting a basic behavior set to generate robust group behaviors, such as following and homing, is presented in this research. Homogeneous mobile robots with ability to learn complex tasks by reinforcement (e.g. foraging) are used in experiments. Due to the reinforcement learning framework, homogeneous robots will make different decisions based on their interactions with the environment.

Guidi-Polanco et al. (2005) present a case study of a passenger transport planning system, in which transportation agents are identified in two layers: the internet layer for
communicating with the external world, and the scheduling layer for scheduling and assignment services. There are three agents in the internet layer: 1) vehicle agents that represent real world transportation vehicles; 2) broker agents that receive the messages from vehicles, and register them in internal database; and 3) Client Agent that capture human users’ requirements and translate them to the service. The scheduling layer also consists of three agents: 1) schedule agents representing route plans of single vehicles; 2) trip-request agents dealing with a client requests; and 3) scheduler agents implementing the assignment and scheduling policy.

Location and routing of agents are two basic components in ride sharing services. (Camp et al. 2002) discuss seven mobility models of entity agents for ad hoc networks: 1) random walking (random direction and speeds); 2) random waypoint (including pause time between changes in destination and speed); 3) random direction (travel to the edge of the simulation area before changing direction and speed); 4) boundless simulation area; 5) Gauss-Markov (using a tuning parameter to vary randomness); 6) a probabilistic version of the random walk (using a set of probabilities to determine the next position); 7) city section (a simulation area representing the street network). In a simplified case, the grid network can be used to represent the street network, and agents can be distributed in the network with random destination. Besides random movement, Leigh (2006) proposes a more realistic mobility model, in which agents prefer travelling on a particular path with an unequal probability distribution in an urban street network.

2.2.4 Simulation Toolkits for Transportation Systems

Several established agent-based simulation libraries exist for modeling transportation systems. Swarm is one of the popular libraries based on Objective C and has a Java wrapper. Repast is a newer Swarm-like conceptual toolkit (North et al. 2006). Repast is a free open source toolkit core in Java, while it has three implementations in Java, .Net and Python. Both approaches support the programming of multi-agent systems that are
composed of larger numbers of agents with functions describing their behavior. RePast was for example used successfully for a large-scale peer-to-peer shared ride system simulation (Tessmann 2006). However, installing and using libraries is in itself a larger effort, and this thesis develops an independent system from scratch.

Object-Based Environment for Urban Simulation, OBEUS, has been developed as a simple implementation of geographic automata systems in .Net (Benenson et al. 2001). It is designed for urban processes and has a built in cellular automata model with transition rules in form of functions. Entities in OBEUS can be one of two types, mobile and immobile entities. In OBEUS no direct relationship is allowed between non-fixed objects. That means that OBEUS is not suitable for our simulation of locally communicating mobile agents.

The Transportation Analysis and Simulation System (TRANSIMS) (Smith et al. 1995) is a project funded by the U.S. Department of Transportation and the Environmental Protection Agency. TRANSIMS is an agent-based simulation approach to create an integrated regional transportation systems analysis environment. This system is capable of tracking dynamic (every second) movements of agents (person and vehicle) through a large transportation area. Without a license, TRANSIMS is not available for the research in this thesis.

Another toolkit named Agenda (Fischer et al. 1999) is designed to provide cooperation-scalable methods based on negotiation between truck agents and the exploiting company. Agenda does not provide communication between truck agents, and another disadvantage is additional board computer equipment needed on trucks, together with sensors broadcasting traffic information needed along roads.

2.2.5 Dynamic Transportation Applications

This section introduces several dynamic transportation applications, and focuses on how
these applications are designed to deal with real-time traffic data. Wu and Miller (2001) report a computational tool to measure travel accessibility with a dynamic network-based space-time prism within time-varying traffic flow, such as traffic congestion scenarios. Unlike classical and network-based space-time prisms, dynamic network time prisms (DNTP) define the travel time between two locations varying on both space and time. Their work did not include activity locations and participation time for individual activities in the calculation of DNTP.

Dillenburg et al. (2002) propose the intelligent travel assistant to combine location and planning technologies and transportation information into a device to reduce congestion and increase the efficiency of transportation network. Such applications need user input (e.g. current location, destination, ranking criterion for plans and budget) to a traffic information centre, which use spatio-temporal query languages, indexing to find potential travel plans, sorted and filtered by the user’s preferences to provide associated trip cost, pickup time, and estimated arrival time. Real-time traffic information is collected and analysed depending upon the information centre.

EasyTransport (Fragouli and Delis 2005) is an on-line navigation transportation system, which provides travellers with multiple near-optimal trajectory transportation options along a route. This system is assessed on merits at response time, accuracy and multiple outlined trajectories, but all data (e.g. transportation information and local maps) reside in the system’s back-end database, which at the time of publication does not incorporate real-time transportation features.

A self-organizing dynamic vehicle navigation system is presented by Yang and Recher (2006). This system allows vehicles with specific inter-vehicle communication equipment to share traffic information, and incorporates self routing based on real-time and historical traffic information, and trip replanning according to exchanged dynamic traffic conditions. The raw real-time travel information is stored and processed in
individual vehicles from their own perspective. When two vehicles are very close at a particular time, their knowledge of current network traffic conditions are updated. Each vehicle is able to find the shortest path to its destination from the real-time traffic condition. If the travel area is not covered by real-time traffic information, historic information is used instead.

### 2.3 Shared Ride Trip Planning

Trip planning is critical in shared ride systems, because both a client and hosts have their own perspective, knowledge and preferences to meet (Tempich et al. 2004). To find an optimal solution, complete knowledge, including the static street network, the dynamic transportation network composed by transportation hosts and the communication network formed by broadcasted messages, is required. However, it is impossible to obtain complete knowledge of all vehicles current and prospective in street network. Without central service, peer users in peer-to-peer shared ride systems need to make decisions depending only on local knowledge from their neighbors. The limitation increases the risk of making a sub-optimal decision. Gaisbauer (2006) discusses the route-choice strategies for shared ride trip planning to avoid gaps between intermediate rides along the route to destination. This section overviews algorithms and solutions in trip planning context that starts from a single shortest path problem and extends to multi-criteria issues.

#### 2.3.1 Shortest Path Algorithm

Finding the shortest path between two nodes in a graph arises when a client desire minimal travel cost, which can be assessed by the weight of edges in the graph, such as travel time or travel fare. In particular, for shared ride agents this problem is a single-source shortest path problem. The well known single-source shortest path algorithm is Dijkstra’s algorithm (Dijkstra 1959). Given a weighted graph $G$, a cost
function $F$ and a start node $s$, the algorithm first updates the cost of all neighbors of $s$ and stores them into a container $Q$, then starts from the one of the neighbor nodes $n_i$ with the lowest cost, and updates the costs of its neighbors and stores them into $Q$; when all nodes go into $Q$ the computation stops. Dijkstra’s algorithm is the best for finding shortest paths from one node to all other nodes in the graph. But for an individual agent with a particular destination in shared ride systems, this algorithm is inefficient because it explores unnecessarily large area.

The A* algorithm (Hart et al. 1968) is a modified Dijkstra algorithm, which introduces a heuristic cost function to control the search to a particular destination and is efficient in one-to-one shortest path searches. The key element of A* algorithm is the function:

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

where $f(n)$ is the cost of the node examined by depth-first search, $g(n)$ is the cost of travelling from the start node, and $h(n)$ is the heuristic cost function.

The heuristic function is an estimated cost from each node to the goal node and its definition always affects the efficiency of A* algorithm. The original A* algorithm is not designed for moving objects or real-time traffic trip planning. Koenig et al. (2004) then developed a Lifelong Planning A* (LPA*) algorithm to find the shortest path from a fixed start node to a fixed goal, when the costs at the edge of the network are changed.

The first search of LPA* algorithm is the same as that of A* algorithm. Subsequent searches then reuse the result of the previous search potentially leading to a faster than uniform search. To derive a dynamic shortest/fastest path for moving objects, Wu et al. (2005) extend the LPA* algorithm with a dynamic start point and constrained shortest path ellipse in the search area. The extended LPA* algorithm is demonstrated to save up to 70%-80% of the computational cost compared to the static A* algorithm.
In transportation trip planning circumstances, the time required between two nodes may vary depending on which vehicle is travelling or if congestions happen. Therefore, a time-dependent shortest path algorithm (Cooke and Halsey 1996) has been developed to suit a more realistic case where an initial starting time is set for each iteration through two nodes.

2.3.2 $K$ Shortest Path Algorithm

Sometimes agents would like to be given a set of optional paths instead of one optimal path from departure to destination, for instance when agents are not sensitive to one criterion, or more than one criterion is considered to balance the choice. In this case, the $K$ shortest path algorithm is designed to solve this multi-choice problem. Because Dijkstra’s algorithm is designed to find shortest paths starting from one node to all other nodes in the graph, it is not suitable to find $K$ shortest paths between two specified nodes. Yen (1971) presents an algorithm for finding $K$ shortest loopless paths from one node to another node in a network. It is assumed that at least one shortest path exists in a network. In order to find the $k^{th}$ ($k = 2, \ldots, K$) shortest path $A^k$, the previous $k-1^{th}$ paths $A^1, A^2, \ldots, A^{k-1}$ are determined. Each time an edge in coincided subpaths is removed from previous paths, a new shortest path is calculated based on the changed network and added in a ranking list that includes all previous shortest paths. Yen’s algorithm is proved to be the best result to generate $K$ shortest paths (Lawler 1972). In peer-to-peer shared ride systems, agents would like to consider a simple algorithm with lesser computational cost to save energy and computing time in order to respond to real-time information. For this reason, the algorithm in this thesis could come up with sub-optimal results of $K$ shortest paths.

2.3.3 Multi-Criteria Optimization

Considering more than one criterion, optimization problems are more complex.
Multi-criteria optimization, also called multi-objective optimization, is a solution to select an optimal offer under multiple criteria (e.g. travel time and fare) in shared ride trip planning.

For example, agents consider travel time as well as travel fare at the same time to choose from offers. This section reviews techniques in the multi-criteria optimization problem. In this problem, each solution can be represented as a vector that projects objectives on various axes. Solutions, which are non-dominated (the Pareto-optimal set), are considered as better solutions than others (Costelloe et al. 2001). However the Pareto-optimal set does not always provide only one solution. To solve this problem, multiple objectives are almost always combined into one scalar objective whose solution is an original Pareto-optimal point. This process has been developed by linear and non-linear techniques. The main techniques are described as follows:

- The weighting sums method: uses a user-defined weighted function to scalarlize a set of objectives into a single objective. By minimizing the sum, this linear technique provides the user with more information about the trade-off among the various objectives. However the setting of appropriate weights is more challenging. The weight of an objective is usually chosen in proportion to the objective’s relative importance in the problem. Also the weights depend on the scaling of each objective. Haque et al. (2006) apply this linear technique to balance the reliability and distance of paths.

- Homotopy techniques (Rao and Papalambros 1989): traces the complete Pareto curve (made by continuous Parate points) in the bi-objective case, which overcomes the sampling insufficient problem in the weighting sums method. But this approach is not suitable to the case of more than two objectives.

- Goal programming (Romero 1991): minimizes one objective while constraining the remaining objectives to be less than given target values. This method is
useful if the user can solve just one objective optimization problem, but the choosing of an appropriate “goal” is difficult especially if the number of objectives is more than two.

- Normal-boundary intersection (Das and John E. Dennis 1996): this method produces an completed even spread Pareto surface, where any point is able to find a set of weights so that this point minimizes a weighted sum of objectives. However, Pareto points in shared ride systems are not continuous, therefore NBI is not suitable for the shared ride trip planning problem.

- Multilevel programming (Bard and Falk 1982): orders the objectives in terms of importance, finds the points with minimum value in the first objective, and then among those points finds the ones that minimize the next objective. The method processes until all objectives are optimized or one point rises. This approach is useful when users have a hierarchical order of importance among objectives, which is likely in the circumstance of shared ride systems. But less important objectives may have no influence on the final result.

### 2.4 Preliminary Work on Peer-to-peer Shared Ride Systems

Previous sections have investigated current applied shared ride systems, which have the following disadvantages for ride sharing in general:

- Requirements of shared ride demanders are not well met by ride providers limited temporally and/or spatially.

- Provided shared rides are not flexible and convenient enough.

- Participants have difficulties in organizing and coordinating share rides.
Shared ride services are limited in area or number of users.

Roussopoulos et al. (2004) propose critical criteria (e.g. low budget, resource relevance to participants, trust, and rate of system change) of how suitable a peer-to-peer solution might be for a particular problem. Under these criteria, a peer-to-peer shared ride system is a scenario to overcome these disadvantages. In this scenario, no central service is needed and transportation demanders and providers communicate directly and plan shared rides by themselves. Preliminary work (Winter and Nittel 2005; Winter and Nittel 2006; Wu et al. 2006) has demonstrated the feasibility and efficiency of this scenario.

