Whitestein Series
in Software Agent Technologies

Franziska Klügl
Ana Bazzan
Sascha Ossowski
Editors

Applications of Agent Technology in Traffic and Transportation

Birkhäuser

Whitestein Series in Software Agent Technologies

Series Editors:
Marius Walliser
Monique Calisti
Thomas Hempfling
Stefan Brantschen

This series reports new developments in agent-based software technologies and agent-oriented software engineering methodologies, with particular emphasis on applications in various scientific and industrial areas. It includes research level monographs, polished notes arising from research and industrial projects, outstanding PhD theses, and proceedings of focused meetings and conferences. The series aims at promoting advanced research as well as at facilitating know-how transfer to industrial use.

About Whitestein Technologies

Whitestein Technologies AG was founded in 1999 with the mission to become a leading provider of advanced software agent technologies, products, solutions, and services for various applications and industries. Whitestein Technologies strongly believes that software agent technologies, in combination with other leading-edge technologies like web services and mobile wireless computing, will enable attractive opportunities for the design and the implementation of a new generation of distributed information systems and network infrastructures.

www.whitestein.com

Applications of Agent Technology in Traffic and Transportation

Franziska Klügl
Ana Bazzan
Sascha Ossowski
Editors

Birkhäuser Verlag
Basel · Boston · Berlin

Editors:

Franziska Klügl
Dept. of Artificial Intelligence and
Applied Computer Science
University of Würzburg
Am Hubland
97070 Würzburg
Germany

Ana Bazzan
Instituto de Informática, UFRGS
Campus do Vale, Bloco IV
Bairro Agronomia
Av. Bento Gonçalves
91501-970 Porto Alegre
Brazil

Sascha Ossowski
Universidad Rey Juan Carlos
AI Group DIET, ESCET
Campus de Móstoles
Calle Tulipán s/n
28933 Madrid
Spain

2000 Mathematics Subject Classification 68T35, 68U35, 94A99, 94C99

A CIP catalogue record for this book is available from the Library of Congress,
Washington D.C., USA

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

ISBN 3-7643-7258-3 Birkhäuser Verlag, Basel – Boston – Berlin

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. For any kind of use permission of the copyright owner must be obtained.

© 2005 Birkhäuser Verlag, P.O. Box 133, CH-4010 Basel, Switzerland
Part of Springer Science+Business Media
Cover design: Micha Lotrovsky, CH-4106 Therwil, Switzerland
Printed on acid-free paper produced from chlorine-free pulp. TCF ∞
Printed in Germany

ISBN-10: 3-7643-7258-3
ISBN-13: 978-3-7643-7258-3

9 8 7 6 5 4 3 2 1 www.birkhauser.ch

Contents

Preface

Building effective and user-friendly transportation systems is one of the big challenges for engineers in the 21st century. The rapid change of location, enabled by plane, high-speed rail, sea and road travel, has constantly become easier and more natural. These days we travel without any of the difficulties that accompanied taking a trip less than a century ago. All we have to do is to organize and to pick up the transport mode that comes closest to our objectives. In much the same way, many new opportunities for the delivery of goods are being explored and commercially exploited.

So, it is not surprising that there is the intense need to understand, model, and govern transportation systems at both, the individual (micro) and the society (macro) level. Still, this raises significant technical problems, as transportation systems may contain thousands of autonomous, "intelligent" entities that need to be simulated and/or controlled. Therefore, traffic and transportation scenarios are extraordinarily appealing for Distributed Artificial Intelligence, and (multi-)agent technology in particular.

The papers in this book are revised versions of the work presented at the Workshop on Agents in Traffic and Transportation (ATT), at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), held in New York on July 20, 2004. The workshop brought together researchers and practitioners working on agent-based tools for modelling, simulation, and management of transportation systems, in particular in traffic and logistics scenarios. All papers have been thoroughly reviewed by renowned experts in the field. In addition, this book contains an invited contribution by Klaus Dorer and Monique Calisti from Whitestein Technologies.

The first contribution to this book, by Davidsson et al., provides a survey of existing research on agent-based approaches to transportation and traffic management. Perugini et al. present an provisional agreement protocol that facilitates the planning required in transportation scheduling problems. Dorer and Calisti put forward an agent-based approach to solve dynamic multi-vehicle pickup and delivery problems with soft time windows. The work by Ossowski et al. reports on an abstract architecture for traffic management, and its application to two real world domains. Yin and Griss describe SCATEAgents, an agent-based intelligent, flexible, and context-aware multi-modal traveller information system. Bazzan and Klügl show how the Braess Paradox, a well known phenomenon in transportation

engineering, can be overcome by means of information manipulation. Yamashita et al. analyse the effect of route information sharing on traffic congestions. Katwijk et al. report on a test bed for multi-agent road traffic management systems. Conde et al. focus on the problem of effective real-time traffic light control using agent technology. Hallé and Chaib-draa present a collaborative driving system using teamwork for platoon formations. Marchal and Nagel propose a simulation model to account for the effect of secondary activities on route choice. Rindsfüser et al. describe a multiagent simulation for the generation of individual activity programs. Rossetti and Liu report on how the abstraction approach of multiagent systems can be used to represent the complexity inherent in the urban traffic domain. Finally, Gloor at al. present a message-based simulation framework for different kinds of mobility simulations.

We are grateful to all these authors for contributing their latest and inspiring work to this book, as well as to the members of the ATT Program Committee, and the external reviewers, for their critical reviews of submissions. Finally, a deep thanks goes to each of the brave members of the AAMAS-2004 local organization team for their hard work in providing (not only) the ATT-2004 event with a modern, comfortable location, and social program.

We hope you will enjoy reading this book and find inspiration for your own work!

October 2004 Ana Bazzan, Franziska Klügl and Sascha Ossowski

Agent-Based Approaches to Transport Logistics

Paul Davidsson, Lawrence Henesey, Linda Ramstedt,
Johanna Törnquist and Fredrik Wernstedt

Abstract. This paper provides a survey of existing research on agent-based approaches to transportation and traffic management. A framework for describing and assessing this work will be presented and systematically applied. We are mainly adopting a logistical perspective, thus focusing on freight transportation. However, when relevant, work of traffic and transport of people will be considered. A general conclusion from our study is that agent-based approaches seem very suitable for this domain, but that this still needs to be verified by more deployed system.

1 Introduction

The research area of agent technology continues to yield techniques, tools, and methods that have been applied or could be applied to the area of traffic and transportation management. The aim of this paper is to present a consistent view of the research efforts made in this area.

We are mainly adopting a logistical perspective, thus focusing on transportation rather than traffic, and on freight rather than people. In particular, we will not survey the extensive work on agent-based modeling of driver and commuter behavior. Also we will not consider approaches to supply-chain management.

In the next section, the areas where agent technology may be useful will be identified. We then present a framework that will be used to classify and assess the research in the area. This is followed by a systematic survey of the work found in the literature. Finally, we analyze our findings and present some conclusions.

2 Background

The development of distributed and heterogeneous systems, such as software for automation of, and decision support for logistics management, poses significant challenges for system developers. *Agent technology* [73], [75] aims to provide new concepts and abstractions to facilitate the design and implementation of systems of this kind. Parunak [51] lists the following characteristics for an ideal application of agent technology:

- *Modular*, in the sense that each entity has a well-defined set of state variables that is distinct from those of its environment and that the interface to the environment can be clearly identified.
- *Decentralized*, in the sense that the application can be decomposed into stand-alone software processes capable of performing useful tasks without continuous direction from some other software process.
- *Changeable*, in the sense that the structure of the application may change quickly and frequently.
- *Ill-structured*, in the sense that all information about the application is not available when the system is being designed.

- *Complex*, in the sense that the system exhibits a large number of different behaviours which may interact in sophisticated ways.

As most transport logistics applications actually fit Parunak's characterisation rather well, this would suggest that agent technology indeed is a promising approach for this area. However, it is not suitable for all applications. For instance, in applications that are monolithic, centralized, static, well-structured, and simple, agent technology will probably not provide any added value, only unnecessary complexity.

3. Evaluation framework

For each paper surveyed we describe the problem studied, the approach taken to solve it, and assess the results.

3.1 Problem description

Each problem description includes the following three parts: the domain studied, the mode of transportation, and the time horizon considered.

3.1.1 DOMAIN We have chosen to divide the problem descriptions into three domains: *transport*, *traffic* and *terminal*. A transport is an activity where something is moved between point A and B by one or several modes of transport. Problem areas that fall under the category transport are e.g. route planning, fleet management, different sorts of scheduling, i.e., functionalities that takes place to support transportation.

While transport refer to the movement of cargo from one point to another, traffic refers to the flow of different transports within a network. One train set is thus a transport, or part of a transport, that takes part in the train traffic flow. Hence, a transport can be part of several traffic networks (air, waterborne, road, rail,) and a traffic network constitutes of several transports. Typical traffic activities are traffic flow scheduling such as railway slot allocation, air traffic management, and railway traffic management.

Within for example a transport chain where the cargo is transported by truck, rail, ship and truck again, there are interfaces between the different modes. These interfaces represent nodes for re-loading and are referred to as terminals. Terminals can be any fixed place where the cargo is handled and require access to different kinds of resources. Typical terminal activities are resource allocation and scheduling of cranes, forklifts and parts of a facility.

3.1.2 TRANSPORT MODE There are five basic modes of transportation: *road, rail, air, water,* and *pipeline* [64]. Although the use of pipelines often offers the cheapest method in transporting bulk fluids in long distances, we will in this paper not regard this modality.

The water transport via sailing vessels offers one of the most used and less costly means of transporting bulk goods. The use of rail is often associated with bulk items transported less costly than road to far distant markets. The flexibility and often-inevitable use of road for the beginning or final transport mode in a transportation chain makes this the most often used form of transport. Road transport is often associated with

faster delivery in short distances and is attractive to shippers and customers that demand choice and flexibility in scheduling. Finally, air transport mode offers the fastest means of transport and usually the most expensive. This mode is usually reserved for high-valued goods that need to be transported across large distances. The use of air is also considered in short supply times, as in the case of disaster relief.

All freight transport modes can include, for example, fleet management techniques, route and maintenance planning, on-board loading/unloading techniques and on-board computers. In all cases, the emphasis will be on the impact on organizational costs and service levels. Usually in freight logistics, transportation represents the most important single element in logistics costs for most firms [5]. Transportation is a key decision area within logistics due to, on average, a higher percentage of logistics costs associated with this activity than any other logistics activity [5]. The selection of which mode of transport is to be used is dependent on several factors associated with the type of cargo/goods, e.g., requirements on speed, handling, costs, distance, flexibility etc.

Intermodal transportation, refers to "movement of goods in one and the same loading unit or vehicle that uses successively several modes of transport without handling of the goods themselves in changing modes" according to the definition of The European Conference of Ministers of Transport [24]. The definition is valid also for personal travelling that includes two or more different modes of transportation.

One of the primary challenges in intermodal transport management is to coordinate several inter-dependent activities within the transport as well as the communication between the multiple actors involved.

3.1.3 TIME HORIZON Historically, the term logistics referred mainly to issues regarding technical and physical flows of products on an *operational* level. Today, the term includes both strategic and tactical issues beside the operational ones and includes the information flow connected to the physical flow. Therefore, the applications and concepts studied and presented are divided into levels of time perspective; *strategic*, *tactical* and *operational* level of decision-making. This is an established classification that is widely used. It can also be seen as a hierarchy in decision time [61]. We will here by *time horizon* refer to at what stage in the decision-making process the application is used, or is intended to be used. There are two dimensions often distinguished, the level of decision-making and its time frame. There is no definite line of separation, but strategic decision-making typically involves long-term decisions concerning determining what to do, while tactical deals with medium-term issues of setting up an action-list, and operational how to conduct the work set out in more specific terms, i.e. short term issues [61]. The time horizon for these levels is highly domain dependent.

In this study we also include the execution of tasks and real-time controlling functionalities within the operational decision-making. For a transport operator, as an example, a strategic issue to address would be where to locate distribution centres, while a tactical issue would be to tailor the vehicle fleet to satisfy the customer demands, and the operational level would involve scheduling of each and every transport and the controlling function with monitoring and ad-hoc planning if necessary.

As can be seen there is no established definition on time frame or content in the different planning hierarchy, and it is highly dependent on what type of business that is addressed.

3.2 Approach

Each approach is described by the following three parts: the intended usage of the agent system, the type of agents used, and the type of coordination chosen.

3.2.1 USAGE The applications studied can be classified, according to this paper, as either to serve as an automation system, or a decision-support system. An *automation* system can be defined as "having a self-acting mechanism that performs a required act at a predetermined time or in response to certain conditions" [46]. In this context it refers to a system's ability to act upon its decisions, i.e. it has a direct influence on the controlled environment and there is no human involved. On the contrary, a *decision-support system*, DSS, has only at most an indirect impact on the decision-making. A DSS is a system that provides output of some specified type to support the decision process for the user. The user, i.e. the decision-maker, takes the suggested decision(s) into consideration, and then acts. Thus, the final decision is made by a person, not the software system.

3.2.3 COORDINATION (CONTROL, STRUCTURE AND ATTITUDE) Researchers in many fields including computer science, economy, and psychology have studied the area of coordination, which can be viewed as "managing the interdependencies among activities" [45]. In any environment where software agents participate, the agents need to engage in cooperative and/or competitive tasks to effectively achieve their design objectives. From the multi-agent systems perspective coordination is a process in which agents engage in order to ensure that a community of individual agents acts in a coherent manner [48]. Coordination techniques are classified here according to the three dimensions control, structure and attitude.

We capture the authority relationships between agents in the dimension of *control*, which is either centralized or distributed (decentralized). The *MAS structure* corresponds to the set of agents constituting the MAS, their roles, and the communication paths between agents. The structure is either predetermined, i.e., static (the set of agents or their roles do not change during the execution), or is changing dynamically. Finally, the *agent attitude* dimension captures the behavior of agents, which is classified as either benevolent (cooperative), i.e., they will comply with social laws and global goals, or selfish (competitive), where the agents' individual goals, e.g., in a market-based economy, will govern their behavior.

3.3 Results

The main classification of the result of the approaches will be in terms of maturity of the research. However, we will also try to assess the performance and the limitations of the approaches.

3.3.1 MATURITY Agent applications can have varying degree of maturity, i.e., how complete and validated an application is. According to Parunak [52], the description of the maturity of an agent application helps the users to assess how much work that remains to carry out the implementation of the agent application. Furthermore, Parunak has suggested a number of degrees of maturity which formed the basis for our refined classification.

The lowest degree of maturity in the classification is *conceptual proposal*. Here the idea or the principles of the proposed application is described with its general characteristics, e.g. if the model is simple or complex. In the literature the term *conceptual model* is quite well-established and well-defined. However, we prefer the more open term conceptual proposal since it otherwise could be more difficult to fit in all applications according to the classification.

The next level in the classification is *simulation experiments*. Here the application has been tested in a simulation environment. The data used in the simulated experiment can either be real data, i.e. taken from existing systems in the real world, or data that is not real, i.e. artificial, synthetic or generated. Further, the type of data has been divided into limited/partial or full-scale data. The full-scale data represents data for a whole system, while the limited/partial data only covers parts of the system.

Field experiment indicates that experiment with the application has been conducted in the environment where the application is supposed to be applied. As in the simulated experiment, the field experiment is also divided into limited/partial and full-scale. The final level, *deployed system*, indicates that the system has been implemented in the real world and also has been or is in use. This is the most mature type of agent applications.

3.3.2 EVALUATION COMPARISON If a new approach is developed to solve a problem which has been solved previously using other approaches, the new approach should be compared to those existing approaches. Such an evaluation could be either *qualitative*, by comparing the characteristics of the approaches, or *quantitative*, by different types of experiments.

3.4 Summary of framework

Table 1 on the next page summarizes the framework for describing and assessing the agent-based approaches to logistics. The Appendix provides a table listing the published work in the area of agent-based approached to transport logistics that we have encountered is classified according to this framework. The papers in the table are first sorted according to domain and then according to mode of transportation. In the case where several papers have been published regarding the same project, we have chosen the most recent publication and/or the most widely available.

	Aspect	**Categories**
Problem description	Domain	1. Transport 2. Traffic 3. Terminal
	Transport mode	1. Air 2. Rail 3. Road 4. Sea 5. Intermodal
	Time horizon	1. Operational 2. Tactical 3. Strategical
Approach	Usage	1. Automation system 2. Decision support system
	Control	1. Centralized 2. Distributed
	MAS structure	1. Static 2. Dynamic
	Agent attitude	1. Benevolent 2. Selfish
Results	Maturity	1. Conceptual proposal 2. Simulation experiment 2.1. artificial data 2.1.1. limited 2.1.2. full-scale 2.2. real data 2.2.1. limited 2.2.2. full-scale 3. Field experiment 3.1. limited 3.2. full-scale 4. Deployed system
	Evaluation comparison	1. None 2. Qualitative 3. Quantitative

Table 1. Classification framework.

4. Analysis of Survey

The survey shows that agent technology has been applied to many different problem areas within transport logistics. Often these agent approaches are distributed and very complex by nature, such as: planning and scheduling, fleet management, transport scheduling, traffic management, and traffic control. In the work reviewed, there was an even distribution between the three domains (transport, traffic, and terminal), whereas the modes of transportation were dominated by air, road and intermodal. It is worth noting that very little work has been done studying strategic decision making. In addition, only a few of the publications concerning air and rail deal with transport-centered issues.

Most of the *rail*-related publications address problems of allocating slots for the railway network, i.e., timetabling. This is a problem seldom found within the other modes of transport besides air traffic (even though railway slot allocation differs significantly from air traffic slot allocation). Market-based approaches [19] have appealed to several of the researchers, where the coordination mechanism is very similar to the negotiation that takes place in practice. In addition, some publications study resource allocation for specific rail transports, but these problems are not modal-specific to the same extent as the slot allocation problem. Several of the approaches have been evaluated experimentally, but no deployed system has been found. Methods that are alternative to agent technology for these kinds of problems are often centralized optimization and simulation technologies.

Regarding the publications that relate to *air* traffic and transportation, the studies on air traffic management is dominating and agent technology seems to have been applied to this problem area for more than a decade. The main topic addressed is distributed air traffic management using free flight, i.e., the aircrafts are allowed to choose their speed and path in real-time and air traffic restrictions are only applied when air space separation is required. No application focusing on airport slot allocation for a tactical setting has been found, which is surprising as many railway scheduling applications exist. Only a few publications in the air domain deal with transport related issues.

In the papers on *road transports*, most of the problems concern transport scheduling, i.e., allocating transport tasks to vehicles. The approaches are distributed and include negotiation in various manners, such as the contract net protocol, and sometimes they are market-based. However, also Multi Agent Based Simulation (MABS) is used in some applications. The agents in these applications represent different roles, e.g., a company, a truck, a customer etc. The transport applications are on a tactical level and the purpose is most often to serve as a DSS to a transport operator since the problem is complex and need some human supervision before the final transport task allocation. Alternative methods to agent technology in road transport are classical mathematical methods, and operations research.

In the *road traffic* domain most of the problems concern traffic management and control to deal with for example congestion of the roads. The applications are designed to inform drivers about the traffic situation and give recommendations, regulate the traffic with signals and messages, and so on. A couple of the applications deal with public transport management where the actual status of the vehicles is compared to the

planned status, e.g. a timetable. The majority of the systems are on the operational level and most of the applications function as a DSS, but some are designed to serve as automation systems. Alternative methods mentioned in the papers, are evolutionary algorithms, knowledge-based systems, neural networks, and fuzzy theory.

Concerning the *sea* mode of transportation, most application of agents have been trying to increase the efficiency of the container terminal operations. Many papers tend to focus specifically at the marine-side interface whilst disregarding the other processes in the terminal that determine overall terminal performance, e.g., the stacking of containers. The terminals are characterized as complex and dynamic systems and researchers find the relationships between the many actors involved having both common and conflicting goals, in which vast amounts of information are not processed adequately to encourage the use of agents. Several papers focus exclusively on the operational processes of communication between the gantry cranes and the straddle carriers in order to reduce idle time and the number of times that a container is handled, whereas a couple of papers deal with tactical and strategic decisions. Unfortunately, the majority of the papers reviewed do not state clearly the type of agent approaches used or how their agents are able to communicate and make decisions. Interestingly, within the sea mode of transportation, most research has focused primarily on the terminal domain with very few papers considering the traffic and transport domains.

Of the reviewed publications regarding *intermodal* transportation, primarily the combination of road and rail has been considered. The problems studied are usually to coordinate several tasks for a specific transport, such as slot request, terminal handling and allocating transport services. The approach is typically to identify a set of different roles, similar to the real-world functions, and allocate agents for each of these. Although only a few publications were found, the work in this area seems to be extensive at the moment and rapidly developing. Some alternative methods for these problems are discrete-event simulation and optimization. In practice, however, they are more often dealt with in an ad-hoc manner with a mix of human-intervention and spread-sheet analysis. For this domain, as for the other domains, the benefits of using agent-based approaches are not explicitly discussed.