The first published paper on peer-to-peer shared ride system is Winter and Nittel (2005), who propose a shared ride system trip planning model on ad-hoc mobile geosensor networks. This model is designed to solve the problem of capacity limitations of centralized travel planning system to large amount of data in large dynamic networks. They demonstrate that without a central service, shared ride trip planning with limited knowledge is possible and computationally efficient in a dynamic environment. Winter et al. (2005) then implement this scenario with a simulation, in which a client with transportation demand, and hosts with transportation supply communicate on a radio base to negotiate and plan trips in a continuously changing environment. This research designs a mechanism for the negotiation process and investigates three communication strategies with different communication neighborhoods. This model is further implemented by Winter and Nittel (2006) with homogeneous hosts and an immobile client. Winter and Nittel (2006) conclude that a mid-range communication strategy in mobile geosensor networks is both effective (leading to travel time comparable to complete current knowledge) and efficient (leading to less communication messages than those for complete transportation knowledge) compared to unconstrained and short-range communication. However, this simulation is limited in an inflexible way-finding strategy (a client follow the shortest distance route) and simplistic
behaviors of agents (all hosts behave homogeneously and the immobile client looks for the quickest trip only). Wu and Winter (2006) then extend this simulation into an agent-based model with multiple types of agents, and illuminate that types of agent enrich the choice of shared rides and result in shorter travel time.

Other work includes a time geography method for efficient shared ride trip planning (Tessmann 2006), agent mobility model based on street centrality (Leigh 2006) and heuristic routing strategies to intermediate location choice (Gaisbauer 2006). Another paper (Wu et al. 2006) presents preliminary results of this thesis. This thesis will elaborate on the design of a multi-agent peer-to-peer shared ride system and develop an extendable simulation model with more complex mobility and trip planning behaviors of agents.

In summary, the previous work on peer-to-peer shared ride systems has constraints that are far from the reality of urban traffic. This thesis will implement a more realistic scenario with diverse agents, investigating whether the scenario will confirm or even improve the preliminary results of the existing model.
Chapter 3

Agents in Peer-to-peer Shared Ride Systems

In peer-to-peer shared ride systems, agents are particularly considered as transportation demanders and providers. Diverse in their roles, the demanders and providers are called Client Agent and host agents separately in this thesis. Both of them are intelligent agents, who are able to perceive their environment, collect data, communicate with other agents, and make decisions based on collected knowledge. Their behaviors directly influence the processes of negotiation and resultant quality of shared rides. Understanding their behaviors is essential not only for designers to employ efficient communication protocols and way-finding strategies, but also for peer users to make appropriate decisions. While the natures of agents are various and extensive in the real world, it is necessary to identify the critical factors contributing to shared ride systems before modelling such systems in simulation. For this purpose, this chapter investigates agents’ mobility, passenger constraints, economic and operational characters and their knowledge in the circumstance of shared ride systems.

3.1 Client Agent

Client Agent is an agent who has transport demand but no vehicle, and depends on the rides from others to reach its destinations in shared ride systems. Client Agent is called client for short in this thesis. In the real world, pedestrians and hikers can be regarded as a client. A client’s desire of traveling causes the negotiation between a client and hosts in shared ride systems; additionally their destinations induce the routing of shared rides.

A client also has preferences about hosts and routing. For example, a rushed client prefer quicker hosts directly heading for destinations such as taxis, while a budget client favor cheaper hosts no matter how they make detours, e.g. buses. On the other hand, a
client may be willing to travel in a non-smoking circumstance, or with an experienced driver. Such preferences are also impact factors to negotiation processes.

Apart from their preferences, a client is also diverse in terms of mobility and knowledge. The following subsections discuss a client with two mobility manners: immobile and mobile client, and discuss how a client with various mobility and level of knowledge behave in shared ride systems.

3.1.1 Immobile Client

Due to its physical status or the weight of carries, a client may be not willing to move by feet. This kind of client completely depends on the rides offered by hosts to move, which means that this kind of client have to wait and stay at the current positions until rides are available. This kind of client is called immobile client.

The trip of immobile client is composed by shared rides only. They are always in one of the two states, either waiting or moving with a host. As constrained by taking available rides only, immobile client is expected spend most travel time on waiting especially when the contributing hosts are rare, for example hosts in a low density or distributed remotely. In the extreme case this disadvantage may result in an unfinishable travel. However, this disadvantageous situation could be moderated when the client demands destinations reached by routes with high frequency of hosts, for instance a shopping centre beside a bus station.

3.1.2 Mobile Client

A client, who is able to walk, is called mobile client. Besides the two states of immobile client, mobile client can be under a third situation, walking, during its way. Mobile client is more flexible than immobile client, because mobile client can still move by its feet when there are no rides available. In a particular case, mobile client can even walk
to the destinations by its own. Therefore, it will not happen that a mobile client fails into an unfinishable trip. Mobile client is expected to achieve shorter travel time against immobile client in general, because mobile client will benefit from more optional rides within the range of walking permitted.

From the economic view, walking is the cheapest way to travel while taking rides may charge the client. But the client should be aware that the speed of walking is normally far slower than the speed of vehicles. Additionally to decide to walk, the client has the risk of missing oncoming rides when leaving places for pick-up, and result in longer travel time.

3.1.3 The Level of Knowledge of a Client

Another factor affecting a client’s behaviors is the level of knowledge. Its mobility, preferences and ability of planning trips are all considered as part of knowledge. Particularly, this section focuses on the environmental knowledge, what a client utilizes to analyze the collected data and make decisions. For a client in peer-to-peer shared ride systems, its decisions are especially about how to choose rides from offers to suit its purpose. The basic purpose of a client is to reach the destination; additional purposes can be travel under its preferences, such as efficiency or economy requirements. In the real world, a client has different level of knowledge, and the limitation of knowledge affect the quality of trips. A demonstration case is that local people are likely quicker than tourists by better guessing potential rides with its knowledge of local traffic when asked to the same destinations. The environmental knowledge of a client can be identified as local knowledge, street network knowledge and traffic pattern knowledge from lower to higher levels.

The lowest level of environmental knowledge is that a client knows its current position and destination only, so called local knowledge. Such low level of knowledge usually
happens on a client who is very unfamiliar with the traveling area, for example tourists. Transfers are the problem for the client when hosts do not go to the client’s destination directly. In this case, hosts knowing the destination and able to plan trips, such as taxis drivers, can meet the requirement.

If given a map, tourists can learn the street network and consider transfers. By taking transfers, a client has more options because individual rides may not reach the destination but bring a client approaching the destination one by one. Additionally, a client with the higher level knowledge of street network is able to plan its trips in advance and request rides along the predefined routes. However, the client may have problem of finding the next transfer when requesting a route out of main streets.

Correspondingly, a client having knowledge of traffic pattern, such as traffic distribution and commute routes upon street network, more likely make a good guess. For instance, the local people are familiar with the distribution of bus lines and the frequencies of bus, therefore drop-offs around bus stations are considered having more opportunities to find the next transfer than other places. Additionally, the knowledge of peak traffic time and distribution is helpful to avoid congestions. It is expected that a client with higher level of knowledge has more advantages for trip planning.

3.2 Host Agents

Host agents are normally considered as a combination of two components: the driver and the vehicle in circumstance of shared ride systems. The driver component controls the movement of vehicles, and has abilities of negotiating and making decision. Hosts’ behaviors vary corresponding to the employment of different vehicles. Vehicles can be any transportation containers furnished spare seats to occupy, such as trains, trams, ferries and airplanes. The vehicle component has relevant attributes, such as speed, type, seat limitation and so on. In this section, three typical hosts are identified to discuss
their diverse behaviors, economic and operation models in the real world.

### 3.2.1 Mass Transit

The term *mass transit* is used in North America, while *public transport* is used in British Isles and most Commonwealth countries. In many cases, transport systems serving the public are owned by private providers, therefore the term *mass transit* is chosen for general in this thesis. Mass transit comprises all transport systems where passengers do not travel with their own vehicles, such as buses, trains and trams, subway, and ferries. It would include air line services. Given a further restriction that mass transit should be shared carriers, it excludes taxis which do not run on a shared mode. Generally, mass transit vehicles supply a larger passenger capacity compared to other means of transport, although with less comfort and privacy. Travel fares are relatively cheap, especial for a long distance travel. Some regions provide free mass transit services, such as airport connectors. Discount fares usually are available for senior or junior passengers. Tickets must be bought in advance or on board, which give the passengers a single or unlimited travels within a period of time. Frequently fares are charged by time regardless how far to travel, but other payment systems exist as well. Various tickets may be required on different modes of mass transit, while in some areas multi-use tickets are employed as well.

Mass transit follows a regular schedule, typically with larger gaps between midnight and early morning and varying frequency over the day. Usually they run on a fixed route back and forth, and passengers are only allowed to get on or off at stops. This means that mass transit does not provide door-to-door transport, nor does it reach some areas in the city at all. Separate infrastructure can make mass transit faster than transports running on common roads, particular samples as light rail and subway, where traffic jams can be avoided. However, other mass transit, e.g. buses and trams, that travel on common roads are normally slower than private transport, because of an initial waiting,
frequent stops, traffic lights and congestions.

3.2.2 Taxicabs

Taxicab, short terms taxi or cab, is another popular means of transport for a single passenger or a small group, typical for individual occupation. Taxicabs are more comfortable and convenient compared to mass transit. Taxicabs can be hailed on the street by passengers as a taxi passing by, or on appointment stands by calls. Taxi services are usually available at any time of the day. Most experienced taxi drivers working in the same region for long are expected to know the major routes and most important places where customers might want to go. Equipped with navigational systems and supported by a control center, taxicabs are able to meet other requests and avoid road congestions. Passengers can head directly for their destinations without compulsive intermediate stops or transfers. Detouring, change of destinations and stopovers are also possible during travel.

The main disadvantages of taxicabs are limited passenger capacity, and corresponding higher trip fares compared to mass transit, especially for distance travel. Normally, taxicabs are furnished with about four seats for passengers, but these are only shared for a group having the same trip. The fare does not depend on the number of passengers traveling together in a taxi. Fares are usually calculated by a flag fall and a combination of travel distance and waiting time measured by a taximeter. When highly demanded or at particular times, such as midnight or early morning, taxicabs will pick up the passenger who offers the highest fare. These attributes mean that taxis are more suitable for individual travellers or small groups travelling together, either for short trips, or when time or convenience is more valued than money.
3.2.3 Private Transport

Opposed to mass transit, private transport takes place in one’s own vehicle, such as a car, motorcycle and bike. The differences of private transport from mass transit are no fixed timetables and itineraries. Private transport frequently has its own travel aim and/or pre-designed route, for instance a commute trip between home and work place. Except for ownership, private cars in their function as hosts for shared rides are similar to taxis: they share the advantage in comfort, and the disadvantage in low passenger capacity. That means that private transport can pick up a client along their trip, but is unlikely to make a detour, and may only serve part of the client’s route. As the drivers own the vehicles, private transport are considered as private space or proxemics (Hall 1966), where drivers might have more rigid interests and preferences in selecting a client, such as non-smoking a client, or a client of a specific gender.

Compared to taxi fares, a ride in a private car could be free, if the incentives for the car drivers are non-monetary, such as being allowed to use high-occupancy vehicle lanes. Alternatively, they can charge proportional to the traveled distance, but to lower rates than taxis because their interest is mostly in sharing costs.

Catching a lift from private transport, called hitchhiking, lifting or thumb up a ride, is an alternative transport method when no mass transit or taxicabs are available for a client. In almost all countries in the world hitchhiking is legal, such as Australia and European countries, while in North America hitchhiking or signaling for a ride is forbidden in some areas for security reasons.
Chapter 4

Formalization in a Multi-Agent Simulation

So far, theoretical considerations applied to shared ride agents, classified by type and relevant behavior, were discussed. To build on this, relevant case studies in the real world are needed to validate the proposed shared ride scenarios and analyse their performance. In this section, a computer simulation is designed for this purpose.

The preliminary work by Winter and Nittel (2006) presents a simulation model for peer-to-peer shared ride trip planning with homogeneous hosts and an immobile client following a geodesic route from the origin to the destination in large transportation networks. They have demonstrated that the mid-range communication schema is both efficient and effective compared to unconstrained and short-range communication. However, their research does not consider various types of hosts and the reality of the client, and the effect of their diverse behaviors on the shared rides. The purpose of this chapter is to develop a more realistic simulation model with types of agents to investigate how their behaviors under different communication and way-finding strategies will influence the shared rides. This model presents the features of various agents and their behaviors as discussed in Chapter 3 by advancing the specification of agents, and modifying their trip planning behavior accordingly. In addition, it is also designed for deeper investigation on shared ride agent behaviors under specific communication and way-finding strategies.

Due to the high fidelity requirements of agent behaviors (i.e. the negotiation process of shared rides), this model is determined as a microscopic model. This model describes the simulation environment and the states of agents in discrete time in response to the continuous change of elements (e.g. the number of communication messages and the position of agents) over time. The object-oriented language Java is chosen to implement
the extensible model of agents with more complex features and behaviors in further investigation. All transportation agents are developed from a basic class – Agent. A client and host agents are represented in Client class and Host class respectively, which formalize a client and host agents’ features with parameters and behaviors with functions. Henceforth, Client class is called Client, and Host class is called Host in this chapter. Results of the simulation are collected for trip quality analysis (Guan 2007). Due to the expected large amount of data reading/writing by agents and the lesser amount of data query, the shared ride information is recorded in files for each agent instead of a database.

This simulation model includes the following components (Fig 4-1):

- Simulation environment: initialize the transportation network, generate agents, specify the communication and way-finding strategies, synchronize their behaviors and evaluate the performance of shared ride trip planning.

- Communication protocol and strategies: define the form of negotiation information shared by agents and the way that the data transfers among agents.

- The negotiation mechanism: define the process of negotiation of shared rides.

- Agents: have knowledge of their environment, are able to communicate with other agents and negotiate shared rides, and employ a particular way-finding strategy according to their type and preferences.

- Simulation model assessment: define the criteria for simulation performance assessment, and collect/acquire data for agent behavior analysis.
In the following subsections, these components are discussed in order.

**4.1 Simulation Environment**

In the proposed peer-to-peer shared ride system, agents have knowledge of their locations within the street network, negotiate with their neighbors for shared rides, make decisions according to their desires and intentions, and travel until the next negotiation takes place. Such system can be seen as a geographic automata system according to the definition of GAS by Benenson and Torrens (2004). To implement geographic automata systems, Benenson and Torrens (2004) suggest establishing a spatially restricted network with “fixed” (immobile) and “non-fixed” (mobile) agents, neighborhood relationships and behavior rules. Due to their interest on urban objects, such as buildings or residential addresses, they use a cellular network. In contrast, agents in shared ride systems move in street networks, and hence, a grid network is used to model the street network, composed by nodes representing street intersections where agents can stop or meet, and edges representing the roads which agents can travel along. In this stage, the grid network is a simplified form of street network in the shared rides environment, however there is no reason to expect changes in the qualitative results to
justify use of the more realistic street network.

The width (number of columns) and length (number of rows) of the grid network are specified in environmental parameters. The coordinate pairs \((x, y)\) of nodes are represented by the position of columns and rows. Since coordinates form a primary key, there is no additional identifier (ID) field required for nodes. Although using identifiers could save data store space, an extra reference list between identifiers and coordinates would be needed resulting in additional computations. Considering that the length of messages is not a concern in the simulation and a large amount of network computations is expected, coordinates are designed in the form of nodes only. Nodes are specified in Table 4-1.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(x)</td>
<td>float position in grid columns</td>
</tr>
<tr>
<td>2</td>
<td>(y)</td>
<td>float position in grid rows</td>
</tr>
</tbody>
</table>

Table 4-1: Node features

Edges, as connection between neighboring nodes in this grid, have a length of unit size. Because all edges have equal length and two end nodes can locate an edge in the grid network, there is no class designed for edge. The dimension of the grid network is scalable by setting its width and length in terms of numbers of columns and rows.

At the beginning of each run, the simulation environment is “empty”: there is no agent. The type and number of various agents are specified by parameters, then hosts and a client are generated with nature and knowledge in groups until the number of agents reaches the maximum quantity. To compare the results from different runs, negotiation proceeds under a stable density and ratio of the types of agents. That means if any agent disappears from the environment (e.g. arrive at destination and finish trip), a new agent of the same type is generated instead. When the client arrives at its destination, the
negotiation process is completed and the particular run is ended.

Other parameters of simulation environment are given as follows:

- Communication range: the size of the communication window.
- Way-finding strategy: a client will follow the geodesic route or any route not excluding detours.
- An internal clock: synchronizes the behavior of agents.
- The total number of negotiation messages.
- The time of finishing the client’s trip.

The content of the simulation environment is defined in the class simWorld as shown in Table 4-2. Agents can refer an instance of the class as environment knowledge.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 length</td>
<td>int</td>
<td>the number of rows of simulation grid system</td>
</tr>
<tr>
<td>2 width</td>
<td>int</td>
<td>the number of columns of simulation grid system</td>
</tr>
<tr>
<td>3 unit</td>
<td>int</td>
<td>length of a grid edge</td>
</tr>
<tr>
<td>4 comRange</td>
<td>int</td>
<td>the communication range: the number of edges</td>
</tr>
<tr>
<td>5 clientNum</td>
<td>int</td>
<td>the number of a client</td>
</tr>
<tr>
<td>6 hostNum</td>
<td>int</td>
<td>the number of hosts</td>
</tr>
<tr>
<td>7 msgNum</td>
<td>int</td>
<td>the counter of the total number of broadcasted messages</td>
</tr>
<tr>
<td>8 time</td>
<td>int</td>
<td>current simulation time (starts at 0)</td>
</tr>
</tbody>
</table>

*Table 4-2: Simulation environmental features*
4.2 Communication Strategies and Protocol

Direct communication between peer agents is especially important in peer-to-peer shared ride systems, shared ride information for each agent is obtained wholly by communicating with their neighbors. For example a client finds a ride to his preference by listening to offers from his neighbors. To make optimal decisions, agents need to consider all transportation information. However, in dynamic traffic, an individual agent may not be able to reach or want to reach all other agents in the street network. This means that agents have to make decisions with local knowledge only. Therefore, how to communicate efficiently and effectively is the motivation to discuss communication strategies and protocol for shared ride agents.

Generally, radio is used as a feasible and economical technique for dynamic entities to communicate in a wireless network (Winter and Nittel 2005). The radio range is the radius of the reception area in which their neighbors can receive messages. The radio range is limited according to the broadcasting technologies, such as Bluetooth or WiFi. Distant agents can be reached by forwarding/re-broadcasting messages. This means that multi-hop broadcasting is needed for the purpose. For a peer-to-peer shared ride system, the synchronized time (communication window) all agents listen and broadcast through needs to be long enough to accomplish a complete negotiation process, consisting of a request, offers, and a booking. This means that from the previously investigated three communication strategies (unconstrained, short-range and mid-range) the unconstrained communication strategy is not feasible in reality (Winter and Nittel 2006). Unconstrained communication means that messages flood to the deepest agents in the network, as long as agents are connected \((comRange = \infty)\). The other two are local communication strategies. In short-range communication, agents only communicate to agents within their radio range (single-hop, \(comRange = 1\)). In mid-range communication, agents forward messages within several hops \((comRange > 1)\). The negotiation process will be simulated for different communication ranges to investigate trip
planning with different levels of transportation network knowledge. The performance of employing various communication strategies will be investigated and analyzed in later sections.

Agents also need to understand the received transportation/ride information from other agents. For this purpose, a communication protocol is established for all agents to share ride information. Nagel (2003) suggests that trip plans always include a start time, a start position, a destination and a sequence of nodes in between. In shared ride planning, agents are additionally interested in the agents involved in the trip, travel time and/or travel fee. Due to the common concern of both a client and hosts on shared ride related information, a uniform information/message format is defined for all kinds of negotiation messages from two groups in this protocol. Each message includes the particular type of message (e.g. request or offer), proposed ride time, route detail (e.g. the start and the end points), and agent detail (e.g. the speed of agent, and the transferring agents list). The message structure is specified in Table 4-3. With no constraint on the number of message packets transmitted between agents and device memory and battery performance in the computer simulation circumstance, the length of the message is not discussed in this thesis. More details of the communication model and protocol are discussed in Winter and Nittel (2006).

<table>
<thead>
<tr>
<th></th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>type</td>
<td>char</td>
<td>the type of message: request r; offer o; booking b</td>
</tr>
<tr>
<td>2</td>
<td>route</td>
<td>Vector &lt;Node&gt;</td>
<td>requested or offered route: the first node is always the start point, the last node is the destination</td>
</tr>
<tr>
<td>3</td>
<td>time</td>
<td>int</td>
<td>the expected start time of the route in the message</td>
</tr>
<tr>
<td>4</td>
<td>agentsList</td>
<td>Vector &lt;int&gt;</td>
<td>a list recording identifiers of agents who transfer the message</td>
</tr>
<tr>
<td>5</td>
<td>speed</td>
<td>float</td>
<td>(average) speed of the original sender of the message</td>
</tr>
<tr>
<td>6</td>
<td>cost</td>
<td>float</td>
<td>the cost of completing the route</td>
</tr>
</tbody>
</table>

*Table 4-3: Message features*
4.3 The Negotiation Mechanism

Having defined communication strategies and protocol, a mechanism is needed to process the negotiations. The principles of the negotiation process are depicted as follows:

- Negotiation mechanisms wake up.
- A client starts from his origin and looks for shared rides by sending out a request message.
- The client’s neighbors receive the request and forward it to their neighbors in the communication window. If any receiver is able to provide a ride (e.g. owns a vehicle with spare seats) corresponding to the request, the receiver sends out an offer message to the client.
- The client receives all offers, compares them to match his preference and chooses the optimal one to book. If no offer, the client requests again in the next negotiation process.
- The chosen host receives the booking message and heads to the appointment position.
- Negotiation mechanisms fall to sleep and all agents move.
- The client meets the host at the appointment and takes the ride.

These actions happen in turn and repeat until the client arrives at his destination. The necessary actions for all agents in the negotiation process are receiving/listening and forwarding messages from/to their neighbors. In particular, a client need request and booking actions; and hosts need offer action.
The principle shows that each negotiation process starts at a request from a client, proceeds by offers from hosts and finishes at a booking. These three main phases happen sequentially, which means that except for the first phase, request, any other phases cannot commence until the previous phase finishes. Detail explanation of these key phases can be found in the next section. It is assumed that the communication time and computing time are very short compared to the time that agents travel over to the next stage. Furthermore, the largest energy consumption of agents in geosensor networks is broadcasting compared to listening, computing, or sleeping. Therefore, each time stamp includes two parts: 1) agents’ communication devices wake up to negotiate shared rides; 2) the devices fall to sleep and agents move. After each negotiation the simulation clock increments. Because travels of agents can change dynamically, agents do not need to keep previous negotiation information in memory. In addition, there is no cancellation phase integrated, because booked rides are regarded as being cancelled if no re-booking/confirmation happens in the following negotiation cycle, or no a client/host shows up for an appointment. Figure 4-2 presents the negotiation mechanism as indicated.

![Figure 4-2: Structure of shared ride negotiation process](image)

So far, only one a client is generated in each individual simulation run ($clientNum = 1$). All hosts serve for this a client. In this case, hosts do not need to consider which a client to contribute to, and so that there is no competition among clients.
4.4 Modeling of Agents

This section describes how typical features and behaviors of agents in peer-to-peer shared ride system are selected, specified and modelled in simulation. This simulation model is designed for the concern of investigating various communication and way-finding strategies, and analysing agent behaviors in shared ride circumstance. Also, this model is available for other interests, for example trip quality analysis and security issues in shared rides. In order to investigate further interests on peer-to-peer shared rides, all agents are developed on an inheritance structure, which allows extending from existing work easily and simply. The types of agents are described in corresponding Java classes, and agents are generated as the instances of class in the simulation. Agents employ the communication protocol and follow the negotiation process as discussed earlier. Detail discussion will address on how to implement different behaviors under various communication and way-finding strategies for particular types of agent. The hierarchy of agent classes is shown in Figure 4-3. The basic class for all agents is described first.

4.4.1 The Basic Agent Class

All agents in peer-to-peer shared ride systems have the following properties:
• A name/identifier.

• Ability of moving at a certain speed of no less than 0.

• Knowledge about the current position and the destination heading to.

• Knowledge about the time.

• Knowledge about the communication protocol and strategy.

• Ability of listening and sending out negotiation messages to neighbors.

Clearly all shared ride agents have these features and behaviors. To build up a superclass for all agent classes to inherit from, some features and behaviors as described above need to be identified and encapsulated in a basic class. This basic class is named Agent. There are two criteria proposed to identify these features and behaviors:

• Can be found on each agent.

• Necessary/contributing for shared ride negotiation.

Having defined the selection criteria, the features and behaviors are formalized as the properties and methods separately in Agent class. These properties include the agent’s identifier, its speed, its type, its state, a reference to its current simulation environment, and information on its travel plan, such as the current position, the destination, and a temporary container of negotiation messages. The travel route contains origin and destination, and for some agents the nodes along the route. For investigation purposes, a second container stores detail of booked shared rides. Detailed descriptions of all properties and their formalization types are listed in Table 4-4. The methods include how to move to the next node, how to listen to neighbors and how to obtain knowledge about current position and state. The input and output data of methods are specified in
Table 4-5.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 id</td>
<td>int</td>
<td>unique identifier of agent</td>
</tr>
<tr>
<td>2 speed</td>
<td>float</td>
<td>speed of agent in terms of units per time window</td>
</tr>
<tr>
<td>3 type</td>
<td>char</td>
<td>type of agent: a client c; host h</td>
</tr>
<tr>
<td>4 time</td>
<td>int</td>
<td>time of the most recent shared ride</td>
</tr>
<tr>
<td>5 state</td>
<td>char</td>
<td>current state of agent: moving m; on a ride t; walking w; stopped e</td>
</tr>
<tr>
<td>6 position</td>
<td>int</td>
<td>index of the current position in route array</td>
</tr>
<tr>
<td>7 route</td>
<td>Vector &lt;Node&gt;</td>
<td>route array, the first element is the start point, and the last element is the destination</td>
</tr>
<tr>
<td>8 messages</td>
<td>Vector &lt;Message&gt;</td>
<td>container of receiving messages at each negotiation cycle. This list is updated when new messages are received, and it is cleared when a new negotiation cycle starts.</td>
</tr>
<tr>
<td>9 services</td>
<td>Vector &lt;Message&gt;</td>
<td>container of messages under consideration. For a client: detail of all offers for its remaining trip. For hosts: detail of all bookings received so far.</td>
</tr>
<tr>
<td>10 world</td>
<td>simWorld</td>
<td>reference to the current simulation environment</td>
</tr>
</tbody>
</table>

Table 4-4: Agent properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 move</td>
<td>null</td>
<td>boolean</td>
<td>move to the next node in route (abstract class)</td>
</tr>
<tr>
<td>2 listen</td>
<td>Message</td>
<td>null</td>
<td>update current message container</td>
</tr>
<tr>
<td>3 forward</td>
<td>Message</td>
<td>null</td>
<td>forward message to neighbors</td>
</tr>
<tr>
<td>4 getPos</td>
<td>null</td>
<td>Node</td>
<td>get the current position of agent</td>
</tr>
<tr>
<td>5 setState</td>
<td>char</td>
<td>null</td>
<td>set the state of agent (input is a state)</td>
</tr>
<tr>
<td>6 indexOf</td>
<td>Vector&lt;Node&gt;, int</td>
<td>int</td>
<td>find a node/ a series of nodes from a specified position in a route array</td>
</tr>
</tbody>
</table>

Return the position of the first node.