The main reasons mentioned in the papers for adopting an agent-based approach are: facilitates distributed control, ability to cope with partial and noisy data, and ability to model complex problems. Although the ability to distribute control is the most cited reason, it is interesting to note that 30% of the projects surveyed make use of centralized control. Also, only half of the applications utilize the possibility of dynamic MAS structures, which is an often cited strength of agent technology. A majority of the work (64%) concerns the use of agent technology in decision support systems.

Regarding the maturity, the vast majority of the approaches surveyed have just reached the level of conceptual proposal (30%) or simulation with limited or artificial data (53%). An obvious danger with simulation experiments based on artificial or partial data is that abstractions are made that simplifies the problem to a point where the results are not relevant for real-world problems. The table below illustrates how the maturity of the projects has developed through the years, i.e. presenting the number of projects found per year and maturity level. The most recent publication found for each project is included. As can be seen, only one deployed system could be found.

Maturity	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
1				1		1	1			2	5	7		
2.1.1	1							1		3	3	1	3	1
2.1.2				1							1			1
2.2.1		1			1			5	1	2	1	1	1	
2.2.2	1						1			1	2	1		
3.1									2			1		
3.2														
4											1			

Table 2. Maturity level over the years.

In two thirds of the approaches surveyed, agents are applied to solve problems without considering current or alternative approaches to solve these problems. Of those that actually are making comparisons, the majority make only qualitative comparisons. The alternative approaches regarded in the papers are, e.g., for traffic management: evolutionary algorithms, knowledge-based systems, neural networks, fuzzy systems; and for transport scheduling: classical mathematical and operations research methods, i.e., mainly centralized approaches.

5 Conclusions

While producing the survey we have identified a number of positive aspects of the current state of agent-based approaches to logistics:

- Many different approaches have been suggested and investigated.
- Many of the logistics problems that have been studied have characteristics that closely match those of an ideal agent technology application very well.
- Especially in the areas of air and road traffic management agent technology seems to have contributed significantly to the advancement of state-of-the-art.

However, there are also some things that can be improved:

- The maturity of the research; few fielded experiments have been performed and very few deployed systems could be found.
- The suggested agent-based approaches are often not evaluated properly; comparisons with existing techniques and systems are rare. Both qualitative assessments explaining the pros and cons of agent technology compared to the existing solutions, and quantitative comparisons to these solutions based on experiments, are desired.
- Some problem areas seem under-studied, e.g., the applicability of agent technology to strategic decision-making within transportation logistics.

Appendix Survey results

	Problem Description			Approach				Results	
Paper	*Domain*	*Mode*	*Time horizon*	*Usage*	*Control*	*MAS structure*	*Agent Attitude*	*Maturity*	*Evaluation Comparison*
[12]	Transport	Air	Oper.	Auto	Centr.	Dyna.	both	1	None
[53]	Transport	all	Tact.	DSS	Distr.	Dyna.	Selfish	2.2.1	both
[77]	Transport	Air	Tact.	DSS	Distr.	Static	Benev.	3.1	Qualit.
[15]	Transport	Rail	Tact.	DSS	Centr.	Static	Benev.	2.2.1	None
[62]	Transport	Rail	Tact.	DSS	Centr.	Static	Benev.	1	None
[9]	Transport	Road	Tact.	DSS	Distr.	Dyna.	Benev.	1	Qualit.
[29]	Transport	Road	Tact.	DSS	Distr.	Dyna.	Benev.	2.1.1	both
[41]	Transport	Road	Tact.	DSS	Distr.	Dyna.	Selfish	2.2.1	both
[58]	Transport	Road	Tact.	Auto	Distr.	Dyna.	Selfish	2.2.1	Qualit.
[11]	Transport	Intermod.	Oper.	Auto	Distr.	Static	Selfish	2.1.1	None
[13]	Transport	Intermod.	Oper.	DSS	Distr.	Dyna.	Selfish	2.2.1	None
[23]	Transport	Intermod.	Tact. & Oper.	DSS	Distr.	Static	Selfish	1	None
[54]	Transport	Intermod.	Strat.	DSS	Distr.	Dyna.	Benev.	1	None
[6]	Transport	Intermod.	Strat.	DSS	Distr.	Static	both	2.1.1	None
[1]	Transport	Intermod.	Strat. & Tact.	DSS	Distr.	Dyna.	Benev.	1	None
[76]	Transport	Intermod.	all	Auto	Distr.	Dyna.	Benev.	2.2.1	None
[14]	Transport Terminal	Air	Oper.	Auto	Centr.	Dyna.	Benev.	3.1	None
[3]	Traffic	Air	Oper.	Auto	Distr.	Dyna.	Benev.	2.1.1	None
[16]	Traffic	Air	Oper.	both	Centr.	Dyna.	Benev.	2.1.1	Quant.
[27,28]	Traffic	Air	Oper.	Auto	Distr.	Dyna.	Selfish	2.1.2	Quant.
[39]	Traffic	Air	Oper.	DSS	Distr.	Dyna.	Benev	1	None
[42,65]	Traffic	Air	Oper.	DSS	Distr.	Dyna.	Selfish	1	None
[44]	Traffic	Air	Oper.	DSS	Centr.	Dyna.	Benev.	2.2.2	None
	Problem Description			**Approach**				**Results**	
Paper	*Domain*	*Mode*	*Time horizon*	*Usage*	*Control*	*MAS structure*	*Agent Attitude*	*Maturity*	*Evaluation Comparison*
[47]	Traffic	Air	Oper.	DSS	Centr.	Dyna.	Benev.	1	None
[49]	Traffic	Air	Oper.	Auto	Centr.	Dyna.	Benev.	2.1.1	None
[73]	Traffic	Air	Oper.	DSS	Distr.	Dyna.	Selfish	2.1.1	None

	Problem Description			Approach				Results	
[69]	Traffic	Air	Oper.	Auto	Distr.	Static	Selfish	2.1.1.	None
[7,8]	Traffic	Rail	Tact.	DSS	Centr.	Static	Benev.	2.2.1	None
[10]	Traffic	Rail	Tact.	DSS	Distr.	Static	Selfish	2.2.1	None
[20]	Traffic	Rail	Oper.	DSS	Centr.	Dyna.	Benev.	2.2.1	Qualit.
[25]	Traffic	Rail	Oper.	DSS	Distr.	Dyna.	Benev.	1	None
[50]	Traffic	Rail	Tact.	Auto	Distr.	Static	Selfish	2.1.1	Quant.
[66]	Traffic	Rail	Oper.	DSS	Distr.	Static	Benev.	1	None
[26]	Traffic	Road	Oper.	DSS	Distr.	Dyna.	Benev.	2.1.1	None
[38] InTRYS	Traffic	Road	Oper.	DSS	Centr.	Static	Benev.	4	both
[18]	Traffic	Road	Oper.	Auto	Distr.	Static	Benev.	2.2.2	Quant.
[2]	Traffic	Road	Oper.	both	Distr.	Dyna.	Benev.	2.1.2	Qualit.
[4]	Traffic	Road	Oper.	DSS	Distr.	Static	Selfish	2.2.1	Qualit.
[30]	Traffic	Road	Oper.	Auto	Centr.	Static	Benev.	2.2.1	None
[34]	Traffic	Road	Oper.	DSS	Distr.	Static	Benev.	3.1	None
[67]	Traffic	Road	Oper.	Auto	Distr.	Dyna.	Benev.	2.1.1	Qualit.
[68]	Traffic	Road	Tact.	Auto	Distr.	Static	Selfish	1	None
[35]	Terminal	Road	Oper.	Auto	Distr.	Dyna.	Benev.	2.2.2	None
[40]	Terminal	Sea	Oper.	Auto	Centr.	Static	Benev.	1	None
[43,75]	Terminal	Sea	Tact.	DSS	Centr.	Static	Benev.	2.1.1	None
[17,55]	Terminal	Sea	Oper.	Auto	Centr.	Static	Benev.	1	None
[64]	Terminal	Sea	Oper.	Auto	Distr.	Static	Benev.	2.1.1	None
[21,22]	Terminal	Intermod.	Oper.	Auto	Centr.	Dyna.	Benev.	2.2.2	Quant.
[32,33]	Terminal	Intermod.	Tact.	DSS	Distr.	Static	Benev.	2.2.2	Qualit.
[36]	Terminal	Intermod.	Strat.	DSS	Distr.	Static	Selfish	1	None
[37]	Terminal	Intermod.	Oper.	Auto	Distr.	Static	Benev.	1	None

References

[1] Abouaïssa, H., Nicolas, J.C., Benasser, A., Czesnalowicz, E., Formal specification of Multi-Agent Systems: Approach based on meta-models and high-level Petri-nets - Case study of a transportation systems, *Second IEEE International Conference on Systems, Man and Cybernetics, Volume 5*, IEEE Computer Society Press, 2002.

[2] Adler, J.L., Satapathy, G., Manikonda, V., Bowles, B., Blue, V.J., A multi-agent approach to cooperative traffic management and route guidance, *Transportation Research Part B*, Vol. 39, pp. 297-318, 2005.

[3] Allo, B., Guettier, C., Lécubin, N., A demonstration of dedicated constraint-based planning within agent-based architectures for autonomous aircraft, *IEEE International Symposium on Intelligent Contro*l, pp. 31-38, 2001.

[4] Balbo, F., Pinson, S., Toward a Multi-agent Modelling Approach for Urban Public Transportation Systems, *Engineering Societies in the Agents World II*, , LNAI 2203, pp. 160-174, Springer, 2001

[5] Ballou, R.H., *Business Logistics Management*, 4th Ed. Prentice Hall International, 1999.

[6] Bergkvist M., Davidsson, P., Persson, J.A., and Ramstedt, L., A Hybrid Micro-Simulator for Determining the Effects of Governmental Control Policies on Transport Chains, *Multi-Agent-Based Simulation IV*, LNAI Vol. 3415, Springer, 2005.

[7] Blum, J., Eskandarian, A., Enhancing intelligent agent collaboration for flow optimization of railroad traffic, *Transport Research Part A*, 2002.

[8] Blum, J., Eskandarian, A., "Domain-specific genetic agents for flow optimization of freight railroad traffic, *8th International Conference on Computers in Railways*, Wessex Institute of Technology, Lemnos, Greece, 2002.

[9] Bouzid, M., On-line transportation Scheduling using Spatio-Temporal Reasoning, *10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic*, IEEE, 2003.

[10] Brewer, P.J., Plott, C.R., A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks, *Int. Journal of Industrial Organisation*, Vol. 14: 857-866, 1996.

[11] Buchheit, M., Kuhn, N., Müller, J. P., Pischel, M., MARS: Modeling a multiagent scenario for shipping companies. *European Simulation Symposium*, Society for Computer Simulation, 1992.

[12] Budenske, J., Newhouse, J., Bonney, J., Wu, J., Agent-based schedule validation and verification", *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1: 616-621, 2001.

[13] Bürckert, H-J., Funk, P., Vierke, G., An intercompany dispatch support system for intermodal transport chains, *33rd Hawaii International Conference on System Science*, 2000.

[14] Burstein, M., Ferguson, G., Allen, J., Integrating agent-based mixed-initiative control with an existing multi-agent planning system, *Fourth International Conference on MultiAgent Systems*, pp. 389-390, 2000.

[15] Böcker, J., Lind, J., Zirkler, B., Using a multi-agent approach to optimise the train coupling and sharing system, *European Journal of Operational Research*, Vol. 134:242-252, 2001.

[16] Callantine, T., Prevôt, T., Battiste, V., and Johnson, W., Agent-based support for distributed air/ground traffic management simulation research, *AIAA 2003-5371*, Reston, VA, U.S.A, American Institute of Aeronautics and Astronautics, 2003.

[17] Carrascosa, C., Rebollo, M., Vicente, J., Botti, V., A MAS Approach for Port Container Terminal Management: The Transtainer Agent. Proceedings of *SCI, IFSR*, 2001.

[18] Choy, M.C., Srinivasan, D., Cheu, R.L., Cooperative, Hybrid Agent Architecture for Real-time traffic control, *IEEE Transactions on Systems, Man and Cybernetics*, Part A, Vol. 33, Issue 5, pp. 597-607, Sept. 2003.

[19] Clearwater, S.H. (ed.), *Market-Based Control: Some early lessons*, World Scientific, 1996.

[20] Cuppari, A., Guida, L., Martelli, M., Mascardi, V., Zini, F., Prototyping freight trains traffic management using Multi-Agent Systems, *IEEE International Conference on Information, Intelligence and Systems*, 1999.

[21] Degano, C., Di Febbraro, A., Fornara, P., Fault diagnosis in an intermodal container terminal. *8th IEEE International Conference on Emerging Technologies and Factory Automation*: 433-440, Vol. 2, 2001.

[22] Degano, C., Pellegrino, A., Multi-Agent Coordination and Collaboration for Control and Optimization Strategies in an Intermodal Container Terminal, *IEEE International Engineering Management Conference,* 2002.

[23] Dong, J-W., Li, Y-J., Agent-based design and organization of intermodal freight transportation systems, *Second International Conference on Machine Learning and Cybernetics*, pp. 2269-2274, 2003.

[24] European Conference of Ministers of Transport, *Terminology of Combined Transports*, United Nations, Geneva, 2001.

[25] Fernández, A., Alonso, E., Ossowski, S., A multiagent architecture for train fleet management, *Fifth UK Workshop on Multi-Agent Systems*, 2002.

[26] Fernández, A., Alonso, E., Ossowski, S., A multiagent architecture for bus fleet management, *Integrated Computer-Aided Engineering*, Vol. 11(2), 2004.

[27] Findler, N.V., Lo, R., Distributed artificial intelligence approach to air traffic control, *Control Theory and Applications*, Vol. 138(6): 515-524, 1991.

[28] Findler, N.V., Elder, G.D., Multiagent coordination and cooperation in a distributed dynamic environment with limited resources, *Artificial Intelligence in Engineering*, Vol. 9(3): 229-238, 1995.

[29] Fischer, K., Chaib-draa, B., Müller, J. P., Pischel, M., Gerber, C., A Simulation Approach Based on Negotiation and Cooperation Between Agents: A Case Study, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 29(4): 531-545, 1999

[30] France, J., Ghorbani A.A., A Multiagent System for Optimizing Urban Traffic, *International Conference on Intelligent Agent Technology*, IEEE, 2003.

[31] Funk, P., Vierke, G., Bürckert, H-J., A Multi–Agent Systems Perspective on Intermodal Transport Chains, *Logistik-Management-Tagung LMT-99*, 1999

[32] Gambardella, L. M., Mastrolilli, M., Rizzoli, A. E., Zaffalon, M., An optimization methodology for intermodal terminal management. *Journal of Intelligent Manufacturing*, Vol. 12(5-6): 521-534, 2001.

[33] Gambardella, L.M., Rizzoli, A. E., Zaffalon, M., Simulation and planning of an inter-modal container terminal. *Simulation* 71(2): 107–116, 1998.

[34] García-Serrano, A.M., Teruel Vioque, D., Carbone, F., Méndez, V.D., FIPA-compliant MAS development for road traffic management with a Knowledge-Based approach: the TRACK-R agents, *Challenges in Open Agent Systems '03 Workshop*, 2003.

[35] Goldsmith, S. Y., et al., A Multi-agent System for Coordinating International Shipping. *Agent Mediated Electronic Commerce, First International Workshop on Agent Mediated Electronic Trading*, Springer, 1998.

[36] Henesey, L., Notteboom, T., and Davidsson, P., Agent-based simulation of stakeholders relations: An approach to sustainable port and terminal management. *International Association of Maritime Economists Annual Conference*, 2003(a).

[37] Henesey, L., Wernstedt, F., Davidsson, P., Market-Driven Control in Container Terminal Management. *2nd International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2003(b).

[38] Hernández, J.Z., Ossowski, S., García-Serrano, A., Multiagent architectures for intelligent traffic management systems, *Transportation Research Part C*, Vol. 10: 473-506, 2002

[39] Iordanova, B.N., Air traffic knowledge management policy, *European Journal of Operations Research*, Vol. 146: 83-100, 2003.

[40] Itmi M., M. D., Pecuchet J.-P., Serin F., Villefranche L., Eco-problem solving for containers stacking. Systems, Man and Cybernetics, *IEEE International Conference on Intelligent Systems for the 21st Century*: vol. 4: 3810-3815, 1995.

[41] Kohout, R., Erol, K., In-Time Agent-Based Vehicle Routing with a Stochastic Improvement Heuristic, *11th Conference on Innovative Applications of Artificial Intelligence*, 1999

[42] Košecká, J., Tomlin C., Pappas, G., Sastry, S., Generation of conflict resolution maneuvers in air traffic management, *IEEE-RSJ International Conference on Intelligent Robots and Systems 97*, 1997.

[43] Lee, T.-W., Park, N-K., Lee, D-W., Design of Simulation System for Port Resources Availability in Logistics Supply Chain, *International Association of Maritime Economists Annual Conference*, 2002.

[44] Ljungberg, M., Lucas, A., The OASIS Air Traffic Management System, *Pacific Rim International Conference on Artificial Intelligence*, Seoul, Korea, 1992.

[45] Malone, T., Crowston, K.: The interdisciplinary study of coordination, *ACM Computing Surveys*, Vol. 26(1), 1994.

[46] McGraw-Hill Encyclopedia of Science & Technology, www.accessscience.com/server-java/Arknoid/science/AS (2003-03-25).

[47] Nguyen-Duc, M., Briot, J-P., Drogoul, A., An application of Multi-agent coordination techniques in air traffic management, *International Conference on Intelligent Agent Technology*, pp. 622-625, 2003.

[48] Nwana, H.S., Lee, L., Jennings, N.R.: Co-ordination in software agents systems, *BT Technol J*. Vol. 14(4), 1996.

[49] Painter, J.H., Cockpit multi-agent for distributed air traffic management, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002.

[50] Parkes, D., Ungar, L., An auction-based method for decentralized train scheduling, *AGENTS'01*, 2001.

[51] Parunak, H.V.D., Industrial and Practical Applications of DAI, In *Multiagent Systems*, (ed.) G. Weiss. MIT Press, 1999.

[52] Parunak, H.V.D. Agents in Overalls: Experiences and Issues in the Development and Deployment of Industrial Agent-Based Systems, *International Journal of Cooperative Information Systems* 9(3): 209-227, 2000.

[53] Perugini, D., Lambert, D., Sterling, L., Pearce, A., Provisional Agreement Protocol for global Transportation Scheduling, *International Workshop on Agents in Traffic and Transportation*, New York, U.S.A., 2004.

[54] Proshun, S-R., Carter, J., Field, T., Marshall, J., Polak, J., Schumacher K., Song, D-P., Woods, J., Zhang, J., Container World: Global agent-based modelling of the container transport business, *4th International Workshop on Agent-Based Simulation*, Montpellier, France, 2003.

[55] Rebollo, M., Vicente, J., Carrascosa, C., Botti, V., A MAS Approach for Port Container Terminal Management. *3rd Iberoamerican workshop on DAI-MAS*, pp. 83-94, 2001.

[56] Rizzoli, A., Funk, P., Gambardella L., An architecture for agent-based simulation of rail/road transport, *International Workshop on Harbour, Maritime & Multimodal Logistics Modelling and Simulation*, 2002.

[57] Rizzoli A.E., Fornara N., Gambardella L.M., A Simulation tool for combined rail/road transport in intermodal terminals, *MODSIM 1999*, Modelling and Simulation Society of Australia and New Zealand, 1999.

[58] Sandholm, T., An implementation of the contract net protocol based on marginal cost calculations, *Eleventh National Conference on Artificial Intelligence*, 1993

[59] Sawamoto, J., Tsuji, H., Koizumi, H., Continuous Truck Delivery Scheduling and Execution system with Multiple Agents, in Kuwabara, K. and Lee, J. (Eds.), *PRIMA 2002*, LNAI 2413, pp. 190-204, Springer, 2002.

[60] Schneeweiss, C., *Hierarchies in distributed decision making*, Springer-Verlag, ISBN 3-540-65585-9, 1999.

[61] Shillo, M., Vierke, G., Multidimensional utility vectors in the transportation domain, *ECAI Workshop on Agent Technologies and Their Application Scenarios in Logistics*, 2001.

[62] Sjöland, T., Kreuger, P., Aronsson, M., Heterogenous Scheduling and Rotation, in A.C. Kakas, F. Sadri (Eds.): *Computational Logic: Logic Programming and Beyond. Essays in Honour of Robert A. Kowalski, Part I*. LNAI 2407, Springer, 2002.