Table 4-5: Agent methods

The algorithms used for key methods, move, listen and forward, are explained in Algorithm 4-1, 4-2, 4-3 respectively:

```java
boolean move ()

    if NOT arrive at the destination, then
        if the movement is available, then
            Set state into “moving”.
            if exist next position, then
                Shift current position to the next node in the route.
            otherwise, find a position to go according to mobility model.
        otherwise, set state into “waiting” and return TRUE.
    otherwise, set state into “ending” and return FALSE.
```

Algorithm 4-1: Move method
void \textit{listen} (message)

\begin{itemize}
\item if the message is new and relevant, \textit{then} backup in temporary memory.
\item else if the message is listened before, \textit{then return}.
\item otherwise, forward the message and \textit{return}.
\end{itemize}

\textit{Algorithm 4-2: Listen method}

void \textit{forward} (agent ID, message, communication range)

\begin{itemize}
\item if the message has reaches the maximum communication range, \textit{then return}.
\item otherwise, add self Agent ID at the end of Agent List, and transfer the message.
\end{itemize}

\textit{Algorithm 4-3: Forward method}

4.4.1.1 Mobility Model

With respect to route property in Agent class, it records the positions where agent was and will be in street network. According to the level of knowledge, agents may have an idea about the route to travel along (higher level of knowledge) or the destination only (lower level of knowledge). For the first group, the route can be decided before travelling or one step in advance during travel. A mobility model is involved for this purpose. There are three mobility models considered in this thesis:

- Random mobility model: Camp et al. (2002) present a random direction mobility model for entities. Under this model agents are assumed to be willing to turn into any direction except turning backward at intersections. The direction is decided arbitrarily by a random variable. This model can be used for agents who have no specific destination.
• Geodesic mobility model: to employ this model, agents need a clear destination. This model will indicate a physical shortest route between the origin and the destination, frequently a geodesic route. There may be more than one geodesic route between two nodes. Without particular construction about the street network (e.g. highways allow higher speed than roads in downtown area), all these geodesic routes work the same in terms of travel time. Therefore this model chooses the one following the x direction first and then the y direction.

• Traffic estimation mobility model: in a more realistic case, the traffic flows are different distributed in the street network. The popular experience is that the commute can be quite slow in the peak time. To avoid traffic jams, agents are allowed to estimate the traffic status according to their knowledge of street network (Leigh 2006) and traffic flow (Gaisbauer 2006); e.g. the main street distribution in city area. In this model, agents are expected to benefit from less waiting time between transfers.

The specific agent classes (i.e. Client and Host) are derived from Agent, and have additional properties and characteristic methods. Their states, travel routes and current positions can change over time, but type and speed are constant within individual simulation runs. The following sections will introduce the design of specific agent classes.

4.4.2 Client Agent

This section describes the properties and methods in Client class to implement the client agent in this simulation model for a peer-to-peer shared ride system.

As discussed in the previous chapter, a client has different mobility in the simulation: immobile client taking rides only, and mobile client that is also able to walk. Immobile client needs to be picked up from its position. It cannot walk and must wait until a
promised ride collects it. Mobile client is able to walk and can move to a mass transit station or other location where it can get a ride in short time.

A client without a preferred route initializes its route property with origin and destination in turn. In other cases a client is able to employ the geodesic moving model to plan the trip in advance for the reason of minimal travel distance. Alternatively, a client willing to take a detour can look after rides one step in advance. The random moving model is not suitable for the client, because it has a clear destination individually. Also, the traffic estimation moving model is not used for a pre-routing client because the client will not insist on the estimated route when it is given actual offers in difference. However, traffic estimation knowledge can be used for mobile client when it has no ride to take and need to decide the next step on its feet to catch rides with the most possibilities and least waiting time.

In short, to generate a client, two features need to be decided from the below options in advance:

<table>
<thead>
<tr>
<th>mobility options:</th>
<th>routing options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) immobile</td>
<td>1) plan during trip</td>
</tr>
<tr>
<td>2) mobile</td>
<td>2) plan in advance</td>
</tr>
</tbody>
</table>

To arrange the next negotiation, a client needs to remember the detail of undertaking shared rides, e.g. starting and ending time and position. To compare received offers and make a decision, a client needs trip planning ability under its preference, e.g. shortest travel time, least travel fare, or both of them. Therefore, the particular properties and methods of Client class are specified as follows:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>search</code> LPASearch (see definition in Section 4.5.1)</td>
<td>instance of shortest path search class</td>
</tr>
<tr>
<td>2</td>
<td><code>sharedTime</code> int</td>
<td>duration of the most recent shared ride</td>
</tr>
<tr>
<td>3</td>
<td><code>mobile</code> short</td>
<td>a client’s mobile flag: cannot walk and must follow geodesic route 0; can walk and take any route 1; cannot walk but can take any route 2</td>
</tr>
<tr>
<td>4</td>
<td><code>sharedRides</code> Vector &lt;Message&gt;</td>
<td>detail of all undertaken shared rides</td>
</tr>
<tr>
<td>5</td>
<td><code>planMode</code> short</td>
<td>mode of trip planning: quickest trip 0; quickest trip with least waiting time 1; cheapest trip 2; multi-criteria optimized trip 3</td>
</tr>
<tr>
<td>6</td>
<td><code>fare</code> float</td>
<td>the total cost of shared rides</td>
</tr>
</tbody>
</table>

Table 4-6: A client properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>null</td>
<td>Message</td>
<td>broadcast request message of remaining trip</td>
</tr>
<tr>
<td>booking</td>
<td>null</td>
<td>Message</td>
<td>book a host providing the offer contributing the first part of the trip planning result</td>
</tr>
<tr>
<td>takeRide</td>
<td>int</td>
<td>boolean</td>
<td>take a ride from potential offers at a specific time (input) and add it into sharedRides</td>
</tr>
<tr>
<td>addNode</td>
<td>int, Node</td>
<td>null</td>
<td>add a new node after a specified position in a client’s route heading to an intermediate (input index of the specified position in a client’s route array and the intermediate node)</td>
</tr>
<tr>
<td>getNext</td>
<td>null</td>
<td>Node</td>
<td>get the position of the next node to go (if at some position between two nodes)</td>
</tr>
<tr>
<td>move</td>
<td>null</td>
<td>null</td>
<td>decide whether and where to move</td>
</tr>
<tr>
<td>cleanOffers</td>
<td>null</td>
<td>null</td>
<td>remove redundant offers in a client’s messages</td>
</tr>
<tr>
<td>initialSearch</td>
<td>short, Node, Node, int</td>
<td>null</td>
<td>initialize the LPA* algorithm variable (search)</td>
</tr>
<tr>
<td>getShortest</td>
<td>Vector &lt;Message&gt;, simulation, int</td>
<td>null</td>
<td>calculate the shortest path using LPA* algorithm according to offers (input)</td>
</tr>
<tr>
<td>getKpaths</td>
<td>short, Node, Node, Vector &lt;Message&gt;, simulation, Vector &lt;Message&gt;, int</td>
<td>Vector &lt;Message&gt;</td>
<td>calculate K (input) shortest paths based on the shortest path (input) and offers (input).</td>
</tr>
<tr>
<td>convert</td>
<td>short, Node, Node, Vector &lt;Message&gt;, simulation, int</td>
<td>Vector &lt;Message&gt;</td>
<td>convert the shortest path result from LPA* algorithm into a Message list</td>
</tr>
<tr>
<td>tripOutput</td>
<td>null</td>
<td>null</td>
<td>output information of shared rides along a client’s trip</td>
</tr>
</tbody>
</table>

Table 4-7: A client methods
How to design the key methods, *request* and *booking*, is discussed in depth in the following sections.

### 4.4.2.1 Request method

Each negotiation process starts at a request from a client. It is assumed that each a client send out one request message at each time stamp before reaching their destination. To obtain efficient responses, a client needs to inform who is it, where to go, the expected time to start, and a preferred route if any. The position to pick up is also necessary, as it is not always the current position of the client because hosts are only allowed to stop at nodes while a client can be any location between two nodes if it is a mobile client. For this reason, a client only requests the position to pick up at nodes. For mobile client, requests continue to be sent out even when it is walking between two nodes. In this case, the pick-up could be any of the two nodes depending on which can give the client more chance of catching a ride. By default, hosts cannot offer earlier than the client’s expected starting time. So in terms of quickest trip, the earlier requests have more chances to find a ride. For a client, the position of pick-up will be the node closer to itself or the one closer to the next node of its itinerary if it is in the middle. The expected time to start the next ride is calculated using the equation below:

$$ t = t_i + \frac{d}{s_{\text{client}}} \quad (4-1) $$

where \( t \) is the expected time to start specified in request, \( t_i \) is the current time, \( d \) is the distance between a client’s current position and pick-up node in terms of number of edges, and \( s_{\text{client}} \) is the walking speed of the client.

Algorithm 4-4 describes the request method:
4.4.2.2 Booking Method

Given many offers, a client would like to choose the one matching their preference best. Therefore, a trip planning algorithm is designed for a client to make decision. This algorithm is specific under different preference. The typical preferences are listed below:

- Quickest trip: a client who prefers shortest travel time can use this algorithm. In this circumstance, an immobile client always takes the earliest offer, while a mobile client uses a modified Lifelong Planning A* (LPA*) algorithm depicted in Section 4.5.1 to search the shortest path in terms of travel time. In LPA search, the weight of edge in network is initialized by the time of a client walking over. The weight of edges where offers overlap is updated by the shortest time of hosts driving over. The heuristic variable in LPA search is the time that it takes the client to walk along the geodesic route from drop-off to destination. The host contributing the result of the LPA search is then booked.

- Cheapest trip: a client who is concerned with travel fare can employ this
algorithm. To use this algorithm, a client needs knowledge about the fee structure of various hosts. Travel fare of each offer can be calculated according to the type of host and its fee structure. The LPA* algorithm can be also used to find the cheapest trip in the circumstance of one-to-one shortest path problem. However it is quite unpredictable which host the client will travel with next. To define the heuristic function directed to the specific destination in the LPA* algorithm, it is assumed that the remaining part from drop-off to destination is calculated by taking a private car. The offer provides that the minimal travel fare is booked.

- Multi-criteria: in some cases, a client will consider the offers under more than one criterion, for example a relatively quick and cheap trip. It is assumed that these criteria are at a different level of importance to a client in this thesis. Instead of only one option, $K$ shortest paths (Yen 1971) under the most important criterion are calculated with values under other criteria. The optimal path arises from the top of descending sorted paths in order of criteria importance.

The modified LPA* algorithm and $K$ shortest paths algorithms are elaborated in Section 4.5. In booking method, a client can calculate the shortest path or a solution from $K$ shortest paths according to the trip planning mode ($planMode$), and then book the host who contributes the first segment of the result. The booking method is explained in algorithm 4-7:
4.4.3 Host Agent

Host agents are the transportation provider in peer-to-peer shared ride systems. They have limited passenger capacity. Generally, hosts are able to control their trips. They can plan their trip in advance or modify during the travel. It is possible for them to leave their predefined travel route and make a detour for a client. To have an equal chance of being booked, hosts are generated at random position in the grid network. Hosts are interested in request and booking messages only, and forward other messages. The properties and methods for all type of host are shown in Table 4-12 and Table 4-13 respectively.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capacity</td>
<td>int</td>
<td>number of seats</td>
</tr>
<tr>
<td>detourFlag</td>
<td>boolean</td>
<td>willing to make a detour for the client (TRUE) or not (FALSE)</td>
</tr>
</tbody>
</table>

Table 4-8: Host properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkCapacity</td>
<td>null</td>
<td>boolean</td>
<td>check availability of seats: available TRUE; unavailable FALSE</td>
</tr>
<tr>
<td>offer</td>
<td>short</td>
<td>Message</td>
<td>offer a trip in three modes: contributing edges only 0; all remaining trip 1; remaining trip through the start point on request 2</td>
</tr>
<tr>
<td>match</td>
<td>Vector &lt;Node&gt;</td>
<td>float</td>
<td>match requested route with own travel plan, if matched, return start time of this matching travel route; otherwise, return -1.</td>
</tr>
<tr>
<td>setRate</td>
<td>null</td>
<td>null</td>
<td>set variables in fee structure (abstract class)</td>
</tr>
<tr>
<td>CalFare</td>
<td>Vector &lt;Node&gt;</td>
<td>float</td>
<td>calculate the travel fare over the specific route (input node array, abstract class)</td>
</tr>
</tbody>
</table>

Table 4-9: Host methods

Under the offer method, hosts need to decide how to contribute to requested routes:

- Certain offer: match their own travel plans with the requested route, and offer the shared sections. This method works only if the requested route is elaborated with every node on route.

- Related offer: when the requested route has no detailed route, hosts have problem to give certain offers. Alternatively they can offer a possible related
option if they will pass the proposed pick-up in future. This related offer is the remaining part of their travel plans starting from the meet point if any.

- Uncertain offer: a client may have ability to plan their route. To provide as many options as possible for a client’ consideration, hosts can offer their travel route ahead no matter how relevant to the request. This method can be used to avoid arbitrary offers without detailed request itineraries.

Hosts also need to inform when they can come over and pick up the client. This proposed pick-up time is calculated by the distance to the appointment and their speed. The offer method is shown below:

```
Message offer (offer type)

if have spare seats, then
    if able to offer, then
        create an offer message with the proposed pick-up time.
        forward the message.
    otherwise, return null.
else, return null.
```

*Algorithm 4-6: Offer method*

There are several types of hosts proposed in this simulation. They are private car, taxicabs and mass transit host agents. Another special host agent is also designed for the improved use of mass transit. These host agents are modeled in Car, Taxi, Bus and BusStop classes separately. These subclasses override some methods in host class and are introduced in turn.

4.4.3.1 Car Class- Private Car Host Agent
Car class represents the features and behaviors of the private car host agents. Hereafter the private car host agent is called car for short in this section. It is assumed that each car has a route of same length in terms of the number of edges, additionally, cars are eager to achieve their own travel goal and unwilling to make detour. Their route is generated based on the random moving model in advance. Once a car finishes its trip, it will “disappear” (does not move, listen or forward messages) from the transport network. The travel fare with cars is charged by distance of shared ride:

\[ f_{\text{car}} = r_{\text{car}} \times d \]  

(4-2)

where \( f_{\text{car}} \) is the travel fare with car, \( r_{\text{car}} \) is the charge rate per edge by cars and \( d \) is distance of shared ride in terms of the number of edges.

The additional properties and methods in car class are listed below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>float</td>
<td>travel fare rate per grid unit</td>
</tr>
</tbody>
</table>

*Table 4-10: Car properties*

<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalFare</td>
<td>Vector &lt;Node&gt;</td>
<td>float</td>
<td>calculate travel fare according to equation (4-2)</td>
</tr>
</tbody>
</table>

*Table 4-11: Car methods*

4.4.3.2 Taxi Class- Taxicab Host Agent

*Taxi* class is designed for taxicab host agent, hereafter, taxi for short. Taxis have no clear destination at the beginning, and employ random moving model to decide where to go one step ahead. Taxis have no fixed length of route, so they stay in the simulation since they are generated. It is assumed that taxis are always willing to make detour for a client,
and provide rides as far as possible, for example from the proposed pick-up to destination directly. It is also assumed that once confirmed by a booking message, taxis are able to head to the appointed position as soon as possible (following the geodesic route) from the next time stamp. The assumption is reasonable because the ability of finding the shortest/quickest route from their current position to the appointment could come from the driver’s practical experience or benefit from a navigation device. Under this assumption, taxis always offer if they get requests.

The travel fare of taking a taxi is calculated by a combination of distance and a flag fall:

\[
f_{\text{taxi}} = F + r_{\text{taxi}} \times d
\]

(4-3)

where \(f_{\text{taxi}}\) is the travel fare with a taxi, \(F\) is the flag fall, \(r_{\text{taxi}}\) is the charge rate per edge by taxis, and \(d\) is the distance in terms of the number of edges.

The design of taxi class is depicted in Table 4-16 and Table 4-17:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>flagFall</td>
<td>the cost of flag fall</td>
</tr>
<tr>
<td>2</td>
<td>rate</td>
<td>travel fare rate per grid unit</td>
</tr>
</tbody>
</table>

*Table 4-12: Taxi properties*
<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>match</td>
<td>Vector &lt;Node&gt;</td>
<td>float</td>
<td>return the time that arrives the start position of request message</td>
</tr>
<tr>
<td>offer</td>
<td>null</td>
<td>Message</td>
<td>always offer the whole requested trip, and add the trip into own travel plan</td>
</tr>
<tr>
<td>getWay</td>
<td>Node, Node Vector &lt;Node&gt;</td>
<td></td>
<td>find geodesic route between two nodes</td>
</tr>
<tr>
<td>calFare</td>
<td>Vector &lt;Node&gt;</td>
<td>float</td>
<td>calculate travel fare according to equation (4-3)</td>
</tr>
</tbody>
</table>

Table 4-13: Taxi methods

4.4.3.3 Bus Class - Mass Transit Host Agent

The class describing the natures and behaviors of mass transit host agent is named *Bus*. The word “bus” is used to indicate all mass transit host agents in this section. For this kind of agent, the significant difference is that every bus runs on a fixed route and follows a predefined timetable. Frequently, several buses serve on one specific route with different timetable back and forth. Like cars, buses cannot leave their route and make detour for a client. Like taxis, buses will not disappear from the time of creation. However, buses do not move from the time when they are created but wait until the earliest turn in their own timetable. Each bus has a timetable on board. The travel fare of taking a bus is one-off - that means a client are charged a fixed fee for using one bus regardless the distance.

A class describing the mass transit line, called *BusLine*, is defined by an identifier, number of stops and the number of nodes between two stops. In this simulation, there are two mass transit lines defined, with arbitrary locations: both cross the central area of the grid network, but approach the fringe in different directions. The structure of mass transit line is presented in Table 4-18, and the arbitrary mass transit lines are shown is
Figure 4-4. The same mass transit lines are used in experiments in Chapter 5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. id</td>
<td>char</td>
<td>the identifier of mass transit line</td>
</tr>
<tr>
<td>2. route</td>
<td>Vector &lt;Node&gt;</td>
<td>the route of mass transit line (includes every node along the route)</td>
</tr>
<tr>
<td>3. num</td>
<td>int</td>
<td>the number of stops</td>
</tr>
<tr>
<td>4. interval</td>
<td>int</td>
<td>the number of grid units between every two stops</td>
</tr>
</tbody>
</table>

Table 4-14: Structure of BusLine

When buses respond to requests, they are also restricted by allowing a client to get on and off at stops only, even if they pass by a client in between. The overridden *match* method is shown below:
float \textit{match} (requested route)

\begin{algorithm}
\textbf{for} each node in own route \textbf{do} \\
\hspace{1em} if find the first node of requested route in the remaining route, \textbf{then} \\
\hspace{2em} if the node is at a stop, \textbf{then} \\
\hspace{3em} set the node as the pick-up. \\
\hspace{2em} for each node in the left requested route \textbf{do} \\
\hspace{3em} if the node in the remaining route, \textbf{then} \\
\hspace{4em} if the node is at a stop, then \\
\hspace{5em} set the node as the drop-off. \\
\hspace{4em} \text{otherwise}, continue the left requested route from the next node. \\
\hspace{2em} \text{otherwise}, \text{STOP}. \\
\hspace{3em} \text{otherwise}, continue the remaining route from the next node. \\
\hspace{1em} \text{otherwise}, continue the remaining route from the next node. \\
if the pick-up and the drop-off exist, \textbf{then return} the time of pick-up. \\
\textbf{otherwise, return} -1.
\end{algorithm}

\textit{Algorithm 4-7: Match method}

The detail of bus class is listed in Table 4-15 and Table 4-16.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 \textit{timeTable}</td>
<td>int[]</td>
<td>timetable of the bus (size of the array is the number of stops)</td>
</tr>
<tr>
<td>2 \textit{busLine}</td>
<td>BusLine</td>
<td>reference to the employed mass transit line</td>
</tr>
<tr>
<td>3 \textit{cost}</td>
<td>float</td>
<td>the travel fare of taking the bus regardless the distance</td>
</tr>
</tbody>
</table>

\textit{Table 4-15: Bus properties}
<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>move</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>offer</td>
<td>short</td>
<td>Vector &lt;Node&gt;</td>
</tr>
<tr>
<td>3</td>
<td>atStop</td>
<td>int</td>
<td>boolean</td>
</tr>
<tr>
<td>4</td>
<td>calFare</td>
<td>null</td>
<td>float</td>
</tr>
</tbody>
</table>

Table 4-16: Bus methods

With so many restrictions, a low occupation rate of buses is expected. To moderate their disadvantages and increase their competence for shared ride, a higher speed for buses is proposed, e.g. double of the other hosts’ speed. To catch such a fast host, Mobile client need a relative large communication range to perceive the oncoming buses. If Mobile client intend to catch buses at a bus stop one block away, a minimal range can be calculated with the following equation:

\[
R = \frac{s_{bus}}{s_{client}} + 1
\]

(4-4)

where \( R \) is the minimal communication range, \( s_{bus} \) is the speed of bus, and \( s_{client} \) is the speed of the Mobile client.

4.4.3.4 BusStop Class– A Static Host Agent

As discussed in the previous section, a client will probably miss oncoming buses if its communication range is not large enough, because it does not have sufficient time to walk to the nearest bus stop. Imagine that buses are two times faster than other hosts, and the speed of a client is one fifth of that of normal hosts. Then the required
communication range is 11, additionally, successful communication needs other agents in between to keep forwarding messages, which is a rigorous requirement for a client. To solve this problem, a kind of static host agent is designed. These static agents are named *BusStop*, because they are located at the position of stops along the mass transit line. It is assumed that they have unlimited battery performance for unconstrained wireless communication, so that they are able to perceive the arrival time of each mass transit vehicle along the mass transit line. Either buses or a client can communicate with bus stops. The negotiation between a client and buses now splits into two parts: negotiation between a client and bus stops, and negotiation between bus stops and buses. The main functions of bus stops are listed as below:

- Accept request from a client and respond to them with information (e.g. arrival time and route) of oncoming buses.

- Keep a timetable of all buses on board for a client’s consideration, including those out of a client’ communication range.

- Accept booking from a client, and inform the indicated bus.

It is expected that with help of bus stops, the occupation rate of buses will be improved. Additionally, due to their distribution in the network and unconstrained communication range, bus stops have potential ability as data collectors. The properties and methods of *BusStop* are specified in Table 4-21 and Table 4-22 respectively.
### Table 4-17: BusStop properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 lineNo</td>
<td>char</td>
<td>the identifier of employed mass transit line</td>
</tr>
<tr>
<td>2 infor</td>
<td>int[][3]</td>
<td>detail of all buses coming to the bus stop: [i][0] bus identifier; [i][1] arriving time; [i][2] number of spare seats</td>
</tr>
</tbody>
</table>

### Table 4-18: BusStop methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 setInfor</td>
<td>null</td>
<td>null</td>
<td>update information of coming buses</td>
</tr>
<tr>
<td>2 getBuses</td>
<td>int</td>
<td>Vector &lt;int&gt;</td>
<td>find all buses that come after a specific time (input), and return list of their IDs</td>
</tr>
</tbody>
</table>

## 4.5 Trip Planning Algorithms

This section describes algorithms applied for trip planning of a client in the simulation model. Other agents can also employ these algorithms to plan their trips. These algorithms, based on the trip planning algorithms discussed in Section 2.3, are modified to suit particular requirements of peer-to-peer shared ride systems.

### 4.5.1 LPA* Algorithm

LPA* algorithm is used to find the shortest path under a given cost function between two specific nodes in networks, where the cost over edges (weight) is variable. In peer-to-peer shared ride systems, the cost can be the travel time or travel fare needed to pass an edge. The network comprising transportation hosts is dynamic: the position of hosts and their routes change frequently. This means that the previous transportation
information used to calculate the shortest path is likely obsolete and the shortest path need to recalculate based on the current information. Thereby there is no memory of previous trip planning result stored in agents.

In the modified LPA* algorithm, a default weight is applied to each edge by the cost in the worse case (e.g. the maximal travel time or most expensive travel fare) or an extreme large value. Each node has an initial value (g-value) that is the cost from the origin to the node. If an offer can cut the cost, the value of the edges, where the offer overlaps, is updated. And the identifier of the host providing the offer is recorded on the edges. The heuristic variable (h-value) of nodes is an estimated cost needed to fulfil the trip to the destination: the default weight is counted where no offer covers. The shortest path extends along the nodes those have the minimal sum of g-value and h-value.