[63] Stock, J. R. and D. M. Lambert, *Strategic Logistics Management*. Boston, U.S.A., McGraw-Hill Irwin, 2001.

[64] Thurston, T. and H. Hu, Distributed Agent Architecture for Port Automation. *26th International Computer Software and Applications Conference*, IEEE Computer Society, 2002.

[65] Tomlin, C., Pappas, G.J., Sastry, S., Noncooperative conflict resolution, *36th Conference on Decision & Control*, pp. 1816-1821, 1997.

[66] Törnquist J., Davidsson P., A Multi-Agent System Approach to Train Delay Handling, *ECAI-02 Workshop on Agent Technologies in Logistics*, 2002.

[67] van Katwijk, R.T., De Schutter, B., Hellendoorn, J., van Koningsbruggen, P., van Gosliga, S.P., A Test Bed For Multi-Agent Control Systems In Road Traffic Management, *Workshop on Agents in Traffic and Transportation*, New York, U.S.A., 2004.

[68] van Katwijk, R., van Koningsbruggen, P., Coordination of traffic management instruments using agent technology, *Transportation Research Part C*, Vol. 10: 455-471, 2002.

[69] Vilaplana, M. A., Goodchild, C., Application of distributed artificial intelligence in autonomous aircraft operations, *20th Conference on Digital Avionics Systems*, Vol. 2, 2001.

[70] Wangermann, J., Stengel, R., Distributed optimization and principled negotiation for advanced air traffic management, *IEEE International Symposium in Intelligent Control*, 1996.

[71] Wangermann, J., Stengel, R., Principled negotiation between intelligent agents: a model for air traffic management, *Artificial Intelligence in Engineering*, Vol. 12: 177-187, 1998.

[72] Weiss G., *Multi-Agent Systems*, MIT Press, Cambridge, 1999.

[73] Wollkind, S., Valasek, J., and Ioerger, T.R., Automated Conflict Resolution for Air Traffic Management Using Cooperative Multiagent Negotiation. *AIAA Guidance, Navigation, and Control Conference*, 2004.

[74] Wooldridge M., *An Introduction to Multi-Agent Systems*, John Wiley & Sons, London, 2002.

[75] Yi, D.W., Kim, S.H., Kim., N.H., Combined Modeling with Multi-Agent Systems and Simulation: Its Application to Harbor Supply Chain Management. *35th Hawaii International Conference on System Sciences*, 2002.

[76] Zhu, K., Bos, A., Agent-based design of intermodal freight transportation systems, *NECTAR Conference*, Delft, 1999.

[77] Zhu, K., Ludema, M., van der Heijden, R., Air cargo transports by multi-agent based planning, *33rd Hawaii International Conference on System Science*, 2000.

Paul Davidsson, Lawrence Henesey, Linda Ramstedt, Johanna Törnquist
and Fredrik Wernstedt
Department of Systems and Software Engineering
Blekinge Institute of Technology Soft Center
372 25 Ronneby
Sweden

Provisional Agreement Protocol for Global Transportation Scheduling

Don Perugini, Dale Lambert, Leon Sterling, Adrian Pearce

Abstract: The global transportation scheduling problem is complex, decentralised, open and dynamic. It typically requires the services of many transport organisations to transport partial quantities along partial routes to fulfill a transportation task. We have applied agents to address this problem. The Provisional Agreement Protocol (PAP) was developed to facilitate the planning required in our transportation problem. A greedy PAP approach has been implemented for the complex global transportation problem, allowing partial quantity and route bids, and backtracking if an infeasible solution is encountered. In this paper, we present the PAP, together with some improvements over that which has been previously presented. Further implementation details and formal evaluation are provided. Our implementation allows a wider range of transportation problems to be solved than previous approaches.

Introduction

Transportation scheduling is vital in military and commercial logistics. It is a major component of the Multi-Agent Logistics Tool (MALT) that is being developed by DSTO for the Australian military to assist in their logistics planning [1]. Transportation scheduling research usually investigates scheduling requirements for a *single* organisation to schedule their own transportation assets, which they have control over, along local routes. Solutions to these problems are typically centralised, i.e. *all* information is processed by a single decision making entity (that may contain distributed/multiple processors).

Military and commercial organisations have changed the way they do business. Increasing deregulation and outsourcing leads to these organisations acquiring services from other organisations in order to achieve their business goals. The Australian military is increasingly dependent on the use of coalition (military) and civilian organisations to achieve their transportation goals. As a result, organisations may not have control over the transportation assets that are used to achieve their transportation goals, nor access to the information that governs their behaviour. Transportation scheduling must therefore be performed in a decentralised open market, cooperating with other organisations that make their own self-interested decisions on how their transportation assets are utilised, in order to obtain the required services to achieve transportation goals. Centralised transportation scheduling approaches are not well suited to the decentralised environment [2].

Military transportation goals are typically to transport large quantities of resources on a global scale. In the commercial sector, due to globalisation, the transportation requirements may also be global. Therefore, an organisation may require many transportation services from numerous transportation organisations to achieve its transportation goal, where each single transportation service (asset) may only be able to transport part of the resource's quantity over only part of the route.

To add to the complexity, the domain is open and dynamic. Transportation organisations of varying capabilities may enter or leave the system at will, and their capabilities may be continually changing, even during the scheduling (*planning*) process, because, for example, they may be concurrently cooperating and providing transportation services to other organisations.

The Provisional Agreement Protocol (PAP) [3, 4] facilitates distributed agent planning in a decentralised, open and dynamic environment. Agents represent organisations, that cooperate via a bidding process, in order to trade transportation services, which are assembled to form a plan to achieve the transportation goal(s). PAP is based on the Extended Contract Net Protocol (ECNP) [5], but allows backtracking of single bids, and permits partial route, in addition to partial quantity, bids. In the following sections, PAP is presented, some improvements to PAP are introduced, as well as further implementation details and formal evaluation.

Problem Definition

A set of (root) transportation tasks, $t = \{t_1, ..., t_{nrt}\}$ must be achieved, where $t_n = \{n_n, q_n, est_n, lft_n, src_n, dst_n\}$, where package n_n of quantity q_n is to be transported from source location src_n to destination dst_n, with earliest start time est_n and latest delivery time lft_n. There are a set of transportation assets, $ta = \{ta_1, ..., ta_{nta}\}$, which can be used to achieve t. $ta_i = \{cap_i, R_i, lp_i\}$ has capacity cap_i, set of routes $R_i = \{r_1, ..., r_{nr}\}$ that it can service, and a local plan that contains a time ordered sequence of actions, $lp_i = \{a_1, ..., a_{na}\}$, $a_j = \{tsa_j, tfa_j, lsa_j, lfa_j, inv_j\}$, where tsa_j and tfa_j are the action start and finish times (dependent on the ta_i's speed, loading time, etc.), respectively, lsa_j and lfa_j are the action start and finish locations (the route r_j), respectively, and inv_j is the inventory, $inv_j = \{i_1, ..., i_{ni}\}$, where $i_k = \{n_k, c_k, p_k\}$, the inventory item carries quantity c_k of package n_k at price p_k. $c_1 + c_2 + ... + c_k \leq cap_i$. There must be enough time for the transport asset to get from the end location lfa_i of one action to start location lsa_{i+1} of the next action. An action may transport partial quantity and route of a task t_n, therefore, if $n_k = n_n$, $0 < c_k \leq q_n$, and lsa_i and lfa_i may or may not be equal to src_n and dst_n, respectively. Packages may be picked up and delivered at any location, and all locations are assumed to have infinite storage capacity.

A distributed transportation plan $P_t = \{\{ta_i, a_j, inv_j\} \mid ta_i \in ta \ \& \ a_j \in lp_i \ \& \ inv_j \in a_j\}$ to achieve t consists of the set of actions and inventories by ta that are associated with t. If $tdel_n$ is the delivery time of package n_n in P_t, then $tdel_n \leq lft_n$. The price to deliver package n_n is pr_n , which is the sum of all inventory prices p_k in P_t associated with n_n. The plan P_t has a cost value cv associated with it, which is a function of $tdel$ and pr. We assume that the earlier the delivery time $tdel$, and the lower the price pr, for all tasks in

t, then the better the plan P_t, and hence the lower the cv. The aim of transportation planning is to find a P_t such that cv is minimised.

Complexity

Greenwald and Dean [6, 7] have investigated a similar problem, called the grey-hound package scheduling problem. It allows *ta* to perform part of the route of transporting a package. They make the restriction that the *ta* schedules are fixed, simplifying the problem [6, 7]. They show that the optimisation problem is still NP-complete [6]. In our transportation domain, this restriction is not made, allowing *ta* to perform any transportation action at any time. Therefore, to run an algorithm that guarantees an optimal solution to most global transportation planning problems (scenarios) are likely to be intractable.

A Greedy Agent Solution

In addition to the complexity, the decentralised nature of the domain constrains how it can be solved. To overcome these factors, we implement a greedy agent solution. Agents accommodate the decentralised domain, representing independent decision making entities, i.e. the organisations and their business processes, and cooperate with each other, using PAP, to form a transportation plan. Agents that represent organisations that require a transportation task to be achieved are called manager agents (MA), and those that represent transportation organisations (assets) are called transport agents (TA). A greedy PAP approach allows a solution to be found in a reasonable amount of time. In domains where complexity or time constraints are not problematical, the PAP allows a more deliberative approach to be implemented [3].

Provisional Agreement Protocol

PAP is represented in figure 1. PAP has five steps, and control lines are used to indicate the next step in the protocol if a particular speech act[1] is used. In the transportation domain, a task is a transportation task that the MA needs achieved, and a bid is an action or service (single inventory item) that the TA can perform to fully, or partially, achieve MA's task.

Commitment and Persistence Policies

Agents implement a commitment and persistence policy [3, 4]. *The commitment policy states that TA are not committed to their bids unless they are (provisionally) granted.* This allows the TA to bid many conflicting bids concurrently. Therefore, TA do not need their bids rejected by MA in order to use them elsewhere, and thus MA must act quickly in order to secure the bids. MA are only committed to a bid when they send a confirm grant for it. PAP allows backtracking, and thus agents store received bids and

[1] An action (change in the state of the world) as a result of an utterance – e.g. "I promise to pay you", "Drop your weapon or I'll shoot", or "I now pronounce you husband and wife".

tasks for future use. Since bids and tasks may no longer be active at a later time when agents decide to use them, *the persistence policy states that agents (MA and TA) tasks and bids are considered persistent unless they reveal otherwise*. Hence, agents will only discover that tasks and bids are inactive (i.e. cannot be bid for or granted, respectively) *after* they try to *use it* (bid for it, or grant it, respectively). This allows MA to not have to reject bids that they do not intend to use, allowing bids to remain available later if required during backtracking.

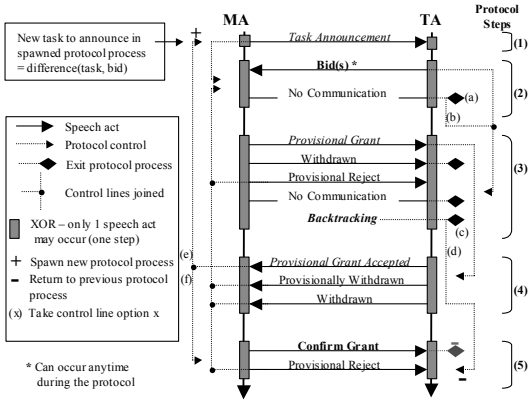


Figure 1. Greedy PAP

PAP steps

1. **(Start – root protocol process)** MA announces the root transportation task(s) to suitable TA. The task contains a deadline *dl*, which is the earliest time that the MA may select a bid for the task, and cost value details (*cvd*), for TA to use in calculating the cost (*cv*) of its bids according to MA's preferences, so that the TA will know which of its possible bids the MA will prefer. Control proceeds to step 2.
   (New protocol spawned from step 4, (e)) If the provisionally granted bid (in step 4) did not achieve the complete task, then the remaining portion of the task becomes the new task to announce (achieve) in a newly spawned protocol process. Control proceeds to step 2.
2. **(From step 1)** If the TA believes that it can achieve, or partially achieve, the task, then it may submit a bid. Only one bid is allowed – TA uses the *cvd* to find its best bid for the task. A bid can be submitted at anytime because the protocol allows backtracking, hence the MA may revisit bids for the task if a granted bid was unsuitable. Therefore, if a new opportunity arises for TA to submit a better bid than the *worst* bid currently submitted, the TA will submit the *updated better bid*. Once a bid is submitted, control proceeds to step 3. If the TA cannot submit a bid, it does not communicate anything, and the protocol process exits for the TA (a). **(From step 3, 4 or 5)** Agents may go back to step 2 if a bid is not currently suitable (provisionally rejected in steps 3 & 5), or no longer (currently) available for granting

((provisionally) withdrawn in step 4). The TA may submit another *updated bid* to replace the unwanted or unavailable bid – the TA submits its *next best bid*. If an updated bid is made, then control proceeds to step 3. If an updated bid is not made, but the TA still has bids submitted that could be considered for the task, then go to step 3 (b). Otherwise, TA exits the protocol process (a).

3. The MA sorts the bids from best to worst, based on the *cv*. The best bid is *provisionally* granted *after* the deadline, then control proceeds to step 4. If the task is no longer available (achieved or part of an infeasible plan), then a withdrawn message is sent, the TA deletes the task and associated bids, and the protocol process exits. If the submitted bid is not currently suitable for the task, say for example, it is similar to another bid that caused backtracking, a provisional reject message is sent, then control proceeds to step 2. The reject is provisional because the bid may be granted again later. Backtracking is required if no bids are submitted by the deadline, and thus no solution exists to achieve the task. If backtracking occurs at the root protocol process, then no plan exists to achieve the task, and thus MA exits the protocol (c). Otherwise, control goes to step 5 of the previous protocol process (d) and give the provisionally granted bid in that protocol process a provisional reject.
4. The TA sends a provisional grant accepted message to accept a provisional grant for its bid. The TA is then committed to the bid, and must pay a penalty if it de-commits. If the bid does not completely achieve the task, then a new protocol process is spawned (e), and the task to announce is the portion of the task that the bid does not achieve. If the bid completely achieves the task, control proceeds to step 5 (f). If the granted bid is no longer available (conflicts with another confirm granted bid), or is no longer available now but may become available later (conflicts with another provisionally granted bid, which may be rejected, freeing TA resources again), then a withdrawn, or provisionally withdrawn, message is sent, respectively. Control proceeds to step 2. MA and TA can implement their own strategy for dealing with provisionally withdrawn bids – when, or if, to grant, or re-submit, the bid again. MA can delete details of bids that it does not intent to use again (e.g. withdrawn bids). TA store details of submitted bids so that it can keep track of which bids it has submitted for a task, preventing the TA from submitting the same bid twice.
5. **(From step 3 in the spawned (next) protocol process)** The granted bid in this protocol process caused backtracking in the spawned protocol process. The MA sends a provisional reject message to reject the bid. The TA is no longer committed to the bid. The MA may grant the bid later, again, if required, in which case the spawned protocol process should not exit. Control proceeds to step 2.

   (From step 4) The provisionally granted bid achieves the complete task, and thus the root task(s). The provisionally granted bid is given a confirm grant. The TA secures the bid (action and inventory item) in its local plan. Both TA and MA are committed to the bid, and thus, if either de-commit, a penalty must be paid. Control proceeds to step 5 of the previous protocol process and a confirm grant is issued for that provisionally granted bid. The process is repeated until the root protocol process is reached, in which case the protocol exits after the confirm grant. A distributed transportation plan for the root task(s) consists of all the confirm granted bids.

In our implemented system, bids that are provisionally rejected or provisionally withdrawn are deleted and not used again by the MA. Therefore, PAP runs like a depth-first search, which ensures convergence (see later).

PAP Planning Process

Figure 2 illustrates the PAP planning process from MA's perspective, represented as a typical planning tree. In figure 2 (a), the MA announces two root transportation tasks, t_1 and t_2 at the root node. $t = \{n, q, est, lft, src, dst, dl, cvd\}$. Say t_2 = {fuel, 110 Kg, 20 hrs, 50 hrs, Adelaide, Cairns, now+5, *cvd*}. TA submit bids b_1, b_2, b_3, b_4, which are root node branches in the tree. $b = \{n, c, tsa, tfa, lsa, lfa, p\}$. Say b_4 = {fuel, 50 Kg, 30 hrs, 38 hrs, Melbourne, Sydney, $1000}.

MA decides to provisionally grant b_1, which has the lowest *cv*, but the bid was provisionally withdrawn (?×). An updated bid for t_1 is not sent. MA then tries to provisionally grant b_3, but the bid is withdrawn (×).

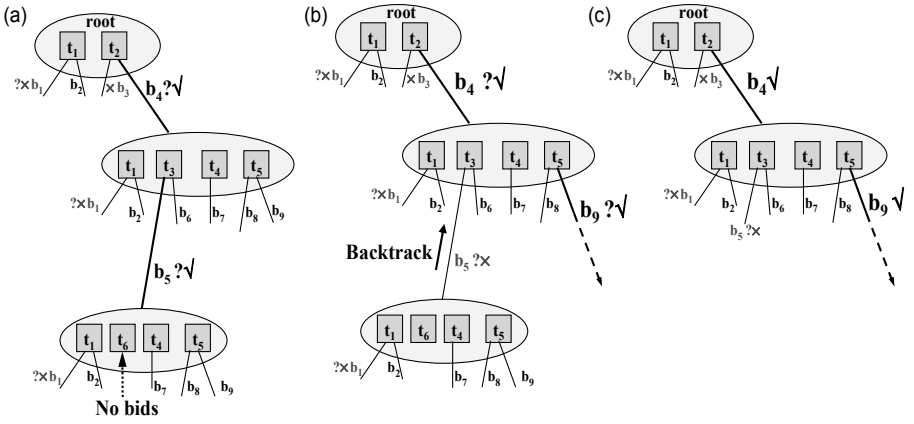


Figure 2. PAP Planning Process

b_4 is provisionally granted (?√). The new task (node) to achieve is the difference between the root node and b_4. Up to three new tasks can be created, to transport the package the remainder of the route (if b_4 is a partial route bid): at the beginning *{n, c, est, tsa, src, lsa, dlnew, cvd$_1$}*; end *{n, c, tfa, lft, lfa, dst, dlnew cvd$_2$}*; and to transport the left over quantity the complete route (if b_4 is a partial quantity bid) *{n, (q-c), est, lft, src, dst, dlnew cvd$_3$}*. In our example, the left over tasks are (beginning route) t_3 = {fuel, 50 Kg, 20 hrs, 30 hrs, Adelaide, Melbourne, now+5, *cvd$_1$*}, (end route) t_4 = {fuel, 50 Kg, 38 hrs, 50 hrs, Sydney, Cairns, now+5, *cvd$_2$*}, and (left over quantity) t_5 = {fuel, 60 Kg, 20 hrs, 50 hrs, Adelaide, Cairns, now+5, *cvd$_3$*}. The new tasks, in addition to all tasks that are not associated with the granted bid (b_4), become the new node. Only new tasks (t_3, t_4 and t_5) are announced, since bids for tasks in the previous node are carried over. This is only allowed if the cost value has certain properties (details beyond the scope of this paper). MA receives bids b_5, b_6, b_7, b_8, b_9.

b_5 is then provisionally granted. The new task created is t_6, which is announced. The new node contains tasks t_1, t_4, t_5 and t_6. After the deadline, no bids are received. Backtracking occurs, b_5 is provisionally rejected (?×), and b_9 is provisionally granted, as shown in figure 2 (b). This process continues until all the tasks in the node are achieved. The transportation plan, which consists of the provisionally granted bids (b_4, b_9, …) are given a confirm grant (√) to secure the plan (individual bids) into the TAs' local plans, as illustrated in figure 2 (c).

Decentralised, Open, and Dynamic Planning

The PAP facilitates decentralised planning. The MA announces a task, allowing TA to submit bids (services) that they are able to, and want to, perform, based on their own self interested goals and private information. TA may enter the system part way through, and join, the PAP planning process, contributing to tasks that are announced from that time. PAP can cope with changing agent capabilities and goals during the planning process. As TA's services change, bids may no longer be available, and thus PAP allows TA to withdraw bids and update them. When new opportunities arise, TA may send a bid that was previously not possible, or a better bid than the bids currently sent. As MA tasks are being solved, or no longer valid due to backtracking or change of goals, MA can withdraw previously announced (invalid) tasks and announce new tasks to be achieved. Therefore, PAP accommodates planning in open and dynamic domains.