As each segment of offers is only available at a specific time, the means of using offers to update weights is the key point in the case of peer-to-peer share ride systems. If a segment cannot be reached at the specific time, the segment should be ignored and not influence the weight. An illumination (Fig. 4-5) below shows the case: at time 8, a mobile client (the red solid circle) with speed of 1/4 edge per time is planning trip; an offer provides a ride (the blue thick line) from Node A to Node D processing from time 10, and the travel time of each edge is 1 by the offer. The default weight in this case is 4, the time that the client needs to walk through an edge without rides. The first value in bracket is the g-value of the node above the bracket, and the second value is the h-value. As seen in Figure 4-5, the client cannot reach Node A and B at the time offered, therefore only the value of segments between Node C and D are updated.
Figure 4-5: Using an offer to update weights in LPA* algorithm

An offer starts at time 10.

A mobile client plans trip at time 8.

The client can reach Node C.

The first two segments are ignored.

Figure 4-5: Using an offer to update weights in LPA* algorithm
The LPA* algorithm used in this thesis is depicted below. A class named `LPASearch` (Table 4-19) defines the methods (Table 4-20) of computing the shortest path, and two classes named `LPA_Node` (Table 4-21) and `LPA_Edge` (Table 4-22) represent the nodes and edge separately in the weighted network. Agents can generate an instance of `LPASearch` class to calculate the shortest path, for example in the `booking` method of `Client` (see detail in Section 4.4.2.2).

```
initialize a network with the start node and the end node, the default value of weight, g-value and h-value of current node and its neighbors.

add the start node into a node list `path`.

for each offer do
    update the weight of edges, and g-value and h-value of nodes where the offer overlaps and the client can reach, the weight determined by smallest cost going through the edges.

for each node from the start node do
    if it neighbors have no g-value or h-value, then compute these values.
    find the node in neighbors with the minimal sum of g-value and h-value.

    if the node is not the end node, then add it in `path`.
    otherwise, STOP.

return `path`.
```

*Algorithm 4-8: Modified shortest path algorithm*
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cID</td>
<td>int</td>
<td>searching agent id (a client id)</td>
</tr>
<tr>
<td>start</td>
<td>int</td>
<td>start node</td>
</tr>
<tr>
<td>goal</td>
<td>int</td>
<td>goal node; i.e. destination</td>
</tr>
<tr>
<td>maxX</td>
<td>float</td>
<td>width boundary of grid network in terms of number of columns</td>
</tr>
<tr>
<td>maxY</td>
<td>float</td>
<td>length boundary of grid network in terms of number of rows</td>
</tr>
<tr>
<td>worstWeight</td>
<td>float</td>
<td>initial value of weight of edge</td>
</tr>
<tr>
<td>nodeList</td>
<td>Vector &lt;LPANode&gt;</td>
<td>list of visited nodes</td>
</tr>
<tr>
<td>edgeList</td>
<td>Vector &lt;LPAEdge&gt;</td>
<td>list of visited edges</td>
</tr>
<tr>
<td>path</td>
<td>Vector &lt;int&gt;</td>
<td>the shortest path</td>
</tr>
</tbody>
</table>

*Table 4-19: LPASearch properties*
<table>
<thead>
<tr>
<th>Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>initialize</em></td>
<td>null</td>
<td>initialize the start and end nodes, default weight value, searching agent id, boundary of x and y, starting time</td>
</tr>
<tr>
<td>2</td>
<td><em>getValue</em></td>
<td>float</td>
<td>get the g value of node</td>
</tr>
<tr>
<td>3</td>
<td><em>hValue</em></td>
<td>float</td>
<td>get the h value of node</td>
</tr>
<tr>
<td>4</td>
<td><em>rValue</em></td>
<td>float</td>
<td>get the r value of node</td>
</tr>
<tr>
<td>5</td>
<td><em>setGValue</em></td>
<td>null</td>
<td>calculate g value of node</td>
</tr>
<tr>
<td>6</td>
<td><em>setHValue</em></td>
<td>null</td>
<td>calculate h value of node</td>
</tr>
<tr>
<td>7</td>
<td><em>setRValue</em></td>
<td>null</td>
<td>calculate r value of node</td>
</tr>
<tr>
<td>8</td>
<td><em>getNext</em></td>
<td>LPANode</td>
<td>get next node id in shortest path</td>
</tr>
<tr>
<td>9</td>
<td><em>update</em></td>
<td>null</td>
<td>update weight of edges where an offer covers a host at a specific time</td>
</tr>
<tr>
<td>10</td>
<td><em>compute</em></td>
<td>null</td>
<td>compute a shortest path between two specific nodes</td>
</tr>
</tbody>
</table>

**Table 4-20: LPASearch methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>ID</em></td>
<td>int</td>
</tr>
<tr>
<td>2</td>
<td><em>gValue</em></td>
<td>float</td>
</tr>
<tr>
<td>3</td>
<td><em>hValue</em></td>
<td>float</td>
</tr>
<tr>
<td>4</td>
<td><em>rhsValue</em></td>
<td>float</td>
</tr>
</tbody>
</table>

**Table 4-21: LPANode (inherited from Node class) properties**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hID</td>
<td>int</td>
<td>identifier of edge</td>
</tr>
<tr>
<td>pID</td>
<td>int</td>
<td>id of the start node of the edge</td>
</tr>
<tr>
<td>sID</td>
<td>int</td>
<td>id of the end node of the edge</td>
</tr>
<tr>
<td>weight</td>
<td>float</td>
<td>travel cost from the start node to the end node</td>
</tr>
</tbody>
</table>

*Table 4-22: LPAEdge properties*

### 4.5.2 K Shortest Paths Algorithm

The LPA* algorithm is adaptive to any single criterion, such as travel time or fares. However, only one optimal path is returned by this algorithm each time, and other possible trips, for example when ranking with multiple criteria, need to be calculated under these criteria separately. For this purpose, the algorithm in this section is designed to find a set of options, and $K$ is the size of the set. This algorithm is based on Yen’s algorithm (1971), which calculates the $k^{th}$ ($k = 2, 3, \ldots, K$) path by arbitrarily not passing (i.e. removing) the subpaths between the first nodes of the $(k-1)^{th}$ path. In the circumstance of peer-to-peer shared ride systems, the weighted network is not street networks but the dynamic transportation networks composed by transportation hosts. In such dynamic networks, it is likely that more than one host provide diverse routes between two nodes, and such routes provide options. Therefore the subpaths to be removed are not street segments between nodes, but offers reaching the two ends of the segments.

There are two lists of paths in such algorithm: List A is the list of $k$-shortest paths; List B is the list of candidates for $(k+1)^{th}$ shortest paths. It is clear that the only necessary condition to calculate the $k^{th}$ shortest path is the first ($k=1$) shortest path, which can be calculated by the LPA* algorithm with all offers. For each $(k+1)^{th}$ path, remove the
coincide offer contributing the first subpath of the $k^{\text{th}}$ path and use LPA* algorithm to recalculate according to the remaining offers. When the path(s) in List A plus those in List B exceed $K$, the set is full and computing is done. In dynamic transportation networks, less computing is desired to respond quickly and save the energy of devices. In this thesis, the $K$ shortest paths are calculated by LPA* algorithm dealing with local knowledge, such approach is efficient and is not always global optimal. The $K$ shortest paths algorithm is explained as follows, where $A^k$ is the $k^{\text{th}}$ shortest path:

<table>
<thead>
<tr>
<th>Algorithm 4-9: modified $K$ shortest paths algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>define the value of $K$.</td>
</tr>
<tr>
<td>calculate the shortest path $A^1$, and move it into List A.</td>
</tr>
<tr>
<td>for the first $i$ ($i = 1, 2, \ldots, k-1$) offers composing $A^{k-1}$ ($k = 2, 3, \ldots, K$) do</td>
</tr>
<tr>
<td>if the $i^{\text{th}}$ offer coincide with the $i^{\text{th}}$ offer of $A^j$ ($j = 1, 2, \ldots, k-1$), then</td>
</tr>
<tr>
<td>remove the offer.</td>
</tr>
<tr>
<td>re-calculate the shortest path $A^k$ from the start node to the destination, and move it into List B.</td>
</tr>
<tr>
<td>if no offer left, then STOP.</td>
</tr>
<tr>
<td>if paths in List B plus List A exceed $K$, then return all paths in List A and B.</td>
</tr>
<tr>
<td>otherwise,</td>
</tr>
<tr>
<td>move arbitrary one from List B to List A.</td>
</tr>
<tr>
<td>revert all offers.</td>
</tr>
<tr>
<td>$k$ equals $k+1$ and repeat the above cycle process.</td>
</tr>
</tbody>
</table>
4.5.3 Multi-criteria Optimization Algorithm

When considering multiple trip-planning criteria (e.g. both travel time and travel fare), a client can use $K$ shortest paths algorithm to obtain more options, which in turn are balanced under all criteria, and then choose one solution to book. The procedure of choosing one solution to meet multiple criteria is called multi-criteria optimization. This section develops a multi-criteria optimization algorithm based on multilevel programming (Bard and Falk 1982), which orders criteria in terms of importance and select options by finding the minimal value under each criterion in turn. Multilevel programming optimization is suitable to the case with any number of criteria, and reasonable as people consider criteria in different level of importance when making decisions. At the stage, only two criteria are considered in such algorithm: travel time and travel fare. Additionally, shorter travel time is assumed more important for a client than lower travel fare. The value of $K$ is defined as the tolerance of a client to suboptimal paths. At first, $K$ shortest paths are computed under the first criterion, travel time, each attaching a value of travel fare. Then, the path with the lowest travel fare in such $K$ shortest paths is chosen as the solution. If more than one path have the minimal travel fare, arbitrary one is taken as the solution. If other criteria exist in case, the paths with minimal travel fare are sorted again under the third important criterion until only one solution rises or all criteria are optimized. The process of multi-criteria optimization is described below:
define the level of importance of multiple criteria.
calculate \( K \) shortest paths under the first important criterion and move them into a list \( \text{candidates} \).

\[
\begin{align*}
\text{for each other criterion in order of importance do} & \\
& \text{compute the value of each path in } \text{candidates} \text{ under the criterion.} \\
& \text{sort these paths by the value.} \\
& \text{find the path(s) having the minimal value, and remove other paths from } \text{candidates}. \\
& \text{if only one path rises, then STOP.} \\
\text{return the arbitrary one path in } \text{candidates}. 
\end{align*}
\]

Algorithm 4-9: modified multi-criteria optimization algorithm

4.6 Simulation Assessment

Having discussed the negotiation process and agent design, the simulation model of a peer-to-peer shared ride simulation is mostly established. This section investigates what and how to collect data in simulation, and the criteria to assess the performance of the simulation model.

This thesis concerns on how introduction of types of agents will affect the shared rides. To analyse the detail of undertaken shared rides, the following information is needed:

- **When**: the time stamp and runtime that a shared ride happened.
- **Who**: which a client travelled with which host.
· **How**: the route that the shared ride ran along.

More interests about the detail of negotiation process can be satisfied if other messages are recorded, e.g. requests and offers. For this purpose, a strategy is proposed to store such information: attach one file to each agent as an observer. When an agent receives a message, the detail is store in the agent’s attaching file in the following format:

<table>
<thead>
<tr>
<th>runtime stamp</th>
<th>a client ID</th>
<th>coordinates of the start node</th>
<th>coordinates of the end node</th>
<th>type of message</th>
<th>host ID</th>
</tr>
</thead>
</table>

Upon review of the file records, it is possible to trace the negotiation process and the shared rides. The performance of the simulation model is assessed by comparing the shared ride trips with the optimal trips from a global view under the same trip planning criterion/criteria. The global optimal trip is computed by all transportation information including those out of the communication range of a client or newly entering the simulation world during sharing rides period. The trip quality is investigated by Guan (2007) and the result is coming shortly. To compare with preliminary research (Winter and Nittel 2006) on homogeneous hosts, average travel time and average number of messages are calculated for each simulation run for specific communication and way-finding strategies. The comparison can show whether and how much the simulation model involving types of agents improves the shared rides.

The assessment process can provide evidence to improve the model. Also, with the help of a database visualization tool, e.g. *Secondo* (Gueting 2006), these files can be used as the description of moving objects and represent the shared rides visually (Guan 2007).
Chapter 5
Experiments

In this chapter, several experiments are designed to investigate agent behavior in the peer-to-peer shared ride system formalized in Chapter 4. These experiments start with a simple case with homogeneous agents, and then involve types of a client and hosts with various mobility models under different communication and way-finding strategies. The *effectiveness* and *efficiency* (see definitions in Section 1.2) of negotiation are assessed by the average travel time of a client and the numbers of broadcasted messages during the period. Although the simulation model allows setting various parameters of grid environment and agents, these parameters need to be specified to conduct individual experiments. Different settings vary the quantitative value of results; however they do not materially influence the qualitative conclusion.