Protocol Improvements

With the protocol specified in [3, 4] (PAP_{old}), TA may send more than one bid in response to a task announcement. In the protocol we have presented in this paper (PAP_{new}), to save communication, we limit the number of bids sent to one, which is the best bid that TA has available. If a bid sent in PAP_{old} becomes inactive (e.g. withdrawn or provisionally withdrawn), the TA sends an updated bid to replace it, even if the MA has not used (granted) the inactive bid and the updated bid is worse than it. This is not appropriate because if the MA has not used the inactive bid, then it would not likely use an updated bid that is worse than it. Also, sending an updated bid whenever a submitted bid becomes inactive, which can occur frequently in a dynamic environment, may result in a large amount of communication and memory for agents to send, receive and store the bids. Therefore, in PAP_{new}, an updated (worse) bid is only sent for an inactive bid if the MA attempts to use the inactive bid (i.e. if a bid is (provisionally) withdrawn when provisionally granted), or when a bid is rejected. This also prevents the TA from scanning all its bids whenever an event occurs in its local plan to check if bids had become inactive, which is computationally intensive. The status of a bid is only checked when the bid is provisionally granted.

In [3, 4], we show the planning process as an AND/OR tree, where each task in a node in figure 2 is considered a separate (AND) branch in the tree, and thus a separate sub-tree is formed for each task. In such an approach, selecting bids in a certain order may result in no solution being found, even though one exists, because selecting a bid in one sub-tree for a task may have used a service that was required in another sub-tree for another task. Hence, a solution to the other task cannot be found, resulting in

backtracking on all the tasks, and their sub-trees, at that node. The approach taken in figure 3 eliminates this problem because all the tasks are considered as a single node, where only one bid can be provisionally granted for the node – for one task, the remaining tasks are carried over to the next child-node. The tree now resembles one used in typical search, where there are only nodes and OR branches. Therefore, all combinations of bids (paths) can be considered for all the tasks in a node, and thus the order in which bids are selected does not matter.

Transport Agent Bidding

Bid prices depend on actions adjacent to it in the local plan – i.e. on *deviation* prices. Deviation is movement for the TA to get from the previous action to the bid, and from the bid to the following action. If actions can be rejected, the TA does not know which actions will end up adjacent to the bid, making pricing difficult. Hence, the bid price is calculated using mathematical expectation, with price as the value function and using the probability to remain for each action, or inventory item, in the local plan. Details are beyond the scope of this paper.

Partial bids are problematic as *any* bid that meets the task constraints, can be potentially part of a plan to achieve a transportation task. Figure 3 illustrates TA's local plan. Bids can be made either in conjunction with existing actions, which is called a *piggyback bid*, or by creating a new action between gaps in existing actions, which is called a *gap bid*. The local plan is checked for bids within the time window specified in the task. Piggyback bids are easier to check as the route (lsa_j to lfa_j) and time interval (tsa_j to tfa_j) are fixed. Gap bids are computationally harder since the TA can consider many routes, and a range of times within the gap time interval.

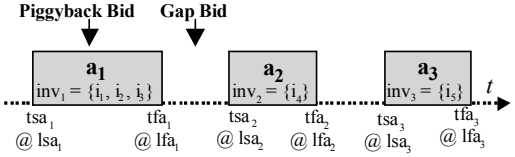


Figure 3. TA's local plan.

The TA always bids for the largest capacity that it can carry, so that TA are not traveling with less than full capacity and so wasting resources. Gap bids are scheduled as early as possible, as delivering packages earlier is better. A rational approach to selecting routes is to give preference to those that achieve as much of the transportation task (route) as possible. Therefore, for the task to move fuel from Adelaide to Cairns, TA should select the route Melbourne to Sydney over Melbourne to London, which actually moves the MA away from its task. TA has a route threshold *rth* where if the route does not achieve a certain portion of the task, then it will not be considered, saving computational time. The route that achieves the greatest amount of the task may not be the best route for *a particular gap*, as a large *deviation* may be required. Routes that minimise the deviation in the gap will likely save time and price, and thus result in a better bid.

In the bidding process for each task, routes are sorted in a list *ls* in order of the quantity that the route achieves the task, removing routes below *rth*. Only piggyback bids with these routes are considered. In each gap checked, routes are sorted in a list *lg* in order of deviation required, and *ls* and *lg* are combined, to form *lc*, where route at the top of *lc* achieves a large portion of the task and requires little deviation in the gap. The top *N* routes in *lc* are used to calculate bids in the gap. All bids formed are sorted in a list, based on the cv. The best bid that has not been previously sent is submitted to the MA.

Formal Evaluation

Convergence

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

(1) *Rejecting or withdrawing of bids:* Two conditions that ensure an infinite loop does not occur between step 2 and steps 3, 4, and 5 are:

(a) Only a finite number of bids should be sent at step 2, thus only a finite number of new bids can be rejected or withdrawn. The TA in our prototype do not send bids with the same, or *similar*, route and time to a previously sent bid, preventing bids with infinitesimal differences in time. The number of routes is assumed to be finite. The task specifies a time window, constraining the time range possible for bids. Capacity in bids are defined as {p, w}, p sub-packages of weight w, and thus the number of bids with different capacity is finite. Therefore, the number of bids that TA can submit is finite.

(b) MA should grant a finite number of bids a finite number of times, otherwise, the MA may indefinitely persist in granting the same set of bids, which will be consequently rejected or withdrawn. In our prototype, granted bids that are rejected or withdrawn are deleted, and thus bids to grant for a task will eventually run out.

(2) *Provisional grant:* It may be possible that bids are granted indefinitely at step 4, never completely achieving the task, resulting in an infinite sequence of protocol processes. This is more likely if bids that move the MA away from its task are selected. Convergence can be ensured: **(a)** If the route threshold *rth* is set such that only routes that move the MA closer to its task (not infinitesimally closer) are selected, hence, eventually the task will be achieved. **(b)** If there are a finite number of bids to grant for the task. (b) is satisfied because the task has a time window and there are finite number of TA. Granting bids will therefore result in the TA's local plan, within

the time window, to occupy with actions until actions can no longer be added, and thus bids can no longer be made.

(3) *Backtracking:* PAP may alternate between protocol processes indefinitely – if the MA continually backtracks, and then selects the same bid in the previous protocol process that caused backtracking. Bids that cause backtracking (are rejected) are deleted, and therefore this will not occur.

Conditions (1), (2) and (3) are satisfied. Thus, the PAP implementation for our transportation domain converges.

Time Complexity

For TA to compute bids, when there are *na* actions in their local plan, *na* actions can be checked for piggyback bids, (*na*+1) bids can be checked for gap bids, which can check *nr* routes. Therefore, the total number of bids that can be made is $tnb = na + (na + 1) \cdot nr$. For each bid, the price needs to be calculated. In the worst case, the deviation price between the bid and each action (*na*), and between each action $(na/2) \cdot (na/2)$, must be considered, therefore $ccp = na + (na^2/4)$. *nr* routes are scanned to determine lists *ls* (once), and *lg* and *lc* (*na*+*1* times), so $clr = nr + (na + 1) \cdot 2 \cdot nr$. The time complexity is therefore $O(tnb \cdot ccp + clr)$, which is of order $O(na^3 \cdot nr)$.

The time complexity for the TA to check for bids does not affect the time for the MA to form a transportation plan, because the MA specifies a deadline for which the bids must be submitted, and the planning to progress. If the MA receives a bid after the deadline, then it may not be considered unless there is backtracking, and thus may affect the quality of the resulting plan. The TA should check as many bids as possible within the deadline. A shorter deadline may mean that the TA must place greater restrictions on the search for bids (e.g. on *rth* and *N*) in order to find a suitable bid within the deadline, which may reduce the quality of the bid submitted, and hence the final plan produced. A longer deadline will provide the TA with greater computation time for bids, and thus a potentially better plan, but will increase the time taken to find a transportation plan.

Our PAP implementation performs a depth-first search. The worse case time complexity for a depth-first search is exponential. If *br* is the branching factor, and *m* is the depth of the tree, the time complexity is $O(br^m)$ [8]. In our domain, most scenarios are likely to have many solutions – many combinations of routes, times, capacities, prices, and TA that can perform the transportation. In these cases, the MA is not likely to get anywhere near the worst case time complexity [8]. Reducing services (number of TA and their capabilities), or restricting task constraints, will likely increase backtracking, and thus time to find a plan.

Δt_d is the duration for the deadline to pass so that the MA can progress with planning. The MA waits Δt_d when a new node is created (for bids to be received), or during backtracking (for the TA with the rejected bid to send an update). For each branch that requires backtracking, the MA must wait twice the duration Δt_d. Therefore, the worst case planning time is approximately $2 \cdot \Delta t_d \cdot \sum_{i \in \{1..m\}} br^i$.

Memory

As with a depth-first search, the MA must store a single path from the root node to the leaf node, along with the remaining unexpanded branches. The maximum number of branches to store is br·m [8]. The maximum number of nodes to store is *(m+1)*, as nodes are only created when branches are selected. Each node may contain more than one (sub-)task. A node at depth *d+1* may contain, at maximum, two more tasks than the node at depth *d* – the bid for a task may be partial in quantity and route, creating three new tasks, and the task associated with the bid is removed. Therefore, the maximum number of tasks that may need to be stored for the nodes, where *nrt* is the number of root tasks in the root node, is $(\mathrm{m}+1)\cdot \mathrm{nrt} + 2\cdot \sum_{\mathrm{i}\in\{1\ldots\mathrm{m}\}} \mathrm{i}$.

Therefore, the memory requirements to store MA's complete planning tree increases as we increase: the number of bids the MA receives; the number of bids required to achieve the root tasks (depth *m*); the number of root tasks; the number of partial bids that result in three new tasks to be created at each node. The number of bids received is restricted. The MA receives one bid from each TA for a task, unless the bid is rejected/withdrawn, in which case the bid is deleted and another bid *might be* submitted. The TA may *occasionally* submit (extra) updated better bids. Thus, the number of bids (branches) at any time for each task is approximately equal to the number of TA that submit bids for the task.

The maximum memory requirement for TA is $mnt\cdot\rho$, where *mnt* is the maximum number of tasks that the TA considers at any one time, and ρ is the maximum number of bids sent for each task. The TA memory requirements grow as tasks are received, and bids are sent. As tasks are withdrawn, or expired (become very old), the tasks and associated bids can be deleted, restoring memory. If there is a constant flow of tasks over time, replacing deleted tasks, the memory requirements should remain relatively static. If not, the TA may set aside a fixed memory limit, deleting the oldest tasks and bids when the limit is reached.

Communication

MA only announce new tasks, hence the maximum number of tasks announced is $\gamma < nrt + 2\cdot \sum_{i\in\{1..m\}} br^i$. The maximum number of tasks sent by MA is $\tau_{MA} = mnta\cdot\gamma$, where *mnta* is the maximum number of TA, the number of tasks received by TA is $\tau_{TA} = \gamma$, the maximum number of bids sent by TA is $\beta_{TA} = \gamma\cdot\rho$, and bids received by MA is $\beta_{MA} = \gamma\cdot\rho\cdot mnta$. A maximum of three speech acts can be communicated for each branch, hence, the maximum number of speech acts communicated for MA is $\delta_{MA} = 3\cdot \sum_{i\in\{1..m\}} br^i$, and TA is $\delta_{TA} = 3\cdot \frac{\sum_{i\in\{1..m\}} br^i}{mnta}$.

As mentioned, for most scenarios, the worst case exhaustive search is unlikely. Therefore, the actual communication will likely be much less than described above. Not all branches in the planning tree will be used (granted), and therefore, due to the

persistence policy, do not require any speech acts to be communicated to inform the TA that the branches are not being used. The TA only send one bid at a time, of their maximum of ρ bids, for a task. Only if a branch (bid) is used, and then rejected or withdrawn, does the TA send another bid. Therefore, it is unlikely that the ρ bids will be communicated. The MA may use domain knowledge to send the task only to the most suitable candidates. Not all the TA that receive the task may submit bids.

Implementation

Over 30 scenarios were executed, which were taken from military logistics training exercises [9]. We manually determined a plan for each scenario, which is how military planners form schedules in training exercises, and then compared the actual solution produced to that plan. Due to the complexity, the optimal plan could not be computed.

Scenarios comprise up to two MA, ten heterogeneous TA, and an agent to provide MA and TA with distances between locations. TA could travel up to 130 routes, and the transportation tasks were to transport one or two large quantities of resources a large distance. Only ten TA were run at once because of the limited computing resources available – one PIII 900MHz to run all agents and communication software. The agents were developed using the ATTITUDE multi-agent architecture [10], and the CoABS grid [11] was used to allow agents to communicate.

Since the implementation of the PAP is greedy, the quality and time taken to arrive at a solution relies heavily on the *rationality* of the *cv*, in order for the MA to make rational choices for the selection of bids. Deliberative approaches, e.g. A*, require an underestimate for *cv* as many paths are tested to provide optimality. Greedy approaches rely on rational decisions as choices cannot usually be altered – backtracking is not allowed if a *bad solution* is pursued, only if an infeasible solution is pursued. Choosing suitable bids will also minimise infeasible solutions, and hence, backtracking and time.

The *cv* takes into consideration the current cost of the plan, and the expected cost to achieve the remaining tasks. If the expected cost returns a low value, then the resulting plan may have the TA performing a route in incremental steps rather than one complete trip, which is less efficient. This is due to the MA selecting partial routes, thinking that a faster and cheaper option is available for the remainder route. When a faster and cheaper option is not found, the TA will again submit another partial bids to perform another step of the route, etc. Making the expected cost conservative will force the MA to select large complete routes over many partial routes. This is a rational approach, because in most circumstances, one large trip is better than many small trips.

In the worst case scenario, there was a single MA with two *large* transportation tasks, requiring a depth in the planning tree of over 60, and four TA, and hence limited available services resulting in considerable backtracking. The deadline duration was set to 2 minutes. The duration was extended to ensure that TA had *more* than enough time to compute bids, with our limited computing resources and relaxed bidding restrictions (*rth* and *N*), before the MA progressed with the planning. A plan was produced in approximately 4 hours. Ideally, each TA will have its own processor and greater bidding restrictions, allowing it to compute bids quickly, and with the deadline duration set to say 5 seconds, providing a plan within approximately 10 minutes. Manually calculating

a detailed complete plan for this scenario took us much longer than 10 minutes, while having all the information regarding TA capabilities available. In a reality, this may not be the case. Therefore, these capabilities may need to be extracted from these TA (transport organisations) by the logistics planners via some means, for example, telephone, Internet, etc. This will likely add to the time required to find a transportation schedule, as well as add to the complexity for a planner to manually find a transportation plan.

In the worst case scenario, the MA stored a maximum of 135 tasks and 514 bids, sent a total of 460 tasks to TA (each task sent 4 times, one for each TA), received a total of 479 bids, and communicated (sent and received) 331 speech acts (grants, rejects and withdrawn messages). The MA backtracked 18 times. The worst case TA stored 115 tasks and 113 bids, received 115 tasks and sent 179 bids (the other TA sent approximately the same number of bids as the number of received tasks) and communicated 163 speech acts. Such communication and memory requirements was easily manageable in our prototype.

The transportation plans produced were similar, or the same, to the plans we expected to be produced. There were a few minor differences. The TA occasionally took an unexpected route, primarily because the TA had a gap in its local plan such that it could perform a cheap partial route bid, where the route was a slight deviation from the direct route. In some plans, the TA may take two or more trips to transport packages, rather than one complete trip. As mentioned, this can be reduced by increasing the expected cost in the *cv*.

Scenarios with heterogeneous TA with wide ranging capabilities (speed and price) resulted in the worst plans as it was difficult to accurately determine the expected cost (speed and time) for the remaining tasks in the *cv* function. In such circumstances, the *cv* associated with the bids may not be accurate, resulting in the MA not selecting the most appropriate bid. Means of overcoming this problem is under investigation.

With the scenarios that were executed, our agent-based global transportation scheduling implementation compared well with manual approaches, in terms of time and quality [9]. Even if a scenario was found in which the agent-based implementation did not perform better than doing it manually, planners can still benefit by automating this complex and tedious scheduling process – particularly having agents to do the running around for the planner in order to extract (and assemble) transport capabilities from various transportation organisations.

Experiments were performed using two MA. If a transportation task is split between two MA, then the resulting plan is considerably worse than if one MA achieves the task. Two MA will compete for services in order to achieve their tasks, resulting in the inefficient use of services. These experiments also showed that the PAP could cope in a dynamic environment, where services for a MA were continually changing because other MA were also using them. The occurrences of withdrawn bids increased because submitted bids were being used by other MA by the time a MA tried to grant it. A shorter deadline duration worked better as it allowed a MA to secure received bids before another MA could use it.

Future experimental work includes: further experiments with other military and commercial scenarios; determining the effects of varying *rth* and *N* on the TA and

overall solution; investigating the scalability of our implementation; and comparing our approach with related transportation scheduling data sets, if available.

Related Work

Fischer's ECNP [5] was developed for transportation scheduling where TA only submit full route bids. Using full route bids in PAP provides the same solution as ECNP. In this case, PAP requires less communication (unused bids do not require rejection), where communication saved is equal to the unused bids $(br \cdot m - m)$. Memory requirements with PAP are greater than ECNP. The MA must keep all bids at each node, and thus an extra $(br - 1) \cdot (m + 1)$ bids. The TA keep all tasks received, and bids sent for these tasks, and thus an extra m tasks and m bids. PAP is preferred if communication costs are high and memory costs are low. TA memory problems can be overcome by deleting tasks and bids that are not granted after some extended time period, or go over a set memory limit.

Fischer uses Simulated Trading with ECNP to improve the solution obtained by ECNP. In Simulated Trading, TA exchange allocated transportation tasks with other TA if the exchange is beneficial (saves money). In this situation, TA are benevolent, and therefore the saving are passed on to the MA. In our domain, TA may not be benevolent, and therefore they may keep the savings, with no benefit to the MA. We require a protocol that focuses on MA-TA interaction, for MA to allocate tasks in the best way possible to TA, rather than TA-TA interaction.

Using only full bids (full quantity and route) reduces PAP to the Contract Net Protocol [12]. Communication for $(br > 2)$ would be less since bids do not need to be rejected, but the selected bid would require both a provisional grant and a confirm grant, hence the saving is $(br - 2)$. Since backtracking is not required, TA can remove tasks and bids sent that are not granted soon after the deadline, indicating the bid was not required.

Our transportation prototype can also solve the grey-hound transportation problem [6, 7] by fixing the actions in TAs' local plans. The time complexity for TA bidding is reduced to O(na), as only piggyback bids are checked for.

Sandholm [13, 14] allows agents to break contracts and take alternative options (backtrack), but pay a penalty in order to do so. With PAP, MA do not fully commit to bids, and thus can backtrack without having to break contracts, making it easier for MA to test various options to find a good solution. This occurs in reality, where one may place a tentative hold on services. Organisations that allow this flexibility are likely to be preferred over those that don't.

In coalition formation [5, 15-17], tasks are distributed among agents in order to increase their utility. The planning problem, decomposing a task into suitable subtasks that can be to allocate to agents, must still be solved. The primary purpose of PAP is to address the planning problem. Coalition formation also focuses on TA-TA interaction (exchanging tasks) rather than MA-TA interaction (allocating tasks).

Multi-unit single-item, single-unit combinatory, and multi-unit combinatory, auctions [18-23] solve a similar problem to PAP. An agent announces a collection of items for sale (a task). The agent receives bids from other agents that request a portion of the items (partial bid). Once bids are received, the agent plans, assembling a set of

bids that maximises its profit. This approach is not well suited to our domain. The TA must submit all possible bids to the MA *before* it starts planning. The number of bids sent for each TA can be very large, and TA may not be willing to release all this information regarding its capabilities. Bids could have dependencies (selecting one causes another to become unavailable) that may be difficult to define in the bids. Due to the domain's dynamic nature, bids sent may become unavailable, and new bids may become available, during the planning process. Dang and Jennings [18] propose a greedy approach to multi-unit combinatorial auctions. It is similar to ECNP, where the best partial bid for a quantity of items is selected, the remaining quantity becomes the new goal to be achieved, and the process repeats. As with ECNP, it does not permit backtracking of single bids.

Conclusion

PAP facilitates distributed agent planning in decentralised, open and dynamic environments. A greedy PAP approach was presented, with improvements, and applied to the global transportation planning domain, allowing partial quantity and route bids, and backtracking if an infeasible plan arises. Formal evaluation and implementation details of the protocol, and its transportation implementation, were presented. Our implementation has greater flexibility than previous transportation scheduling approaches as it can address Fischer's full route (ECNP), and the grey-hound, transportation problems, as well as our own global transportation scheduling problem. Further experimental evaluation is required.

We would like to acknowledge Steven Wark, Chris Nowak and Andrew Zschorn, for their assistance.

References

[1] Perugini, D., et al. Agents for Military Logistic Planning. in Workshop "Agent Technology in Logistics" as part of the 15th European Conference on Artificial Intelligence (ECAI-2002). 2002. Lyon, France.

[2] Perugini, D., et al. Agents in Logistics - Experiences with the Coalition Agents Experiment Project. in Proceedings of the International Workshop "Agents at work", held in conjunction with the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03). 2003. Melbourne, Australia.

[3] Perugini, D., et al. Distributed Information Fusion Agents. in Proceedings of the 6th International Conference on Information Fusion. 2003. Cairns, Australia.

[4] Perugini, D., et al. A Distributed Agent Approach to Global Transportation Scheduling. in IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003). 2003. Halifax, Canada.

[5] Fischer, K., J.P. Muller, and M. Pischel, Cooperative transportation scheduling: an application domain for DAI. Applied Artificial Intelligence, 1996. **10**(1): p. 1-33.

[6] Dean, T. and L. Greenwald, Package Routing in Transportation Networks with Fixed Vehicle Schedules: Formulation, Complexity Results and Approximation Algorithms: Technical Report CS-92-26. 1992, Department of Computer Science, Brown University.

[7] Dean, T. and L. Greenwald, A Formal Description of the Transportation Problem: Technical Report CS-92-14. 1992, Department of Computer Science, Brown University.
[8] Russell, S. and P. Norvig, Artificial Intelligence: A Modern Approach. 1995: Prentice-Hall International.
[9] Perugini, D., et al. Agent-Based Transport Scheduling in Military Logistics. in Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04). 2004. New York, U.S.A.
[10] Lambert, D. Automating Cognitive Routines. in Proceedings of the 6th International Conference on Information Fusion. 2003. Cairns, Australia.
[11] Kettler, B.P.e., The CoABS Grid: Technical Vision, *http://coabs.globalinfotek.com/*.
[12] Smith, R.G., The contract net protocol: high level communication and control in a distributed problem solver. IEEE Transactions on Computers, 1980. **C-29**(12): p. 1104-13.
[13] Sandholm, T. and V. Lesser, Leveled Commitment Contracting: A Backtracking Instrument for Multiagent Systems. AI Magazine, 2002. **23**(3): p. 89-100.
[14] Sandholm, T.W. and V.R. Lesser, Advantages of a leveled commitment contracting protocol, in Proceedings of the Thirteenth National Conference on Artificial Intelligence 1996, pp.126-33 vol.1.
[15] Kraus, S., O. Shehory, and G. Taase. Coalition Formation with Uncertain Heterogeneous Information. in Proceedings of The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems. 2003. Melbourne, Australia.
[16] Sandholm, T.W. and V.R. Lesser, Coalitions among computationally bounded agents. Artificial Intelligence, 1997. **94**(1-2): p. 99-137.
[17] Rosenschein, J.S. and G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. 1994: The MIT Press. 231.
[18] Dang, V.D. and N.R. Jennings. Polynomial algorithms for clearing multi-unit single item and multi-unit combinatorial reverse auctions. in Proc. 15th European Conf. on AI (ECAI-2002). 2002. Lyon, France.
[19] Walsh, W.E., M.P. Wellman, and F. Ygge. Combinatorial Auctions for Supply Chain Formation. in Second ACM Conference on Electronic Commerce. 2000.
[20] Gonen, R. and D. Lehmann, Optimal Solutions for Multi-Unit Combinatory Auctions: Branch and Bound Heuristics. EC, 2000: p. 13-20.
[21] Leyton-Brown, K., Y. Shoham, and M. Tennenholtz, An Algorithm for Multi-Unit Combinatory Auctions. AAAI, 2000: p. 56-61.
[22] Nisan, N., Bidding and Allocation in Combinatory Auctions. EC, 2000: p. 1-12.
[23] Sandholm, T. and S. Suri. Market Clearability. in IJCAI. 2001.

Don Perugini[1,2], Dale Lambert[1], Leon Sterling[2], Adrian Pearce[2]

[1] Defence, Science and Technology Organisation
Edinburgh, S.A., 5111
Australia

[2] Department of Computer Science and Software Engineering
The University Of Melbourne, Carlton, Victoria, 3010
Australia

{don.perugini, dale.lambert}@dsto.defence.gov.au; {leon, pearce}@cs.mu.oz.au

An Adaptive Approach to Dynamic Transport Optimization

Klaus Dorer and Monique Calisti

Abstract. In this paper, we present the agent-based approach we have developed to solve dynamic multi-vehicle pickup and delivery problems with soft time windows. While many of the existing research frameworks have been focusing on reaching near-optimal solutions, the central theme of our work is the optimization of real-world sized problems in near real time. In order to describe our approach and analyse its performance, we introduce a real case scenario in which a logistics company must dynamically optimize a set of transportation requests. The aim is to show how our adaptive solution produces significantly better results than can be achieved with manual optimization of professional dispatchers.

1. Introduction

Today, like for several industries and markets, the logistics sector faces the extensive and fundamental challenges associated with globalization. With shrinking margins, and in many cases being barely able to cope with huge cost pressure, companies are having to substantially revise their product and service offerings, business processes, and operational systems. In the transport logistics market, in particular, the problem is usually faced by (1) distributing the transportation business across a number of regional dispatching centers and increasing the number of dispatchers, and (2) deploying computer-based solutions for strategic network planning and short-term route optimization planning.

However, business distribution is complicated by the fact that a considerable amount of communication and remote coordination is necessary to detect opportunities of combining transportation between regional dispatching centers. More precisely, delays and coordination costs can heavily impact the ability of optimizing resource consumption (i.e., deployed vehicles, number of dispatching centers) and finally fulfil all orders as expected. As a matter of fact, for instance, the average utilization of vehicles in charter business is fairly low (55% in the example

described in section 4). Furthermore, while a number of existing computer-based solutions allow the automatic creation of the dispatching plans, they usually do not, or only marginally, support the case of plan deviations resulting from unexpected, or previously unknown events.

In this perspective, we argue and show that intelligent transport optimization systems able to dynamically adapt transport plans and scheduling according to possible deviations and unforeseen events have the great potential of reducing overall transport costs, by improving the coordination process of distributed dispatchers and improve resource consumption (i.e., better usage of available resources than allocation of additional ones). The problem of organizing the pickup and delivery of transport requests in a timely fashion by deploying multiple vehicles is known in the Operations Research literature as dynamic m-PDPSTW. We give a short introduction to the dynamic m-PDPSTW problem in Section 2. More extensive overviews of the problem can be found, for instance, in [14, 11, 4, 1]. Several agent-based approaches have been proposed to deal with this kind of dynamic optimization problem, e.g., [3, 9]. The main motivation is derived from the fact that multi-agent systems ideally reflect the distributed nature of the problem and are able to deal with the dynamics of planning and execution in near real-time settings. The work described in this paper introduces an agent-based optimization system that focuses on real-world transportation problems which often differ from theoretical problem formulations, as expressed in most of the research literature, in terms of the number of constraints to be observed and in the size of the problem itself [16]. As of today, we are aware of only a few research publications that deal with such large problem instances, e.g., [7, 6, 5]). Furthermore, the dynamicity of the general attribute values (e.g., orders to be delivered, availability of trucks, etc.) requires the capability to dynamically adapt previously computed transport plans. The main contribution of our work is in the effective combination (as proven by the results obtained with real data and discussed later in the paper), of classical optimization techniques with software agents (problem solvers) that dynamically coordinate in order to adapt solutions (transport plans) according to variations occurring in the problem attributes.

The remainder of the paper is organized as follows: Section 2 covers the problem formulation, the constraints that must be taken into account in order to obtain delivery routes that are navigable in the real-world and the cost model used for empirical evaluations. Section 3 describes the optimization algorithms and agent-based design that enables us to deal with problem instances consisting of thousands of transportation requests. Section 4 gives and discusses the empirical results obtained by solving a real world optimization problem. Finally, in Section 5 we discuss future work directions before concluding the paper.

2. The Dynamic Multi-Vehicle Pickup and Delivery Problem

The dynamic multi-vehicle pickup and delivery problem with soft time windows (dynamic m-PDPSTW) [13, 14] consists of finding optimal routes for serving transportation requests of (a usually large) set of customers. The problem is called *dynamic*, as opposed to its static version, if the transportation requests are not all known in advance. This means that new requests can be received and treated during the optimization process itself. An even more dynamic version of this type of problems deals with changes that can occur to the transportation requests (e.g., changes in the size of orders) and/or to the transportation capacity (e.g., some trucks can become temporarily unavailable), in real-time. These changes reflect reactions to information drivers provide when picking up specific orders, or because of unforeseen situations such as accidents and traffic jams.

The problem is defined as 'single-vehicle' when individual transportation requests are served by a unique vehicle. In our framework, we deal with the 'multi-vehicle' problem which implies multiple vehicles can be used for each delivery. The vehicles may be of different type and have different capacities. As opposed to vehicle routing [10, 14] in *pickup and delivery problems* (PDP), vehicles do not necessarily start or end in the same location. Transportation requests may be characterized by the same *pickup* and *delivery locations*, although they are usually characterized by different ones. The pickup and delivery of orders has to occur within a specific *time window*, even though time constraints can potentially be violated within some tolerated degree. These kind of problems are called PDP with soft time windows.

The *solution* of an m-PDPSTW consists of a set of routes including a schedule that specifies the times at which the vehicles must be at selected locations. The result is also called a *delivery plan* and its quality or goodness is given by an *objective function*, which in our case is a cost-based function (see Section 2.3 for more details). In our framework, the term *node* is used to indicate the combination of both location and time (arrival and departure time) for a given vehicle at that specific location. A *leg* is the path between two nodes. A *tour* is a sequence of nodes a vehicle visits. The vehicle is assumed to be empty at the beginning and at the end of a tour, but not while the tour is incomplete (occasionally in the literature, tours are also called active periods or mini-clusters [11]). A *route* is a sequence of $n \geq 1$ tours travelled by a vehicle. The terms *order* and *transportation request* are used synonymously to indicate a customer's request to transport some goods from a pickup location (within a specified pickup time window) to a delivery location (within a specified delivery time window). More details on the transport request attributes are given in the following section.

2.1. Characterization of Transportation Requests

As previously mentioned, for dynamic forms of the m-PDPSTW problem, transportation requests are usually not all available when starting the route planning and delivery scheduling process, but can arrive subsequently. Therefore, some

transport requests may have already been served when new ones arrive. Every transport request is characterized by the following parameters:

- Order type.
- Volume (in loading meters).
- Weight.
- Pickup location.
- Pickup time window.
- Delivery location.
- Delivery time window.
- Time at which the order is known to the system.

Additional information is also available for each vehicle:

- Vehicle type.
- Capacity (volume, in loading meters).
- Capacity (weight).
- Start location.
- Availability time (at start location).
- Tariff.

The tariff indicates a cost class to which the vehicle belongs (see Section 2.3). A mapping function defines which order types fit to which vehicle type.

2.2. Problem Constraints

The optimization algorithm has to obey a number of constraints considered during the calculation of routes. Constraints are classified as *hard constraints* and *soft constraints*. Hard constraints express conditions that must hold. Soft constraints express conditions that may be violated to some degree. The former category includes:

- Load constraints:
  - Precedence (pickup has to be before delivery);
  - Pairing (pickup and delivery have to be done by the same vehicle);
  - Capacity limitation of a vehicle;
  - Weight limitation of a vehicle;
  - Order type and vehicle type compatibility.
- Time constraints:
  - Earliest pickup - maximum allowed early time;
  - Latest pickup + maximum allowed delay;
  - Earliest delivery - maximum allowed early time;
  - Latest delivery + maximum allowed delay;
  - Maximum time a vehicle may wait at a location for new transport requests;
  - Maximum duration of a tour;
  - Maximum duration of a route;
  - Lead time for ordering a vehicle;
  - Maximum driving time of a driver.

- Maximum number of locations on a tour.

Soft constraint violations produce violation costs that impact the overall optimization process and can then be accounted for in the final overall solution costs. Violation costs are introduced so that constraints are only violated if it is 'worthwhile', i.e. only if the cost savings achieved by violating a soft constraint exceed the violation costs such constraint violation implies. A soft constraint is defined in terms of a start value above (or below) which the condition is soft-violated, an end value above (or below) which the constraint has a hard violation.

Fixed violation costs are incurred if any constraint is "softly" violated, *variable violation costs* grow proportionally to the amplitude of the soft violations. In this perspective, the fixed violation costs can be used to control the number of soft constraint violations, while the variable violation costs enforces the amplitude of constraint violation to be kept low (see Section 2.3). The following soft constraints have been considered in the real world settings we consider:

- Earliest pickup time;
- Latest pickup time;
- Earliest delivery time;
- Latest delivery time.

2.3. The Cost Model

The major concern of logistics companies is to reduce their costs [2]. Therefore, the objective function used to evaluate the optimization results is cost-based. In our framework, the cost model was defined in order to properly take into account real-world costs and constraints and thereby enable comparison between the optimization results of our agent-based m-PDPSTW approach with real transport plans created manually by professional dispatchers. The cost model distinguishes between three kinds of costs: variable, fixed and violation costs (which can again be decomposed into fixed and variable as explained in Section 2.2).

Variable costs are assigned according to the length of a route, the amount of transported load and the start location of the specified route. The variable cost of a route is calculated as:

$$c_{var} = \sum_{tours} c_{km} * dc_{route} \tag{1}$$

with c_{km} the distance cost computed as:

$$c_{km} = d_{tour} * l_{max} * tf(region, d_{tour}, l_{max}). \tag{2}$$

where d_{tour} is the driven distance of the tour, l_{max} is the maximal load (volume) that is transported on a leg of the tour, and tf is the cost class (tariff) which the tour belongs to. In the real case we analyzed, 7 tariff regions within Germany, 3 distance classes for each region and 16 load classes for each region and distance class have been defined. The values we used for tf have been derived from real data.

dc_{route} is a discount that is granted to special routes. A discount is granted if a route contains multiple tours (tramp tours). This reflects the fact that in the

charter business cheaper offers are given when multiple consecutive tours can be offered to a subcontractor. The empty driven kilometers between two tours must be limited (e.g., 70 km), and the period of time between the end and start of two tours must be restricted (e.g., 4 h). A higher discount is granted if a route with multiple tours terminates close to (< 100 km) the start location (back tours). This reduces the workload of the subcontractor in terms of looking for freight (and drivers) for a back tour of the vehicle.

Fixed costs may be assigned to a vehicle on a daily basis. This means that each day a vehicle is used fixed costs are assigned. In the analyzed case, fix costs have been replaced by minimum costs. If the vehicle's variable costs are below a minimum threshold, the final costs of the route correspond to the minimum costs independent of the trip duration and usage of the vehicle. The total cost of a route is then calculated as:

$$c = \max(c_{min}, c_{var}) \tag{3}$$

where c_{min} is the configured minimum costs of a vehicle.

As explained in Section 2.2, violation costs, arising when soft constraints are violated, are used by the optimizer during the solving process[1]. Each soft constraint violation causes a fixed violation cost independent of the amplitude of the violation. If, for example, the fixed violation cost is equivalent to 100 cu[2], the cost savings the optimizer must achieve by allowing a soft constraint violation has to be larger than 100 cu. Violating a constraint for less than the corresponding cost savings is not allowed. In this way the number of constraint violations is kept under control.

A soft constraint violation also generates a variable violation cost, which proportionally increases with the amplitude of the violation. If, for example, the variable violation cost is set to 50 cu per hour then the cost savings for a 3 hour delay must be at least 150 cu. In this way the average amplitude of violations can be controlled.

3. Agent-Based Optimization

The optimization process incrementally reflects the dynamics of the underlying m-PDPSTW settings. Whenever a new transport request is made available to the system, the current delivery plan is updated. This is done in a two-phase approach: First, a new valid solution is generated including the new transportation request. Then, the obtained solution is improved by cyclic transfers of transportation requests. The next two sections explain these two steps and Section 3.3 describes how these algorithms can be distributed across multiple agents.

[1] For final costs estimation, violation costs were not added for specific real world settings we considered, in order to be able to compare the agent-based optimization results with plans defined manually by professional dispatchers.

[2] cu stands for currency unit

3.1. Solution Generation

The first step taken when receiving a new order is the generation of a new valid solution. The algorithm used for this is a sequential insertion of transportation requests [7]. All available vehicles are checked to see if they are able to transport the order and what additional costs are incurred. One of the available vehicles is a new empty charter vehicle assumed to be available at the pickup location. The process of sequential insertion is summarized in Algorithm 1. Each time a

Algorithm 1: The order insertion algorithm.

```
For all vehicles
  Check availability
  For all pickup insertion points
    For all delivery insertion points
      Check constraints and schedule
      Calculate quote
      If cheapest quote then store quote
Assign order to cheapest quote
```

transportation request is added to a route either 0, 1 or 2 new nodes must be added depending on whether the pickup and/or the delivery nodes of such request are already part of the route or not. In this way, multiple pickups and deliveries per location are possible. Based on the insertion algorithm, it is also possible that a vehicle visits the same location several times. Finally, for all combinations of existing nodes a quote is generated for inserting a new pickup and delivery node.

Each time a new node is added to a route the order of nodes can be optimized. This corresponds to solving a traveling salesman problem (TSP) and is intractable in the general case [8]. In our approach, we do not solve the general TSP. Instead, the order of existing nodes is never changed and for every new node all points between existing nodes are checked for possible insertion of the new one. In a concrete case where 200 orders were considered, the experimental results obtained by comparing the performance of an optimal TSP solving algorithm and the outcomes of our approximation algorithm have shown that only twice did our algorithm produce sub-optimal routes with just 17 additional kilometers ($< 0.2‰$ of the complete distance traveled). For real world sized problems, the gap between the optimal solution and a slightly sub-optimal one can be considered negligible when compared to the gain in terms of runtime performance.

When no vehicle can transport an incoming new order, which means all vehicles would have to break hard constraints, the orders service level is lowered. This implies the order may be transported with configurable soft constraint violations, allowing violations of pickup and delivery times. If still no vehicle is found the order remains unallocated. This is usually the case if order data is invalid and the consequent transport planning problem is over-constrained (e.g. impossible driving times, volume or weight above vehicle limits).

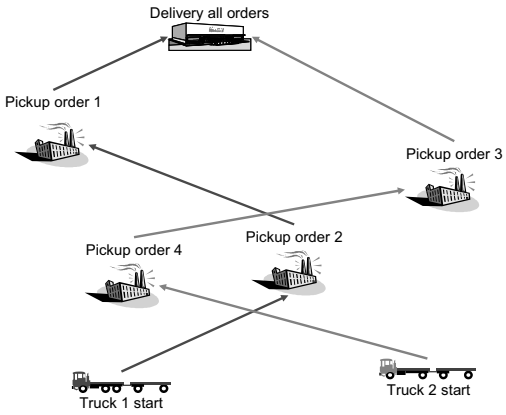


FIGURE 1. Example of a suboptimal solution given by the insertion algorithm for 4 orders.

3.2. The Optimization Approach

Sequential insertion with requests for quotes to all vehicles potentially produces suboptimal solutions, see for instance the example given in Figure 1. Order 1 is the first to arrive and is assigned to vehicle 1's route. Order 2 is also optimally assigned to vehicle 1's route since that produces least additional costs (and kilometers). When order 3 arrives vehicle 1 is fully loaded, therefore a new vehicle 2 is used for order 3 and later for order 4.

In order to improve the solution a further optimization step is performed by cyclic transfers between vehicles [15]. A cyclic transfer is an exchange of orders between routes. More specifically in a $b-cyclic\ k-transfer$ k orders are cyclically exchanged between b routes [11]. In our case, k defines the maximum number of orders transferred. A 3-cyclic 3-transfer might then be equivalent to moving 3 orders from the first to the second route, namely 1 order from the second to third route and 2 orders from the third to the first route. Figure 2 shows how the suboptimal example in Figure 1 is improved by a 2-cyclic 1-transfer. Order 2 is transferred from route 1 to 2 while order 4 is transferred from route 2 to 1.

The optimization procedure must determine which b-cyclic k-transfers should be triggered. Adding a new transport request changes a single vehicle's route r_1 (orders with a volume above a vehicle's capacity are assumed to be split before starting the optimization process). Assuming that at a time t before an order insertion an optimal solution is found, the only point for improving transfers is r_1.

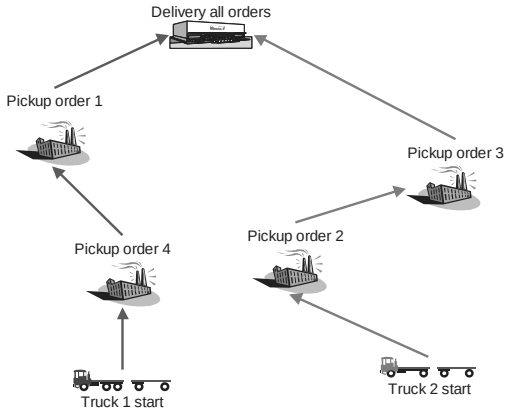


FIGURE 2. Optimal solution, for the example given in Figure 1, after a 2-cyclic 1-transfer.