In experiments, one client is initialized with specific origin and destination, and hosts are distributed randomly with a capacity and fixed speed in the grid network. There are three types of hosts capable of providing transportation: private cars, taxicabs and mass transit, whose parameters are determined as shown in Table 5-1 for all experiments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacity</th>
<th>Speed</th>
<th>Route</th>
<th>Detour</th>
<th>Fare Rate</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>private car</td>
<td>2</td>
<td>1</td>
<td>predefined by 12 edges</td>
<td>FALSE</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>taxicab</td>
<td>1</td>
<td>1</td>
<td>variable</td>
<td>TRUE</td>
<td>1</td>
<td>flag fall is 1</td>
</tr>
<tr>
<td>mass transit</td>
<td>10</td>
<td>2</td>
<td>along a bus line</td>
<td>FALSE</td>
<td>-</td>
<td>one-off charge is 2; time schedule</td>
</tr>
</tbody>
</table>

*Table 5-1 Parameter settings of transportation hosts*

The first part of the chapter presents the results of six experiments: it is assumed that a
client have a preference of quickest trip in the first five experiments; the last experiment examines a more complex trip planning issue, in which a client considers not only travel time but also other criteria (e.g. travel fare) when looking to find an optimal trip. Then, final chapter discussions centre on how negotiation is affected by agent behaviors, followed by results of assessment of various shared ride trips.

5.1 Simulation Results

This section presents the results of experiments, which are calculated from 1000 simulation runs under the same environmental parameters and agent initialization. Experiment purpose and setting details are elaborated in turn.

5.1.1 Boundary Effect with Homogeneous Agents

As the origin and the destination of one client are specific and hosts are distributed randomly in all experiments, a thought of whether this arbitrary positioning of a client will influence the simulation outcome arises. Because the grid network is symmetrical, the positioning is typically variable with relation to the boundary, the so called boundary effect in this section. To examine whether the boundary effect exists in the simulation model, the first experiment conducts a simple case with homogeneous agents. The size of grid network is 10×10 nodes, and the length of each edge is unitary. One immobile client requests a geodesic route of 5 edges, which are typically positioned (Figure 5-1): 1) along the boundary; 2) reaching the boundary; 3) within the boundary. Hosts are homogeneous: all are private cars. The predefined routes of private cars are generated on the random mobility model discussed in Section 4.4.1.1. The density of hosts varies from 24 to 144 in each simulation. The boundary effect is tested under the middle-range communication strategy: \( \text{comRange} = 3 \).
Figure 5-1: Positioning of a client’s route
Figure 5-2 (a): Travel time along 3 routes

![Graph showing travel time along 3 routes.](image)

Figure 5-2 (b): The number of messages along 3 routes

![Graph showing the number of messages along 3 routes.](image)
Figure 5-2 shows that requested route along the boundary (route 1) causes significantly shorter average travel time and less broadcasted messages than the other positioned routes, which means that a client has much greater chances of gaining contributed rides along the boundary, although the difference is moderated as the number of hosts increases. Comparatively, the average travel time and the average number of messages when following the route reaching the boundary (route 2) and the route within boundary (route 3) are not widely different. The reason is that under random mobility model the probability of a client to catch a shared ride is defined as:

$$P = \frac{n_{host}}{n_{node}} \times P_{\text{turning}}$$ (5-1)

where $P$ is the possibility of catching a ride at a node $(0 \leq P \leq 1)$, $n_{host}$ is the number of random moving hosts in a grid network, $n_{node}$ is the number of nodes of the grid network and $P_{\text{turning}}$ is the possibility of turning into the requested direction at the node.

Random moving hosts have equal possibility of turning into any direction at nodes, which means that $P_{\text{turning}}$ is 25% at nodes within boundary, 33% at nodes on boundary and 50% at four corners. Route 1 comprises a corner node and 5 on-boundary nodes, so its $P_{\text{turning}}$ is much higher than other two. Besides particular positioning of the requested route, higher density of hosts can also decrease travel time. However, increment on host number brings only minimal changes to the properties of host density indistinctively: 50 more hosts only increase density by 0.5 in a 100-node grid network. To avoid the boundary effect of the simulation grid network, one client is located in the center area in the following experiments.

### 5.1.2 A Client with Various Motilities and Level of Knowledge

This experiment compares a client with various nobilities and level of knowledge: 1) an immobile client who sticks to a geodesic route, 2) an immobile client who is willing to make detours, 3) a mobile client who sticks to a geodesic route, and 4) a mobile client
who is willing to make detours. Each a client departs at (0, 5) and heads to the destination at (9, 5) in a grid network of 10 × 10 nodes. The geodesic route is defined as a trip comprising nine edges along X direction between the two nodes. A client consistently chooses a cost function of travel time. Mobile client have a walking speed of $v_c = 0.25$ edges per time unit (they need four time units to finish traveling along an edge), compared to the host speed of $v_h = 1$ edge per time unit. In this experiment, hosts are homogenous: all are private cars. The densities of hosts vary from 0.24 to 1.44. Each host travels over twelve time units along a travel route that is generated using the random mobility model. Agents communicate within a range of three. Figure 5-3 (a) shows the average time of shared rides by a client with different mobility and knowledge, and Figure 5-3 (b) shows the corresponding numbers of broadcasted messages.
Figure 5-3(a): Travel time of a client with different mobility and knowledge

Figure 5-3(b): The message numbers of a client with different mobility and knowledge
The above figure shows that the performances of a client with different mobility and knowledge are variable: generally, mobile client achieves relatively shorter trips in terms of travel time, and the difference moderates as the density of host increases. Within communication range, a client might know a ride available in the next parallel street. In this case, mobile client has the advantage to walk over one edge or four time units. The disadvantage also exists: a client has the risk of missing better offers from the hosts who just enter the communication range or were generated, while the client are walking between two nodes (positions for pick-up). Immobile client insisting on a geodesic route travels longer than other client, because the possibility of hosts moving on the geodesic route is rare under the random mobility model; additionally, such a client cannot walk forward, they have to wait for a hosts to appear on the geodesic route, and the waiting time is long especially when the density of host is low. A client, who is free to make detours, benefits by more choices to other directions. But when the client leave the geodesic route, the physical length of the trip becomes longer. The increasing number of communication messages proves that a client needs more negotiation effort to find offers for the longer trip.

5.1.3 Types of Hosts

This experiment is designed to investigate the difference of employing types of hosts separately under mid-range communication strategy. These hosts are private cars, taxicabs and mass transport.

5.1.3.1 Shared rides by private cars

Private cars are specified by a speed of one edge per time unit, a route that is determined by random at their departure and from which they will not deviate, and a low passenger capacity (Table 5-1). The latter is not relevant in a simulation with one a client only. The simulation that tests a peer-to-peer shared ride system consisting of one Immobile client
and this type and specification of hosts has been investigated in previous work already (Winter and Nittel 2006). In that work it was shown that mid-range communication delivered trips nearly as quick as with unconstrained communication, for all densities of hosts.

However, with the introduction of different a client types (Figure 5-3), it turns out that Mobile client, due to their increased choice, have advantages over Immobile client. If travel time is the optimization criterion, a shared ride system for private cars would lead to combined trips of rides and walks.

5.1.3.2 Shared rides by taxicabs

Taxis are specified by a speed of one edge per time unit, a route that is determined randomly at their departure, but from which they are willing to deviate any time (if not occupied), and a low passenger capacity (Table 5-1). With this specification, a peer-to-peer shared ride system for taxis leads to a client trip times close to the theoretical optimum, which is defined by the distance of the client’s departure and destination, and the host speed. A taxi comes as soon as possible and heads directly to the client’s destination without detours. Only the density of taxis determines the (average) waiting time of the client. The higher the density, the shorter the waiting time. Since the simulation result is predictable, we abstain from a diagram.

Communication range has a minor impact in a peer-to-peer shared ride system for taxis. Since the nearest (free) taxi will always be chosen, and no other agents are present in this system, the nearest taxi has either to be in direct communication range, or currently occupied taxis can bridge by message forwarding. Only if taxis are employed in a shared ride system together with other hosts of larger numbers, the communication connectivity established by the other hosts will make a significant difference.
5.1.3.3 Shared rides by mass transit

As discussed in previous sections, mass transit has significant disadvantages, such as a fixed itinerary and time schedule. To make mass transit more attractive to a client, their speed is specified as two edges per time unit (twice the speed of other hosts). The capacity of mass transit is relatively larger: 10. This value is not comparable to the real number of seats in buses or trains, but in having only one a client in each simulation, the exact value will not influence the occupation of a client. Buses are used as the representation of mass transit in this experiment then. Buses run by going to and from predefined and fix routes (Figure 4-4), on scheduled frequencies. In this experiment the frequency of a bus at stops is set to every time unit in two directions. The world is of size 20 × 20 nodes. Two cases are designed for the mode of buses: 1) case 1 is a simulation with a bus line through the parallel street of the client’s geodesic route (bus line 1 in Figure 4-4); 2) case 2 is a simulation with a bus line overlapping with a major part of the geodesic route (bus line 2 in Figure 4-4). A client travels between node (5, 10) and node (15, 10). Both cases are investigated for (a) buses being the only hosts in the simulation, (b) buses being hosts among 480 occupied private cars, so that private cars establish communication connectivity but do not offer rides, and (c) buses and bus stops being the only hosts in the simulation, so that missing communication connectivity to buses is balanced by the presence of bus stops within the direct communication range of the client.

Figure 5-4 demonstrates the results, both in terms of average trip times as well as numbers of broadcasted messages. It turns out that the presence of bus stops is of advantage compared to both other scenarios. The numbers of messages shown by cured lines are larger within limits than that in the case of buses only, but the average travel times represented by yellow bars are significantly shorter. Bus stops also help to reduce the communication effort in the presence of other hosts that are willing to establish connectivity within comRange.
5.1.4 A Mixed Case with All Types of Agents

Having studied types of a client and hosts respectively, this section investigates a mixed case with Mobile client capable of walking and willing to make detours and all types of hosts. Mass transit is called bus in this experiment. This case employs the two bus lines designed in Figure 4-4 for buses, and tests three communication strategies: short range \((comRange = 1)\), middle range \((comRange = 3)\) and unconstrained \((comRange = 40)\) in a world of \(20 \times 20\) nodes. The parameters of agents are set as in Table 5-1. There are five case studies with the same density of transportation hosts but different portions: 1) 144 private cars only; 2) 96 private cars and 48 buses (12 buses run on each direction of the two bus line); 3) 96 private cars, 48 buses and 24 bus stops to help transferring bus travel information; 4) 96 private cars and 48 taxis; and 5) 48 private cars, 48 taxis, 48 buses and 24 bus stops. The average travel time and number of messages are shown in

\[\text{Figure 5-4: Comparison of hosts as mass transit running on two lines}\]
Figure 5-5.

*Figure 5-5(a): Travel time of different host composition and a mobile client*

*Figure 5-5(b): Number of messages of different host composition and a mobile client*
The first case refers to the preliminary investigation (Winter and Nittel 2006) about three communication strategies with homogeneous a client and hosts at a host density of 0.72. The results present a decreasing travel time as multiple types of agents are added, particularly the fifth cases, with the most types of agents achieving the shortest travel time. And the middle range communication is demonstrated as coming up with the shorter trip very close to the unconstrained communication which collects all information of connected agents, but only broadcasts about 50% of the messages.

5.1.5 Agent Mobility Model with Street Centrality

A related research by Leigh (2006) is given on a more sophisticated agent mobility model, in which agents have a preference of traveling on a particular path, i.e. on the centrality of street networks. For example, agents have knowledge of main streets in the downtown area, and prefer to travel on those streets that they are familiar with. The distribution of the main streets in a world of $11 \times 11$ nodes is shown in Figure 5-6. They are named from AA to AI separately.
This experiment focuses on the various behaviors of agents having different knowledge of the street distribution. One client travels between node (0, 5) and node (10, 5): one group has knowledge and is able to walk; the other group has no knowledge and follows the geodesic route in between, which crosses the central area of the streets and partially overlaps with the street AG and AC. 120 hosts are declared by these two groups too: those with knowledge will be more likely to arrange their trips on the main street AA – AI; others without knowledge employ random mobility models: go every direction with equal probability. Figure 5-7 presents a comparison among agents.

Figure 5-6: Test network containing named streets (Leigh 2006)
without knowledge, only hosts having knowledge, and both hosts and a client having knowledge.

In the above figure, the blue bars show the average travel time of each group, and the wine polyline presents the differences of average number of message. It is seen that hosts tending to the main streets bring advantages (shorter travel time and less communication effort) to the travel of a client who follows a geodesic route covering some main streets. The advantages are maximized in the third group, in which both a client and hosts have the same level of knowledge.

5.1.6 Multi-Criteria Trip Planning

The previous five experiments are tested on the assumption that a client care about travel time and want quickest trips. Nevertheless, people may consider more factors when planning their trips in the real world. The main considered factors include the
travel fare, the convenience (in terms of transfers), comfort, and security. More criteria make the planning of trips more complex: to make an appropriate decision, people need to balance among criteria. That means the decision may not be best on individual criterion but good enough as a whole.