Therefore, r_1 is initiating transfer requests to all other vehicles. The number of k-transfer requests between two routes is determined by the number of groups g of orders of a route r with at most size k $|g_{r,k}|$. All combinations of pairs of groups between the two routes produce a transfer request. The number of requests therefore grows considerably and limits the applicability of the algorithm to domains with few loads (< 10) per vehicle or to small k values. From the thus created transfer neighborhood the most significant cost-saving b-cyclic k-transfer is performed, i.e. the transfer that reduces the most the overall costs. This changes route r_1 and b other routes r_i. This hill climbing process is continued with all changed routes until no more cost-saving exchanges can be performed. In our case, only 2-cyclic transfers were allowed for performance reasons (see Section 4.3). The workflow of a 2-cycle k-transfer hill climbing algorithm is summarized in Algorithm 2.

3.3. The Agent-Based Design

Solving a dynamic m-PDPSTW can be distributed among multiple interacting agents in order to (1) achieve scalability of performance with growing sizes of problem instances; (2) directly reflect the distributed nature of transportation networks/organizations and decision making centers; (3) facilitate the handling of local deviations without the need to propagate local changes and recompute the whole solution, and (4) increase robustness (avoiding single point-of-failure).

Several agent-based designs are possible to distribute the work. The most radical one is to represent each vehicle by an agent [3, 9]. Solution generation by sequential insertion is then handled by a contract-net interaction protocol [3]. The optimization algorithm can then be modified in order to trigger a transfer between two vehicles whenever this improves the objective function, rather than to look for the best of all possible transfers. This modification improves the handling of multiple vehicle routes which can change concurrently. As a matter of fact, deciding

Algorithm 2: The 2-cycle k-transfer algorithm.

```
Store changed route r1 in list l
 For all other routes ri
  Check compatibility of vehicles
  For all g r1,k
    For all g ri,k
      Perform exchange
      Check r1 constraints and schedule
      Check ri constraints and schedule
      Calculate cost savings
      If highest cost savings
        Store option
 If option found
  Perform best exchange
  Add changed routes to l
 Else
  Remove r1 from l
Repeat until l is empty
```

on a transfer only after having asked all other vehicles has a higher probability that the transfer is then no longer possible. However, in the modified version of the optimization algorithm, the hill-climbing is no longer along the steepest slope and it is possible that more transfers are necessary to achieve the same solution. Moreover, the optimization process may end in a different local optimum. In such a fully distributed architecture, vehicle agents with a changed route start an optimization process in parallel to the other active vehicle agents. The main advantage of such an approach is in its fine granularity and high scalability, while its main disadvantage stems from a considerable overhead in computation time and resource usage. The overhead in computation time is mostly due to more expensive agent communications when compared to a fully centralized solution, while the overhead in resource usage depends on the memory and processing footprint of an agent.

The agent design chosen for our work reflects the way logistics companies today manage the complexity of this domain. The transportation business is usually divided into regions/clusters. Transportation requests arriving at a cluster are first tentatively allocated and possibly optimized within that cluster. If the order's pickup or delivery location is in a different cluster, the other cluster is also informed and asked to handle the request if it can do so in a cheaper way. In our agent-based framework, distinct software agents represent different regional clusters. All vehicles starting in the region of an agent are managed by a local *AgentClusterManager*. Incoming transport requests are distributed by a centralized *AgentDistributor* according to their pickup location (see Figure 3). Sequential

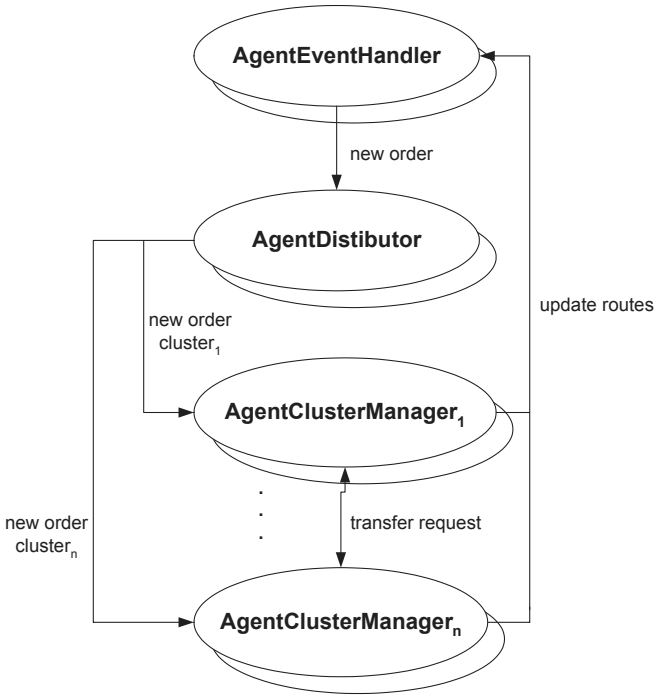


FIGURE 3. A summarized view of a cluster-based agent design.

order insertion can be achieved as summarized by the Algorithm1 (see Section 3.1). The only difference is that 'all vehicles' in this case is restricted to all vehicles within a given cluster. A sensible extension is to forward a transportation request to another cluster, in case transportation is not possible within its current cluster. The quality of the solution (from a global perspective) is expected to decrease with the number of defined clusters since order insertion does not take vehicles of other clusters into account. This is compensated by optimization transfers. The optimization algorithm within a cluster is the same as described in Section 3.2. Additionally, vehicles with routes spanning across other clusters may also initiate transfer requests among clusters.

The main advantage of this latter design stems from its direct mapping to today's transport business organization and its good scalability. The computational overhead incurred by such a multi-agent based solution is also much lower than that occurring in a fully distributed solution. Nevertheless, besides degradation of the solution's quality when compared to a fully centralized approach, the main disadvantage (also with respect to the fully distributed option) is that optimization within a cluster and among clusters has to be handled slightly differently.

4. Experimental Results

The empirical tests were run for a European logistics company with the aim of evaluating the potential cost savings of the introduction of a transport optimization system. The dataset contains roughly 3500 real-business transportation requests. The constraints and cost model have been modelled and used as described in Section 2. While the primary goal of the logistics company has been reduction of costs, further objectives have focused on the reduction of the number of deployed vehicles and the total amount of driven kilometers, while increasing the utilization of the vehicles.

The agent platform used for the evaluations is Whitestein Technologies' *Living Agents Runtime System* (LARS)[3]. Tests were performed on a 2 GHz Pentium machine running on Linux. For the results presented in section 4.5, four 800 MHz Pentium PCs running Linux were used.

4.1. Comparison to Manual Dispatching

In order to produce results of agent-based optimization that are comparable with the results achieved by the professional dispatchers, it has been necessary to address a number of issues. We used the same underlying geo-coding information system (distance and drive time information) that was used by the dispatchers. The average cost values for tariff classes (see Section 2.3) have been obtained from the real costs incurred in the corresponding tariff classes. Soft time windows and therefore soft constraint violations were only allowed if order data did not allow an in-time pickup or delivery. The resulting delivery plan was checked by dispatchers for feasibility and drivability.

Table 1 summarizes the comparison of the major results. A total of 11.7% cost savings was achieved, where 4.2% of the cost savings stem from an equal reduction in driven kilometers. Another 2.2% is achieved by increasing the number of cost-saving tramp traffic (see Section 2.3) by 380%. The remainder stems from buying cheaper vehicles. The cost-based optimization prefers routes that start in cheap regions. An additional important achievement is that the number of vehicles used is 25.5% lower compared to the manual solution. This is due to a higher utilization of the vehicles and an on-average longer usage of a single vehicle. In this sense, the cost savings would even be higher if fixed costs for the vehicles were feasible, which is not the case in charter business, but possibly in other transportation settings.

4.2. Allowing Order Transfer

The evaluated logistics business is a so called *part load business*. This means that multiple orders are usually loaded on any given vehicle. Therefore the optimization result should improve with a growing number k of orders involved in a single transfer. Figure 4 shows the optimization results for the above described example when growing k ($k = 0$ means that no transfers are allowed). The corresponding outcome is therefore the result of the sequential insertion algorithm described in

[3] Today's "Living Systems Technology Suite", see http://www.whitestein.com/, for more details.

Evaluation Parameter	Savings (agent-based)
overall cost	11.7%
driven kilometers	4.2%
deployed vehicles	25.5%

TABLE 1. Savings (in percentage) achieved by the agent-based approach when compared to manual dispatching.

Section 3.1. For $k = 1$ a single order may be moved from one vehicle to another or two vehicles may exchange a single order. The achieved cost savings over $k = 0$ is 6.7%. An additional 0.3% savings are achieved with $k = 2$ and 0.1% with $k = 4$. For $k = 3$ the result is minimally ($< 1\,‰$) worse compared to $k = 2$. This occurs because all results are potentially local optima.

The execution time for optimization of the orders grows as expected with increasing k. The almost linear growth of execution time can be explained by the average number of orders per vehicle $|\bar{o}| = 1.7$. This means that the number of candidate vehicles with two or more orders is decreasing with an increasing k. The absolute runtime of 13.29 minutes for this dataset emphasizes the near real-time nature of our optimization approach. In particular, real time events such as data change or delays of vehicles are usually inserted and optimized within one second.

4.3. Using Several Vehicles for Order Transfers

The above results were obtained by using 2-cyclic transfers of orders. There exist situations where no exchange between two vehicles will produce cost savings, but exchanges between three or more vehicles will. To estimate the gap in solution quality of 2-cyclic transfers to 3-cyclic transfers we generated a set of 10 million random transport situations with three vehicles containing one order each. Each situation was first optimized using all possible 2-cyclic exchanges with cost savings. The solution was then compared against all possible 3-cyclic exchanges.

An additional optimization was possible in just 0.04% of the total cases, saving only 0.02% costs. The gap should be even smaller for $b > 3$, although we did not test it. The increase in execution time for checking b-cyclic exchanges would on the other hand be considerable. The additional gain is therefore not considered to be achievable in near real time.

4.4. Soft Time Windows

Some logistics companies today offer different levels of service for the delivery of transport requests. They offer the customer the choice of a more expensive 100% in-time pickup and delivery or different cheaper levels with possible delays. This can be modelled using soft time windows.

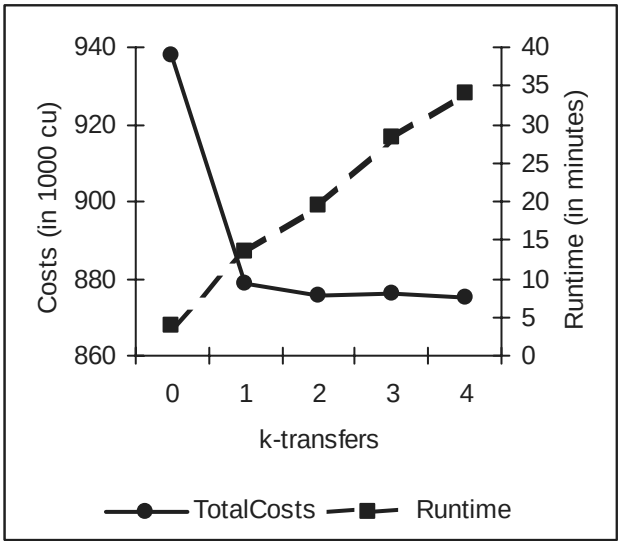


FIGURE 4. Influence of transfer parameter k on result quality (cost) and runtime.

To estimate the impact of soft time windows a series of optimization runs with increasing allowed delay for pickup and delivery of orders have been performed. Time violations were perturbed by 100 cu (currency unit) fixed and 50 cu/hour variable violation costs. Therefore only opportunities for cost-saving delays with a relatively high cost saving are taken. This happens, for instance, when an order is loaded on a vehicle that already drives from the order's pickup to delivery location, despite some minutes delay on the delivery, instead of allocating that load to an additional and possibly almost empty separate vehicle.

Figure 5 shows the results we obtained in this case. As expected, the less stringent constraints must be obeyed in relation to the lower the overall costs for transportation. On the other hand, the number of constraint violations grows. Logistics companies could use this type of agent-based simulation in order to estimate their possible cost savings and thereby tune and refine their cost structure accordingly.

4.5. Distributed Optimization

As described in Section 3.3, the solution generation and optimization can be distributed across multiple agents and therefore also across multiple agent platforms. The above results were obtained using a single agent for global optimization. Currently, by means of a distributed cluster-based approach running the optimization across four agents and four servers we obtained promising initial results.

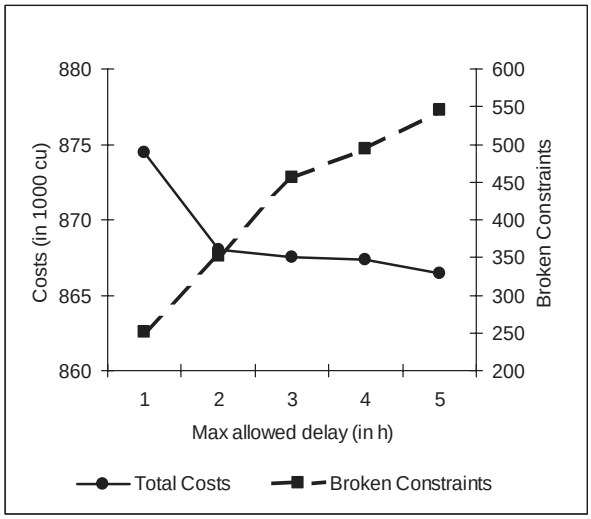


FIGURE 5. Influence of soft time windows variations to the optimization result versus the number of constraints violations.

The corresponding clusters are disjunct and no inter-cluster optimization has been deployed in these initial experimental tests. Optimization turned out to be nine times faster than when using a centralized agent-based approach. Roughly a factor of three improvement is achieved by the distribution of agents on four servers, another factor of three improvement is due to the regional clustering itself. The loss in quality, around 1.2% cost increase in comparison to the results discussed above, should be reduced by enabling transfers between clusters. Experimental tests are currently ongoing.

5. Conclusion

In this paper, an agent-based approach to solve a real-world dynamic pickup and delivery problem with soft time windows has been presented. The problem model, including constraints specification and cost function definition, has been expressed in such a way as to make it possible to obtain realistic delivery plans (validated in real-world transportation business scenarios) that can be executed during daily business. The presented solution produced significantly better outcomes when compared to the results of professional dispatchers. Such results are currently achieved by hill climbing-based optimization.

In a real-world productive environment, optimization is an ongoing process running the whole day. When less busy times arise and more computational power becomes available, it is possible to envisage the use of more sophisticated optimization algorithms to further explore and refine the solution space so as to improve the

quality of the achieved results[12]. In this perspective, dynamic coordination of the various computational entities (agents) makes it possible to orchestrate a change in the deployed problem solving techniques. This is the basis of our agent-based approach.

Our current work focuses on the distribution of the optimization across multiple agents and agent platforms. The distributed optimization has been validated and already deployed to produce first experimental results. However, a number of improvements are ongoing, mainly concerned with enabling transfers between clusters and simplifying the process of configuring distributed agent servers. Finally, we also plan to estimate the possible performance and measure the real overhead in runtime and memory usage for a fully distributed version.

References

[1] T. G. Crainic. Long haul freight transportation. In R. W. Hall, editor, *Handbook of Transportation Science*. 2^{nd} Edition, Kluwer, 2002.

[2] Exel. Cost reduction still the biggest pressure on logistics for the industrial sector. In *http://www.exel.com/mediacentre/newsreleases/releases.asp?int_articleID=710*, London, UK, 2004.

[3] K. Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: an application domain for DAI. *Journal of Applied Artificial Intelligence*, 10, 1995.

[4] M. Gendreau and J. Potvin. Dynamic vehicle routing and dispatching. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 115–126. Kluwer, 1998.

[5] M. Hickman and K. Blume. A method for scheduling integrated transit service. In 8^{th} *International Conference on Computer-Aided Scheduling of Public Transport (CASPT)*, 2000.

[6] I. Ioachim, J. Desrosiers, Y. Dumas, M. Solomon, and D. Villeneuve. Clustering algorithm for door-to-door handicapped transportation. *Transportation Science*, 29(1):63–78, 1995.

[7] J.-J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research*, 20 B(3):243–257, 1986.

[8] R.M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[9] Robert Kohout and Kutluhan Erol. In-time agent-based vehicle routing with a stochastic improvement heuristic. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99); Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*, pages 864–869, Menlo Park, Cal., July 18–22 1999. AAAI/MIT Press.

[10] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[11] Snezana Mitrovic-Minic. Pickup and delivery problem with time windows: A survey. Technical Report TR 1998-12, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 1998.

[12] W. P. Nanry and J. W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research*, 34B:107–121, 2000.

[13] H. Psaraftis. Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, 61:143–164, 1995.

[14] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[15] P. M. Thompson and H. N. Psaraftis. Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41(5), 1993.

[16] Hang Xu, Zhi-Long Chen, Srinivas Rajagopal, and Sundar Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364, 2003.

Klaus Dorer and Monique Calisti
Whitestein Technologies AG
Pestalozzistrasse 24
8032 Zürich, Switzerland
e-mail: {kdo,mca}@whitestein.com

Designing Multiagent Decision Support Systems for Traffic Management

S. Ossowski, A. Fernández, J.M. Serrano,
J.L. Pérez-de-la-Cruz, M.V. Belmonte, J.Z. Hernández,
A.M. García-Serrano and J.M. Maseda

Abstract. Modern Decision Support Systems (DSS) not only store large amounts of decision-relevant data, but also aim at assisting decision-makers in exploring the meaning of that data, so as to take decisions based on understanding. To this end, a distributed approach to the construction of DSS has become popular: decision-support agents are responsible for parts of the decision-making process in a (semi-)autonomous (individually) rational fashion. However, despite the advances in the field of agent-oriented software engineering, a principled approach to the design of multiagent systems for decision support is still to come.

In this paper, we outline design guidelines for the construction of agent-based DSS. In particular, setting out from an organisational model of decision support environments, we present an abstract architecture for multiagent DSS. We then show how this architecture can be instantiated to a particular real-world domain, traffic management, in a principled fashion and discuss the lessons learnt from this enterprise.

1. Introduction

Decision Support Systems (DSS) provide assistance to humans involved in complex decision-making processes. Early DSS were conceived as simple databases for storage and recovery of decision relevant information [20]. However, it soon became apparent that the key problem for a decision-maker (DM) is not to access pertinent data but rather to understand its significance. Modern DSS help decision-makers

Work supported by the Spanish Ministry of Science and Technology (MCyT) under grant TIC2000-1370-C04. The authors would like to thank the Public Works and Transport Dept. of the Local Government of Bizkaia (DFB) as well as the Malaga Local Transport Consortium (EMT) for their cooperation.

explore the implications of their judgements, so as to take decisions based on understanding [9].

Knowledge-based DSS [15] are particularly relevant in domains where human operators have to take operational decisions regarding the management of complex industrial or environmental processes [10][11]. Due to the inherent (spatial, logical, and/or physical) distribution of these domains, a distributed approach to the construction of DSS has become popular [5]. Decision-support agents are responsible for parts of the decision-making process in a (semi-)autonomous (individually) rational fashion: they collect and facilitate decision relevant data [20], but also provide advanced reasoning services to analyse the meaning of this information [17]. However, despite the various advances in the field of agent-oriented software [13][12], a principled approach to the design of knowledge-based multiagent systems for decision support is still to come.

This paper reports on the design of two multiagent DSS for real-world traffic management problems. In addition, it shows how the use of an abstract multiagent architecture for DSS may provide valuable design guidelines for this kind of systems. Section 2 sketches the design process that leads to the definition of this abstract architecture. Section 3, outlines its role in the design and implementation of actual DSS prototypes for the problems of road traffic management in the greater Bilbao area, as well as bus fleet management scenarios pertinent to the Spanish town of Malaga. In section 4, we give examples of the dynamics of our systems, and summarise the lessons we have learnt from this case study in section 5.

2. An abstract architecture for multiagent DSS

In line with the mainstream in agent-oriented software engineering (e.g. [22]), we first model agent-based DSS in terms of organisational concepts [23][6], that are then further refined, so as to give rise to an agent-centred model, which will be the basis for our abstract architecture. Additional requirements for the design process are the need to take advantage of current standards (in particular FIPA [8]) and to allow for the integration of legacy software for decision support (such as the KSM knowledge modelling tool [4]).

The key point in modern DSS is to *interact* with the DM to assist her in exploring the implications of her judgments. So, in order to develop a generic organisational model of DSS, it is natural to set out from an analysis of typical *decision support dialogues* between DM and DSS [19]. Based on their (macro-level) functionality, we have identified the following *types* of communicative interactions [18]: *information exchange*, *explanation*, *advice*, and *action performing*. In DSS with multiple DM, additional brokering and negotiation interactions are often present. Various *communicative roles* participate in interactions, and are characterised by the Communicative Actions (CA) [1] that they perform, and by the communication *protocols* they may engage in. For instance, the *information seeker*

communicative role that takes part in *information exchange* interactions is characterized by the FIPA CAs *query-if*, *query-ref*, and *subscribe*, and may take part in *FIPA-query* and *FIPA-subscribe* protocols [18].

Still, roles in agent-based DSS require domain competence as well, so we specialise communicative roles into social roles based on the different elements of the domain ontology that define the contents of the corresponding communicative interactions (information exchange, explanation etc.). This ontology will at least contain concepts relating to system *problems*, problem *causes* and *control actions*. A minimum domain competence of a DSS will include reasoning services to (1) identify situations with decision-making options (classification), (2) express system problems in terms of causal features of the situation (diagnosis), (3) generate various decision alternatives (action planning) and (4) simulate potential consequences of decisions (prediction) [17]. Fig.1 summarises the result of our organisational analysis of DSS by means of a social role model in UML notation.

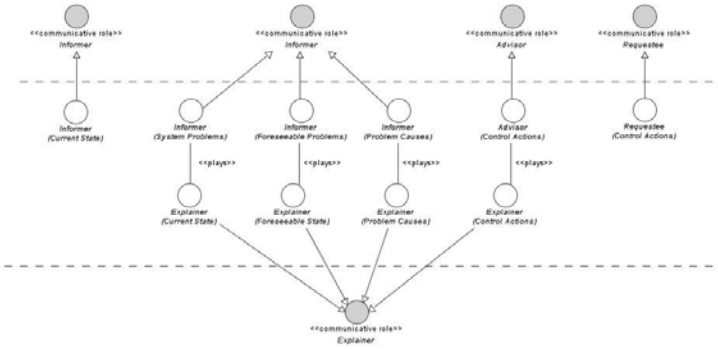


FIGURE 1. Social Role Model for DSS

In order to develop an agent model for decision support applications, social roles need to be mapped onto types of agents that will eventually play these roles in the DSS. Usually, both of the following cases occur:

- one agent - several roles: this is the usual approach aimed at simplifying the resulting software architecture (and avoiding an unnecessary replication of knowledge)
- one role - several agents: Due to the "a priori distribution" that is typically present in decision support domains, it is usually convenient to let several agents play the same role, but in different "parts" of a system. In this way, the agent model may better reflect a human organization, reduce communication requirements, or simply decrease the complexity of the necessary reasoning processes.

Based on the social roles that we have identified previously, we have come up with different abstract agent types for DSS. Note that, in order to better reflect "a priori distribution", several instances of the following agent types will coexist:

- There are two classes of *Connection Agents* that directly interact with the environment: *Data agents (DA)* play the informer role with respect to the current state of a certain part of the system. As such, they are in charge of information retrieval from different information sources like sensors or databases, and its distribution. *Action Implementation Agent (AIA)* play the requestee role and are in charge of actually executing the actions that the DM has chosen to take.
- *Management Agents (MA)* play the remaining informer roles outlined in Fig.1 as well as the advisor and explainer roles. By consequence, they need to be endowed with knowledge models that allow them to report on (and justify) problems, causes, potential future states etc, as well as to suggest potential management actions. They may rely on the services of *Coordination Facilitators (CF)* that provide support for negotiation and matchmaking (recruiting, brokering) interactions [16]. CF agents are particularly relevant in DSS for *groups* of decision-makers.
- *User Interface Agents (UIA)* play the remaining social roles outlined in Fig.1 (informee, requester, etc.) on behalf of the user. Note that by conveniently sequencing and/or interleaving conversations, they are capable of answering a variety of questions (e.g., "What is happening in S?", "What may happen in S if event E occurs?", etc) [17].
- *Peripheral Agents (PA)* represent the support infrastructure for the DSS. Directory Facilitators and Agent Management Systems as required by the FIPA abstract architecture [8] usually constitute PAs, but they may also encapsulate third-party value added services.

Fig.2 shows the resulting abstract agent-based DSS architecture. Notice that the instantiation of this architecture to a particular domain may induce a further subdivision of abstract agent types (e.g. to allow for more flexible dialogues with the decision-maker). In addition, the use of wrapper agents for legacy software may require that some abstract agent type be finally implemented by means of a society of agents.

3. DSS for Traffic Management

In this section we illustrate the instrumentation of our approach by two case studies in the domain of traffic management. The present research is the continuation of a long line of previous work in this field [3][10][7]. In the sequel, we show how the abstract agent types of our abstract architecture are instantiated in both scenarios.

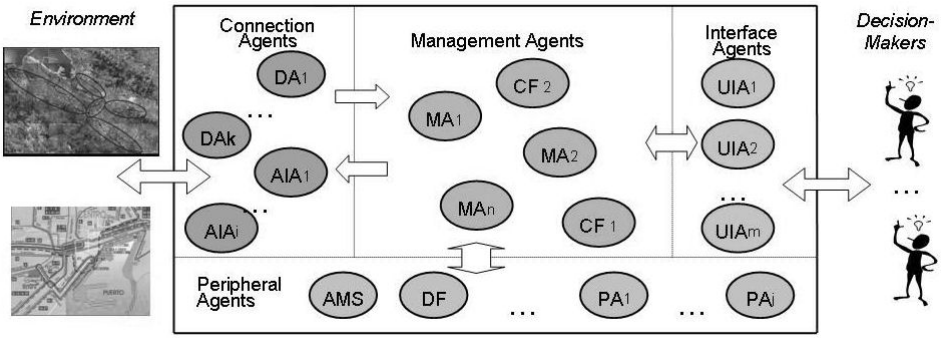


FIGURE 2. Abstract architecture

3.1. Problem Scenarios

Bilbao demonstrator. The first application of the multiagent DSS architecture refers to the domain of road traffic management. In particular, we are concerned with a part of the high-capacity road network in the Bilbao area, comprising the town's ring road as well as four of the main accesses to the metropolitan area.

Regular information about the traffic state in this highly used area, registered by loop detectors, is received in the Mobility Management Centre located at Malmasin, in the vicinity to the city of Bilbao. On the basis of this data, traffic operators have to take decisions on the control actions to apply in order to solve or minimise congestions. These actions include (1) displaying messages in Variable Message Signal (VMS) panels installed above the road to warn drivers about traffic problems or recommend alternative routes and/or (2) contacting local authorities to send appropriate people to manage the situation. As the traffic control infrastructure becomes more complex, there is an increasing need to assist operators in their management task, helping them to configure consistent control plans for the whole road network, and exploiting adequately the available signal devices from a global perspective. This is precisely the purpose of the Bilbao DSS demonstrator described in the following sections.

Malaga demonstrator. In many major cities, urban buses are equipped with radio and GPS devices that provide operators in a Bus Fleet Management (BFM) centre with up-to-date information on bus locations, and allow them to communicate with the drivers. A typical task of a BFM operator is to detect incidents (delays, advances, breakdowns,..) by comparing a bus' schedule with current location data, and to send orders to bus drivers (increase/reduce speed, change timetable regulation to frequency regulation,..) so as to maintain or re-establish an acceptable quality of service.

In the Spanish town of Malaga, control centre operators of the local transport consortium (EMT, Empresa Malaguena de Transporte) use an Exploitation Support System that presents information related to the status of the buses with regard to the scheduled services. There is a line inspector for each line who takes decisions in order to adjust services to unforeseen circumstances, and supervisors that are in charge of taking broader and more complex decisions for several lines.

In the following sections, we report on the architecture of a prototype DSS that extends the functionality of the Exploitation Support System, engaging in dialogues with EMT operators respecting the causes of problems and the best control actions to take. Our prototype faithfully reproduces the real operating conditions of the EMT bus lines that cover a sector in western Malaga.

3.2. Connection Agents

Bilbao demonstrator. The whole traffic network scenario in Bilbao has been divided in 12 overlapping problem areas to support a better analysis and understanding of the causes and evolution of traffic problems than if performed from a global perspective. A problem area is identified as a section of the network where traffic behaviour can be locally studied and suitably classified.

Every Data Agent (DA) in the Bilbao demonstrator is in charge of collecting the information about the traffic state in a particular problem area, delivered by the loop detectors (i.e. speed, occupancy and flow), and providing this information to any interested agent. This communication is supported by the FIPA-Query interaction protocol playing the DA the participant role.

Malaga demonstrator. *BFM Data Agents (DA)* indicate the state of the bus lines as part of information exchange interactions. In particular, when a bus arrives at a stop or an incident takes place, the DA forwards this information to any agent that previously subscribed to this service by means of the FIPA-Subscribe protocol. *BFM Action Implementation Agents (AIA)* simply forward commands to bus drivers via radio.

One way to implement the domain competence of BFM connection agents is to provide a wrapper for the EMT Exploitation Support System, which collects information on the actual state of the buses in real-time by means of GPS technology. However, as we were not allowed to establish such an online connection, for our prototype we have implemented a simulator based on the actual EMT bus schedules (Fig. 3). The simulator's functionality and its interface emulate the EMT Exploitation Support System. Furthermore, we have provided the simulator with additional functionalities that allow us to generate complex problem scenarios (concurrent delays, advances, saturations, breakdowns, etc), so as to calibrate and evaluate our prototype.

3.3. Management Agents

Bilbao demonstrator. For road management demonstrator, Management Agents have been instantiated into a society of agents. On the one hand, as we will outline

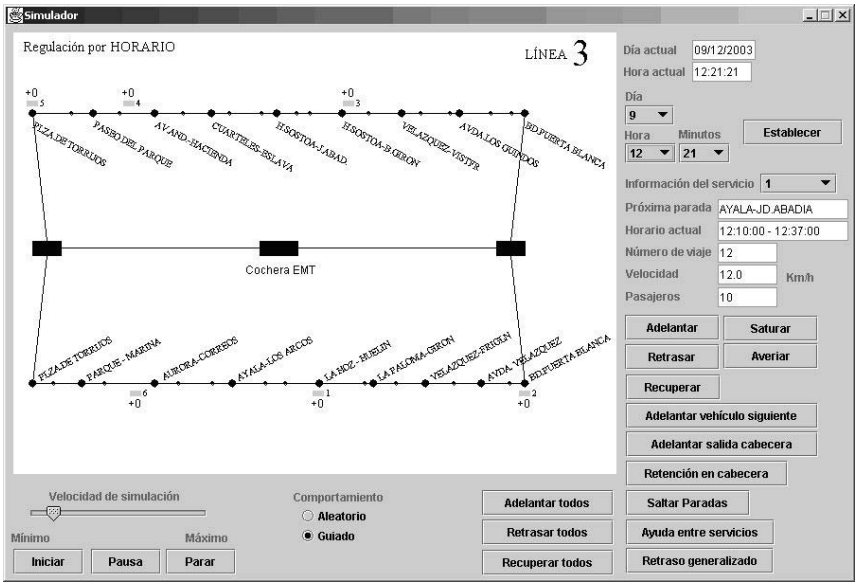


FIGURE 3. Screenshot of the BFM simulator

below, the reuse of parts of the KSM knowledge tool required the introduction of wrappers called KSM agents. On the other hand, the different granularity at which traffic management engineers conceive problem identification and diagnosis on the one hand (problem areas), and action planning and impact estimation on the other (control zones), suggested the use of Problem Detection Agents (PDA), and Control Agents (CA), respectively. Fig.4 reflects the resulting structure of the Bilbao demonstrator.

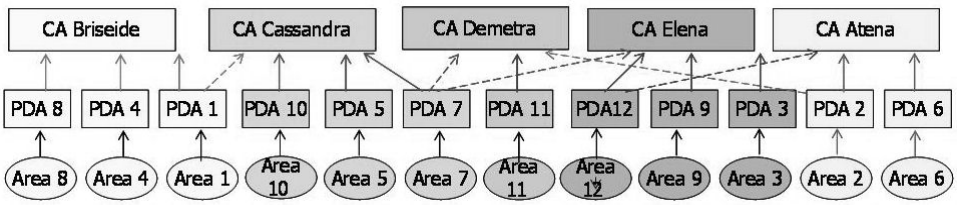


FIGURE 4. Problem areas and control zones

PDAs are responsible for monitoring the traffic flow in a problem area and detecting congestions. The arrow from Area_i to PDA_i in Fig.4 represents this relation. Whenever a problematic situation is detected the PDA requests a CA to generate possible action plans to overcome it (solid arrows in Fig.4). A CA's control

zone comprises several problem areas, so it may have to solve/alleviate various congestions simultaneously. For this purpose, it asks the PDAs that monitor the problem areas surrounding the congestion for information about their general state (dashed arrows in Fig.4), so as to generate coherent signal plan proposals for the entire control zone. Such a control proposal consists of a collection of messages to be displayed on VMS panels with warnings or recommendations for alternative routes for drivers approaching the congestion. When several problems are detected in different zones, and two or more CAs compete for the use of the same VMS panels, the CAs involved exchange their control proposal and find an agreement based on conventions that establish (time-dependent) priorities.

The knowledge model supporting this entire process assigns PDAs data abstraction knowledge to infer additional values that help to evaluate the global state of a problem area, as well as problem identification knowledge. The former is represented by means of JESS [14] production rules, while the latter is realised by matching the current traffic situation against prototypical problem scenarios, a task delegated to a KSM agent which hosts the corresponding frame knowledge base. On the other hand, CAs require knowledge about which PDAs to consult to diagnose a congestion, the definition of signal proposals, the detection of conflicts with other CAs for the use of VMS panels, and conflict resolution. In fact, for the latter purpose two different knowledge bases are used: one to solve local conflicts between proposals within the same control zone (different for every CA) and another with conventions on how to solve conflicts between different zones (shared among CAs). Besides the conflict resolution knowledge, which is represented using JESS production rules, the domain competence required by a CA relies on a frame-based representation, so it is instrumented by means of requests to KSM agents.

PDAs use the following interaction protocols: (1) FIPA-Subscribe: whenever a problem is detected the information about it is delivered to the agents subscribed to this service (CAs and UIAs), playing the PDA the participant role. (2) FIPA-Query: the PDA initiates this protocol to request new traffic data from the DA, and it plays a participant role to provide a CA with information about the state of the problem area. Regarding CAs, FIPA-Subscribe is used to support the subscription to the PDAs that should inform the CA about the presence of a problem and to provide the UIA the description of new signal proposals, playing the CA the initiator and participant role respectively. FIPA-Query is also used to ask a PDA for information on the traffic state in its problem area, and to exchange signal plan proposals with other CAs in case of conflict.

Both PDAs and CAs interact with their associated KSM agents to request the execution of a pattern matching process using FIPA-Request (to load the frame knowledge base) and FIPA-Query (to obtain the result of the process), always in the initiator role. For instance, Fig.5 shows one of the congestion patterns that KSM agents, on request of the corresponding PDA, use during the matching process.

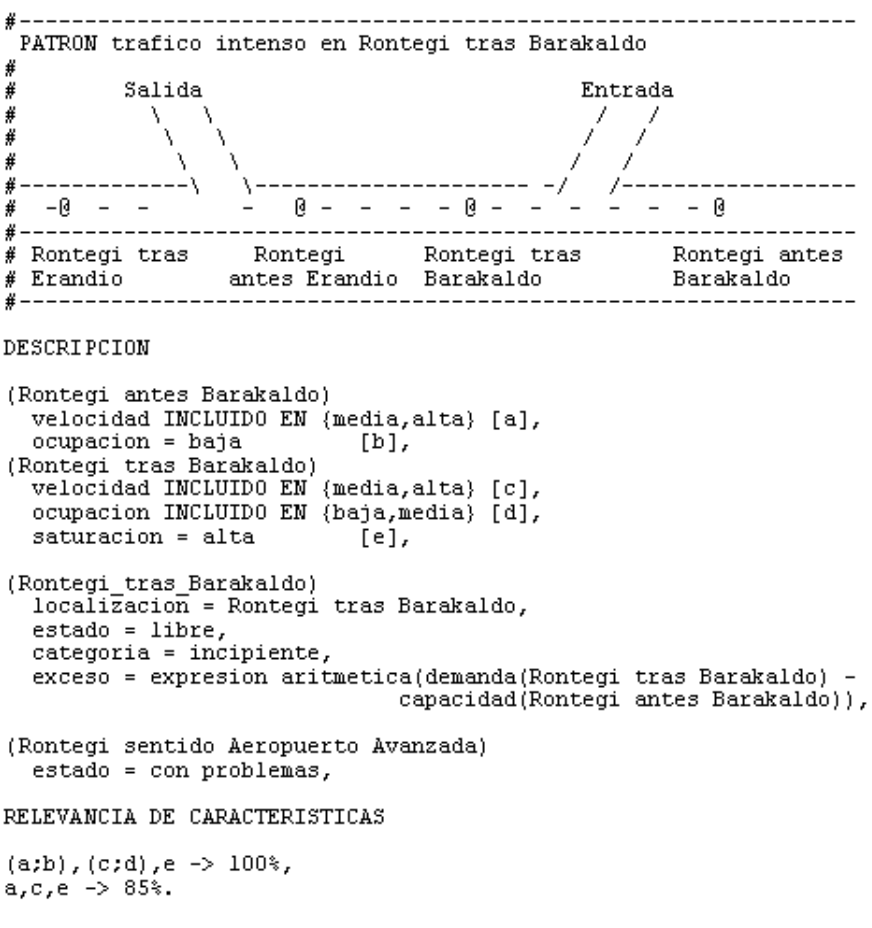


FIGURE 5. Problem detection frame

Malaga demonstrator. *Line Management Agent (LMA)* are the direct counterpart to MAs in the BFM domain. They are in charge of bus line supervision and, in accordance with the actual conceptual and organisational structure en EMT's control centre, there is one LMA for each line. An LMA's main purpose is to participate in information exchange interactions respecting incidents, problem causes, and control recommendations.

As Fig.6 indicates, LMAs use the following interaction protocols: (1) FIPA-Subscribe: the LMA plays the initiator role to obtain information about arrivals and incidents from the DA; and plays the participant role to facilitate information about problems, causes, and control recommendations of the line to the UIA; (2) FIPA-Brokering: an LMA may need a reinforcement (reserve) service and try to find other lines willing to hand over a vehicle. The LMA initiates this interaction

with a Coordination Facilitator (CF) Agent (see below), who is in charge of locating adequate LMAs by means of a FIPA-query sub-protocol. (3) FIPA-Query: before starting a negotiation with others to obtain a reinforcement service, an LMA asks its DM (through the UIA) for authorisation. In addition, an LMA may need to reply to the CF's query as to whether it is willing to accept a certain service transfer deal; and (4) FIPA-Request: LMAs use this protocol to actually execute an agreement to shift a vehicle from one line to another.

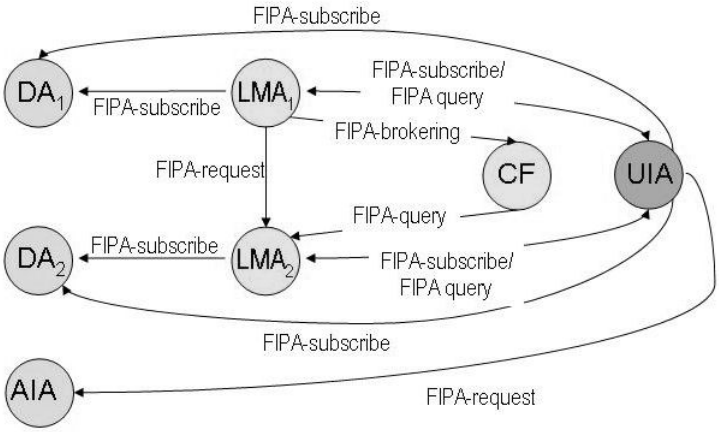


FIGURE 6. BFM Interactions

The instrumentation of LMA domain competence requires a knowledge model that allows them to identify or diagnose problems, suggest or recommend sets of management actions and predict future behaviour of the line. In fact, several elicitations interviews have been performed with the EMT operators in order to extract the knowledge and logs of real situations, and have been analysed in order to simulate and solve real problems. LMA knowledge has been represented by a set of JESS production rules, and its corresponding reasoning services are carried out by forward chaining inference. The ontology of the system has been modelled and instrumented by means of the PROTEGE-II tool.