For this reason, this experiment is designed to investigate multi-criteria trip planning issues for agents in peer-to-peer shared ride systems. The agents are classified into three groups: 1) prefer the quickest trip; 2) consider both travel time and fare; and 3) care about travel fare only. Because walking is always the cheapest way to travel, the simulation result is predictable in case three with mobile client: the travel time is the distance of the geodesic route multiplies the speed of the client and the travel fare is nil. Therefore to avoid the definite result, it is assumed that a client is not able to walk, but would not mind making detours in this experiment. Choices of hosts are plentiful for them in this experiment, including private cars, taxis, buses and bus stops. Distribution of bus lines is the same in Figure 4-4. The grid network density of transportation hosts is 0.72. The fare rate of diverse hosts is set as shown in Table 5-1. It is assumed that for the client, travel time is more important than travel fare when planning trips under these two criteria. The value of $K$ is three that means for the multi-criteria trip-planning client the solution comes up with the cheapest one among three options having the minimal travel time.
Figure 5-8: Comparison of multi-criteria trip planning with single criterion

The above figure presents the differences between various trip planning strategies. The blue bars show that average travel time of the first group is 4.22% and 5.48% shorter than the second and third groups respectively, but the average travel fare of such group shown by the wine polyline is significant expensive: 13.76% and 14.09% higher than the second and third group. On the contrary, the third group achieves the lowest travel fare and suffers the longest travel time. It is clear that multi-criteria trip planning (shown as the second group) moderates the travel time and fare outcomes: neither quickest nor cheapest, but relative cheaper and quicker compared to the first and third groups respectively.

5.2 Discussion

In experiments, the density of agents is critical in deference to the number of agents. Because in various size of grid network, the same number of agents (e.g. 72) is not
comparable, that means only 25% possibility of sharing rides in a 20 × 20 network than that in a 10 × 10 network. The same density of agents in various sizes of grid networks provides the same probability of sharing ride that is the crucial impact factor to the average trip time, while the different numbers of agents will influence the number of messages: more agents likely cause more communication messages.

As explained in Section 5.1.1, a client is particular located in central area to avoid boundary effect in other five experiments. If a client was located along boundary alternatively, average shorter travel times and less numbers of messages are expected.

Section 5.1.2 is a realistic representing of a client. Immobile client refers to those unable to walk for physical difficulties, such as elder pedestrians and people with heavy luggage. Mobile client presents others who are more flexible and able to walk to the near intersection by them own. A client insisting on geodesic route behaves as the pre-routing travellers, who learn the direct route from maps. A client willing to make detours is people able to re-plan trip during their travel. This experiment shows that mobile client has the risk of missing potential rides during walk. For example, a walking a client can see a bus passing along if this bus did not exist at departure time of the client, or if the bus was still out of the client’s communication range (Fig 5-3). This risk can be reduced by choosing communication ranges large enough to provide the client with all relevant offers for this period. But extra communication increases energy costs.

Section 5.1.3 examines the typical transportation vehicles (i.e. private cars, taxicabs and mass transit) in the real world. An arbitrary design of the simulation is that mass transit is running on parts of the geodesic route of a client. As discussed before, the restrictions of such kind of hosts, such as pick-up at stops only and fixed timetables, limit their occupation. Figure 5-4 shows that under some conditions buses (i.e. mass transit), if traveling along parallel streets, can even not contribute to a client trips at all. This
happens when the communication range is not large enough to inform the client in time to start walking to the parallel street. Globally adapting the communication range to the speed of the hosts helps in this situation (if other hosts establish multi-hop connectivity). In this way, the communication effort is increased significantly by involving large number of connecting hosts, because most communication effort is used to help transferring transportation information between buses and a client. Alternatively, the agency of bus stops helps. In the simulation, bus stops currently offer information of each approaching bus, in any distance, in an individual message. This increases the number of messages in Figure 5-4, but can be reduced by offering only relevant buses.

The much realistic peer-to-peer shared ride scenario in Section 5.1.4 demonstrates that types of agents affect peer-to-peer shared ride systems under all three communication strategies. In particular, it can be seen that the existence of taxis brings much more efficiency and effectiveness. The reason is discussed in Section 5.1.3.2. But taxis only benefit in the case of looking for quickest trip. It will be another story if a client looks for cheapest trip, because the charge of taxis is most expensive (defined in Table 5-1) among all hosts.

Section 5.1.5 experiments a more sophisticated mobility model of agents: agents prefer walking on their familiar streets. This is also a realistic scenario, because usually in the real world there are plenty of vehicles running in main streets. For those looking for a lift, it is more possible to get a ride along main streets than on others.

So far, all trips are designed to achieve the quickest trip. In practice, a client probably looks at other criteria as well, such as travel fares. The application of other criteria is proved to change the results (Figure 5-8). In Section 5.1.6 \( K \) quickest paths are calculated and decision is made by ranking other criteria in level of importance. A relative small \( K \), three, is employed to reduce computing in the experiment. In the particular case of two criteria, other cheaper trip may exist out of the set of the first
three quickest paths. Therefore a lower travel fare and longer travel time can be expected if \( K \) is bigger. In the extreme case that \( K \) is not smaller than the number of combination of offers, all possible paths are explored and the multi-criteria trip planning becomes the same as finding the cheapest trip.

As mentioned in Section 2.3.3, multilevel programming optimization has a disadvantage that less important criteria may have no influence on the final result. And also, the different importance of criteria needs to be pre-decided. This approach cannot suit the situation that the order of importance cannot be decided or there are several criteria in the same importance. The weighting sums method (see introduction in Section 2.3.3) may solve the problem by setting different weight of each criteria. In two criteria case, the bigger weight represents the more important criterion; equal weights represent the same importance. However for this method, more possible paths are required to draw a safe solution, thereby finding the solution from \( K \) shortest paths may not come up with the best result when \( K \) is relative small. Alternatively, the Pareto-optimal set could be considered to meet the requirement.
Chapter 6
Conclusions and Future Work

The final chapter evaluates the simulation model of a peer-to-peer shared ride system and the simulation results, and draws conclusions with regard to the hypothesis in Section 1.2. To investigate peer-to-peer shared ride systems, this chapter then gives several scenarios of using the simulation model for practice. Limitations and open questions for future work are also discussed.

6.1 Conclusions

Previous research has studied the performance of a peer-to-peer shared ride system with an immobile client following a geodesic route, with homogeneous hosts (now called private cars). It finds that for a peer-to-peer shared ride system, mid-range communication is both efficient and effective (Winter and Nittel 2006). This thesis extends the previous research with types of a client and hosts, and has achieved the following objectives:

- Identification of the typical client (immobile and mobile) and hosts (private cars, taxicabs, mass transport and bus stops), and their relevant properties and behaviors in peer-to-peer shared ride systems.

- Design of a communication protocol for agent communication.

- Implementation of three mobility models of agents: random mobility model, geodesic mobility model and traffic estimation mobility model.

- Introduction of fare models of transportation hosts to choose the cheapest trip.

- Trip planning of a client by application of k shortest paths with multi-criteria
optimization.

- Development of an object-oriented simulation model, in which agents are formalized in specialization architecture. This simulation model enables application of various parameters of environment and agent, and is able to be extended easily.

- Export of negotiation information into text files attached to each agent to facilitate further investigative work.

Agents with different mobility models, and communication and way-finding strategies are examined in experiments, which are conducted under specific settings of environment agent parameters. From the results of experiments, it can be seen that multiple types of agents enrich the choices of a client and as a result lead to trips of generally lower costs. The largest impact is seen with a peer-to-peer shared ride system with Mobile client and all types of host agents, since it provides the most choice for a constant communication range. Particular locations of the requested route, a client’s preferences for trip planning are also factors influencing shared ride trips. Mid-range communication still delivers trips of durations close to a (fictional) unconstrained communication range, but has much lower communication costs. For individual simulation, parameters are specific, but the simulation model allows various settings of parameters. In the previous chapter, various settings of parameters were tested in experiments, however the behaviors of agents do not change; therefore the conclusion is generic: employing other types of agents changes the trips significantly, but mid-range communication is still preferable; and the trip planning with local knowledge comes up with results close to optimal ones. No different behavior is expected even if other realistic street networks are employed. Hence, the hypothesis is supported.
6.2 Evaluation of the Simulation Model

The multi-agent simulation model in this thesis is particularly designed to investigate the performance of realistic peer-to-peer shared ride systems. This model is capable of implementing experiments with various parameters of the street network and agents, as well as testing other mobility models of agents, communication strategies and way-finding strategies. The object-oriented manner of the model also makes developing other types of agents much easier: more agent objects can be extended from an existing class which has the closest properties and behaviors.

Besides the experiments conducted in Chapter 5, the simulation model also can be used for other studies. Observing communication messages allows a deeper understanding of negotiation the process. Additionally, the attached text file of each agent can be seen as a knowledge database, which provides history experience of shared rides and information about other agents. Presented here are several scenarios of using the simulation model:

- Collecting completed transportation information during one simulation run allows finding out a global optimal trip. As the trip planning strategies proposed in this thesis are based on local knowledge, a client risk missing the better rides provided by distant hosts and hosts entering the traffic after they make a booking. Therefore the consequent trip could be sub-optimal from a global view. Upon comparison of the global optimal trip with the actual trip, the quality of trip planning strategies can be assessed. Another research (Guan 2007) is currently working on the scenario.

- By applying virtual observers, another type of agent who is able to query shared ride information in agents’ knowledge database, and collecting responses, it is possible to monitor the performance of participants in shared
ride systems. Shared rides then can be reconstructed based on the collected information to show who travels with whom and when. By this way, agents’ trips can be tracked, and uncompleted trips are suspected if any agent is missing. This manner solves the security problem in peer-to-peer shared ride systems.

- With a knowledge database, agents can learn the traffic history situation of a shared ride area and the performance of other agents, which can be used to assist agents to make more intelligent decisions. For instance, a client can learn the traffic counts at particular intersections, predict the chance of being picked up at specific nodes, and assess potential transfer points in the trip planning process. This idea has been implemented by Gaisbauer (2006).

- Without central management, agents can alternatively obtain dynamic traffic information from other agents. For example, agents can provide the mean speed of a traffic flow in communication messages, and as a result, other agents receiving the messages can construct the traffic distribution and re-plan their remaining trips if necessary.

In summary, the simulation model satisfies the requirements for investigating agent behaviors in peer-to-peer shared ride systems proposed earlier in the thesis. It is also capable and extensible for other studies. Therefore, this model can be seen as a good starting point for more complex and realistic simulations. Other simulation toolkits, such as Repast, are not particularly designed for shared ride systems and often documentation is sparse. An ideal simulation model for realistic peer-to-peer shared ride systems needs to handle GIS data (e.g. the real street networks) and the communication between a large number of agents, to model different types of agents, and also to provide analysis approaches and visualization.
6.3 Limitations and Future Work

This section points out several limitations with the simulation model as proposed in the thesis, and overviews some potential solutions and open questions for future work.

The thesis operates simulations on a grid network, which is a rather simple and overly regular approximation to real street networks. With respect to conclusion about the effectiveness and efficiency of mid-range communication strategy in peer-to-peer shared ride systems, there is no reason to believe that more realistic street networks will bring differences. However from a practical view, more realistic street networks will make the proved communication and way-finding strategies by simulation more convincing to be implemented in the complex transportation systems of the real world. The application of a more realistic street network (e.g. real streets with traffic flow constrains and a meaningful distribution of mass transport lines) is one task needing future attention.

The origin and the destination of a client are arbitrary in simulation experiments. As discussed in Section 5.1.1, the random movement of hosts causes variations resulting in differing positions of the requested route to the boundary of the grid network. In simulation circumstances, the random mobility model of agents does not provide a sufficient real traffic fit. Random routing is a good starting point but needs further thoughts when the number of transfers is considered. A more sophisticated mobility model where a client with knowledge of the street network is able to predict the possibility of being picked up was examined in Section 5.1.5. Another investigation also to reduce the waiting time between transfers is done by Gaisbauer (2006). The speed of transportation hosts is constant in the model, however variable speed may be required in the real traffic situation, for instance when speed limitations are applied on some roads, or when traffic jams happen. More meaningful routings, such as a commute trip between home and work, are also worth considered in the future.
Another future extension of this system comes with admitting other a client to the simulation ($clientNum > 1$). Then passenger capacity of hosts becomes a critical resource. A client would compete with each other, which might recommend more booking ahead. But aggressive booking strategies conflict with the hosts’ interests of traveling with occupied vehicles, since travel plans are highly dynamic. Balancing these interests needs to be investigated.

It is demonstrated that traveling agents benefits from knowledge of travel area. This simulation develops a history database of shared rides information, but does not build a learning framework for agents. In the mid-range communication strategy, agents obtain real time knowledge of local traffic only. For the regions out of the communication range, agents could learn useful information from history knowledge. For this purpose, further investigation also needs to address how to use knowledge experience to promote intelligent decision making.
Bibliography


Leigh, R., 2006: Agent Mobility Model Based on Street Centrality. Final Year Project Report, Department of Geomatics, The University of Melbourne, Melbourne, Australia, 31 pp.


Raubal, M., 2001: Ontology and Epistemology for Agent-Based Wayfinding Simulation. International


Author/s: Wu, Yunhui

Title: Agent behavior in peer-to-peer shared ride systems

Date: 2007-01


Publication Status: Unpublished

Persistent Link: http://hdl.handle.net/11343/39228

File Description: Agent behavior in peer-to-peer shared ride systems

Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.