A *Coordination Facilitator (CF)* supports the coordination among lines. More precisely, it acts as a mediator in the negotiation among LMAs to obtain a reinforcement service. When a line needs a reinforcement service (e.g. when the corresponding LMA detects the breakdown of a service, or the frequency of the services in the line is too low), the LMA asks the DM (by means of UIA) for a permission to negotiate a reinforcement service. If she accepts, the LMA requests negotiation support from the CF by means of the FIPA-Brokering protocol. So, the CF takes on the broker role and additionally plays the informee (initiator) role [19] in the

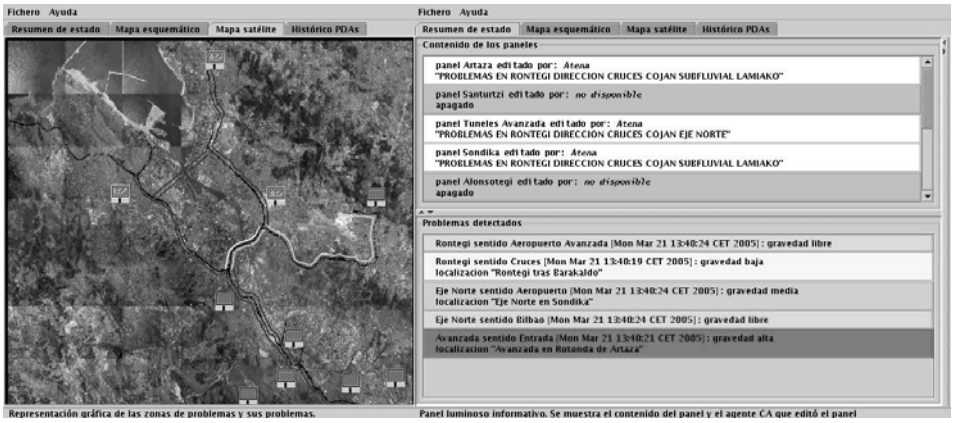


FIGURE 7. Traffic Management UIA

FIPA-Query sub-protocol. So, the CF's domain competence at least requires a model of the spatial relation of lines and their head stops.

3.4. Interface Agents

Bilbao demonstrator. *User Interface Agents (UIA)* show information to the operators (DM) at the Mobility Management Centre. This information includes the status of the traffic flows (problems) and the set of messages to be displayed on VMS panels (control proposal). For this purpose, the UIA initiates FIPA-Subscribe protocols to obtain information about current problems from PDAs and control recommendations from CAs.

The graphical interface generated by the UIA (Fig. 7) allows operators to select different visualisation modes for traffic information. On the left side of the figure, the traffic state is shown in relation to a satellite photo of the Bilbao area. The main roads are highlighted, and their colour represents their current status (free, and low, medium, or high congestion levels). The same information may be shown by a graphical schematic view, and in a textual mode. Details about traffic problems may be obtained by clicking on the corresponding entities. The right part of the interface shows control recommendations and their justifications.

Malaga demonstrator. *User Interface agents (UIA)* display selected information to the BFM operators (DM) at the control centre. This information shows the status of the line, informs about incidents or problems and notifies control recommendations or proposals coming from the LMAs.

With the goal of displaying this information, a UIA subscribes to the service offered by the DAs to obtain information about arrivals and incidents of each line. Besides, it initiates a FIPA-Subscribe protocol with LMAs, in order to obtain diagnostic information and control recommendations for each line. In addition, the

UIA plays the participant role in interactions, driven by the FIPA-Query protocol, to authorise negotiations for reinforcement services.

The UIA graphical interface is similar to the simulator (Fig. 3). However, it has the added functionality of displaying the control recommendations and the possibility of accepting or rejecting them by the DM. There is a different window or panel for each line.

3.5. Peripheral Agents

Both demonstrators have been implemented on top of the JADE platform [2]. At the time being, their only peripheral agents are the Directory Facilitator and an Agent Management System provided by JADE.

4. DSS agent interaction: two examples

4.1. Road traffic Management

The currently running version of the Bilbao demonstrator covers five problem areas and two control zones, so we have 5 DAs, 5 PDAs, 2 CAs, 11 KSM agents (one per problem area and three per control zone), as well as 1 UIA. Suppose the scenario shown in Fig.8 where three congestions happen simultaneously in problem areas 2, 6 and 11.

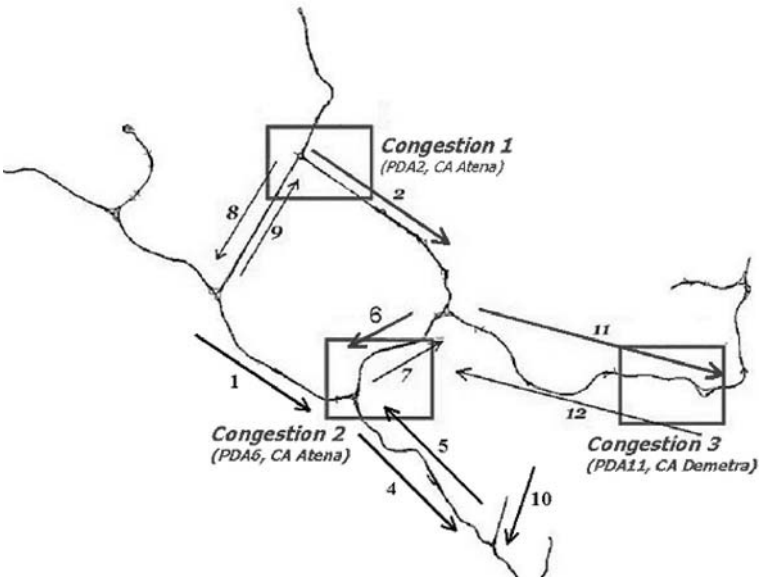


FIGURE 8. Example congestion scenario

The corresponding PDAs detect the problems from the traffic data sent by the DAs and inform the subscribed CAs (Atena and Demetra) about this situation. After gathering information on the general state of neighbouring areas (querying PDA12 and PDA7 respectively), Atena generates potential solutions to the congestions in problem areas 2 and 6, while Demetra configures signal plan proposals for the problem detected by PDA11.

Every CA in the DSS knows with which other CAs it may come into conflict so, once Atena and Demetra have achieved a consistent signal proposal for their control zones (making use local conflict resolution knowledge), they inform each other about these proposals in order to identify potential non-local interdependencies. In the example scenario, Atena and Demetra's best signal proposals happen to be mutually exclusive, since they try to display different messages on the same VMS panel. This conflict is solved locally by each CA, applying conventions respecting priorities for the use of the VMS panels, based on the severity of the problems and the time of the day when these problems occur. Finally, both CAs send potential signal plans for their control zones to the UIA, which shows them to the DM as a coherent global action proposal.

The knowledge bases of the Bilbao demonstrator are being refined offline with the AIMSUN/2 4.1 tool [21] that instruments micro-simulation models of different problem scenarios. Fig.9 shows a snapshot of one of the congestions mentioned above created with this tool.

FIGURE 9. Simulated congestion in area 6

4.2. Bus fleet Management

An interesting case of complex agent interactions in the BFM domain is given when a vehicle of a certain line suffers a breakdown. As EMT Malaga does not maintain a pool of reserve vehicles, it is necessary to negotiate with other lines, so as to agree on a transfer of a service from one line to the other. The following constraints apply: (1) for the supplier line the cost of losing its service should be lower than the cost of losing its service for the LMA that initiates the negotiation; and (2) the distance between the head stops of both lines (supplier and receiver) should be as low as possible. This "cost" is to measure the degradation in the quality service that a line offers to its users, and can be computed from statistics about passengers or, as in our example, just as a function of the reduction in the frequency of services (notice that this knowledge is private and not communicated to the remaining agents).

Suppose LMA12 detects a breakdown of a vehicle serving EMT line 12 in western Malaga, and recommends to ask for a reinforcement service. Furthermore, assume that the DM replies positively to the LMA's request to initiate a negotiation process. Fig.10 depicts the essentials of this negotiation process, which unfolds along three "rounds".

In a first round, LMA12 asks the CF for support to find LMAs with lines sharing its head stop (in the example, this is only LMA3); in addition, it queries them whether they are willing to transfer a service at an initial price of 110 (being LMA12's real cost 120). LMA3 determines its cost of losing a service as 169, so it rejects the offer. Subsequently, LMA12 increases the distance to the head stop and the CF forwards the same offer to LMA15 (with its head stop located at 1km), but it also refuses because of its cost of 264. At this moment, LMA12 initiates a second negotiation round increasing its offer to 154. The CF repeats brokering with the new price but all the LMAs decline again. Finally, in the third negotiation round, LMA12 offers a price of 198, which is accepted by LMA3 as it exceeds its actual cost. The CF communicates the acceptance to the LMA12, which sends a control recommendation to the UIA notifying that line 3 has accepted to transfer a reinforcement service. Again, the DM is in charge of the taking final decision, before the actual request for shifting the service is forwarded to line 3.

5. Conclusions

In this paper we have shown how multiagent technology can be successfully applied to build DSS for real-world traffic management problems. In particular, we have put forward design guidelines for the construction of agent-based DSS, leading to an abstract multiagent architecture. Furthermore, we have shown in detail how this abstract architecture has been used to design DSS prototypes for the domain of road traffic management in the greater Bilbao area, as well as bus fleet management scenarios pertinent to the Spanish town Malaga.

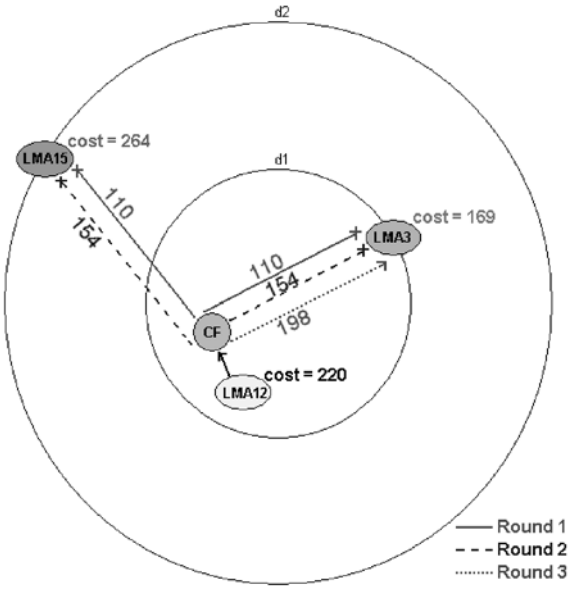


FIGURE 10. Example negotiation process

The implementation of the demonstrators, that required the integration of various software technologies and tools (JADE, KSM, JESS, Protégé-II), has been initially complex but required a reasonable amount of programming work. In particular, the agentification of the KSM knowledge representation and inference schemes [4] and their subsequent integration into JADE went smoothly. By contrast, the effort necessary to integrate the different ontologies (and the representations used in the different tools) was rather high, so we see a need for standards and tools that facilitate this kind of task.

Furthermore, we found that some communicative roles and interactions are not adequately supported by FIPA. In order to remain FIPA compliant, for instance, in our prototypes we had to implement *advisory* interactions by means of *information exchange* interactions (using FIPA-query or FIPA-subscribe protocols). As a result, we have developed a method to build principled extensions to ACLs (and FIPA ACL in particular) [19], as well as a set of software components that encapsulate the corresponding dialogical behaviour for its use by JADE agents, to be used in future applications.

We are currently refining the knowledge models of the prototypes, so as to evaluate their performance in collaboration with the Public Works and Transport Dept. of the Local Government of Bizkaia and the Malaga Local Transport Consortium (EMT). Future work comprises the integration of additional Peripheral

Agents (e.g. supply services of traffic management agents to the BFM system), and the use of mobile devices (e.g. onboard driver information systems).

References

[1] J. Austin. *How to do Things with Words*. Clarendom Press, Oxford, 1962.

[2] F. Bellifemine, A. Poggi, and G. Rimassa. JADE – a FIPA-compliant agent framework. In *4th Int. Conf. on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM'99)*, pages 97–108, Apr. 1999.

[3] J. Cuena, J. Hernández, and M. Molina. Knowledge-oriented design of an application for real time traffic management. In *European Conference on Artificial Intelligence*, pages 217–245. Wiley & Sons, 1996.

[4] J. Cuena and M. Molina. KSM – an environment for design of structured knowledge models. In Tzafestas, editor, *Knowledge-Based Systems*. World Scientific, 1997.

[5] J. Cuena and S. Ossowski. Distributed models for decision support. In Weiss, editor, *Multi-Agent Systems - A Modern Approach to DAI*. MIT Press, 1999.

[6] J. Ferber, O. Gutknecht, C. Jonker, J. Muller, and J. Treur. Organization models and behavioural requirements specification for multi-agent systems. In *ECAI 2000 Workshop on Modelling Artificial Societies and Hybrid Organizations*, 2000.

[7] A. Fernández, S. Ossowski, and A. Alonso. Multiagent service architecture for bus fleet management. *Integrated Computer-Aided Engineering*, 2(10), 2004.

[8] FIPA – The Foundation for Intelligent Physical Agents. http://www.fipa.org/.

[9] S. French. *Decision Analysis and Decision Support Systems*. The University of Manchester, 2000.

[10] J. Hernández, S. Ossowski, and A. García-Serrano. Multiagent architectures for intelligent traffic management systems. *Transportation Research C*, 5(10), 2002.

[11] J. Hernández and J. Serrano. Environmental emergency management supported by knowledge modelling techniques. *AI Communications*, 1(14), 2000.

[12] K. Hoa Dam and M. Winikoff. Comparing agent-oriented methodologies. In *Workshop on Agent-Oriented Information Systems (AOIS-2003)*, 2003.

[13] C. Iglesias, M. Garijo Ayestarán, and J. González. A survey of agent-oriented methodologies. In *Intelligent Agents V*. Springer-Verlag, 1999.

[14] JESS – Java Expert System Shell. http://herzberg.ca.sandia.gov/jess/.

[15] M. Klein and L. Methlie. *Knowledge-Based Decision Support Systems*. John Wiley, 1995.

[16] M. Klusch and K. Sycara. Brokering and matchmaking for coordination of agent societies: A survey. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 8, pages 197–224. Springer-Verlag, Mar. 2001.

[17] S. Ossowski, J. Hernández, C. Iglesias, and A. Fernández. Engineering agent systems for decision support. In P. Tolksdorf; and Zambonelli, editors, *Engineering Societies in an Agent World III*, LNAI, pages 234–274. Springer, 2002.

[18] S. Ossowski, J. Pérez de la Cruz, J. Hernández, J. Maseda, A. Fernández, M. Belmonte, A. García Serrano, J. Serrano, R. León, and F. Carbone. Towards a generic multiagent model for decision support. In *Spanish Conference on Artificial Intelligence (CAEPIA'03)*, LNAI. Springer, 2004.

[19] J. Serrano, S. Ossowski, and A. Fernández. The pragmatics of software agents – analysis and design of agent communication languages. In Klusch, Bergamaschi, Edwards, and Petta, editors, *Intelligent Information Agents – An AgentLink Perspective*, volume 2586 of *LNAI*, pages 234–274. Springer, 2003.

[20] M. Silver. *Systems that Support Decision Makers*. John Wiley, 1991.

[21] TSS. The getram/aimsun traffic simulation environment. http://www.aimsun.com/.

[22] M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multiagent Systems*, 3(3):285–312, 2000.

[23] F. Zambonelli, N. Jennings, and M. Wooldridge. Organizational abstractions for the analysis and design of multi-agent systems. In Ciancarini and Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 235–252. Springer-Verlag, 2000.

S. Ossowski, A. Fernández, J.M. Serrano
Universidad Rey Juan Carlos
Dpt. of Computer Science
e-mail: {sossowski,afernand,jserrano}@escet.urjc.es

J.L. Pérez-de-la-Cruz, M.V. Belmonte
Universidad de Málaga
ETSI Informática
e-mail: {perez,mavi}@lcc.uma.es

J.Z. Hernández,
A.M. García-Serrano
Universidad Politécnica de Madrid
Dpt. of Artificial Intelligence
e-mail: {phernan,agarcia}@dia.fi.upm.es

J.M. Maseda
LABEIN, Technological Centre
Information Society Unit
e-mail: maseda@labein.es

SCATEAgent: Context-Aware Software Agents for Multi-Modal Travel

Min Yin and Martin Griss

Abstract. In this paper we describe our research on an agent-based intelligent, flexible, and context-aware multi-modal traveler information system, SCATEAgent[1]. The work targets the representation and manipulation of core user, preferences, and context models which facilitate highly-customized and adaptable agents playing key roles in an agent-based Advanced Traveler Information System (ATIS). This ATIS finds and fuses information from multiple sources on routes, congestion, incidents, weather, alternative transit modes and schedules, provides proactive real-time updates and context-specific guidance as the user travels, and is vigilant about impacting events. A highly intelligent, personalized, and user-friendly assistant to the traveler, especially in case of transitions among different travel modes, SCATEAgent will promote effective individual traveling and also help to smooth the transportation load in general. The expected contributions include the design of user, preference, context and travel ontologies; user, context, task models based on these ontologies, a set of representations to drive agent behavior and communication, and a compatible integration of rules, machine learning, information retrieval, and semantic web. Currently, we have completed the first stage of our research, producing first pass ontologies and models, an initial prototype of a small-scale test-bed incorporating GPS-enabled cell-phones, and multiple mechanisms for agents to handle and adapt to such models and travel related events.

1. Introduction

Travelers using their personal or shared vehicles, or public transit options, are facing increasing congestion, more incidents, and longer delays. Most surface travel is done in single occupant vehicles (SOV), on primary highways and some arterial roads. Most of the driving is done by commuters on their regular routes, who often spend over 50% of their time delayed by traffic [Telcontar]. Too few drivers consider alternate routes, adjust schedules, or select alternate modes of transit. Effective Transportation Systems Management is critical for supporting the nation's continued population and economic growth. The evolution of Intelligent Transportation Systems[Adler & Blue, 1998] has shown that effective solutions must combine both supply-side traffic management techniques to control traffic flow and demand-side initiatives to encourage more efficient travel choice behavior. The challenge is to find and implement solutions that

[1] SCATE: Santa Cruz Agent Technology and Environment.

achieve an efficient reallocation of network capacity over time and space without seriously violating any individual user's preferences for mode, routing, departure, and/or arrival time. Additional value would be to recommend ancillary services appropriate to the user's priorities, location and needs.

Our work focuses on the demand side, seeking to improve the usability and quality of the advice given by an Advanced Traveler Information System (ATIS). In a Department of Transportation study and analysis of a survey of ATIS needs and customer types [Lapin 2000; Lapin 2000a], Lapin identifies several types of ATIS customers and their differing needs. She notes distinct gender related preferences, different levels of comfort with technology, and interests in differing kinds of ATIS advice. This analysis highlights the opportunity for a more dynamic, flexible ATIS that adapts its behavior to the specific user's needs, preferences, and activity at the time of planning and especially while executing a trip. For Mobile ATIS Lapin states "Drivers want reliable, accurate, relevant traffic information while driving. For many trips, the traffic information provided pre-trip is outdated when the driver reaches a potential route diversion decision. This is where the greatest demand for ATIS exists. Drivers recognize the safety challenge of delivering information to a driver, and most respondents expressed safety concerns about mobile phone use when driving. Nevertheless, drivers wish they could press an ATIS button when approaching congestion or a route choice and quickly know which route would be least congested."

Cell-phones are a primary potential tool for interacting with an ATIS, particularly if the interaction could be highly personalized, mostly hands-free, and very context-specific in order to reduce distraction and danger. In 2000, there were over 120 million cell phones in use in the United States, at least 500,000 drivers were using cell phones at any given time during the day, and more than 50% of all drivers had wireless phones in their vehicle at all times [Cellphone]. Tomorrow's ubiquitous computing environment will be filled with even more intelligent, location-aware devices: on your person, in your office, car, bus, train and home, on the street, in stores, movies, museums and restaurants. Future devices and systems will exhibit emergent intelligence - the seeming ability to act intelligently by integrating a variety of information and resources pertinent to the user's current situation.

Nowadays agent technologies have seen a lot of use to support mobile users [Griss et al, 2002; Goni et al, 2001; Blechschmitt & Strodecke, 2002]. The autonomic, intelligent and collaborative nature of agents play a key role in mobile systems and applications, such as personal email systems, personal news systems, wireless web applications, etc. The key idea is to apply software agent technology to support these mobile users. Intelligent agent technology promises to provide new levels of highly personalized, adaptive, context-aware functionality. The combination of wireless mobile appliances and intelligent agents using physical context to mediate the interaction with many services would greatly enhance the user's ability to plan, select and execute travel, and to choose appropriate travel modalities, routes, schedules, and relevant ancillary services.

SCATEAgent is a set of agents, utilities, and components developed to implement a portion of this vision. It leverages agent technology to provide personalized, dynamic and context-aware services to the mobile traveler. Our key contributions are in the areas

of the specific ontologies and models we have designed to represent trips, services, and traveler preferences to enable intelligent multi-modal travel support. We have also novel approaches to the architecture of the agent-based ATIS and to the way we have integrated the rules and the agents.

The rest of the paper is organized as follows. Section 2 gives a brief description of the architecture of SCATEAgent and the integrated agent-based information processing system. Section 3 describes the ontologies and models. Section 4 introduces related technologies. Section 5 describes our current working status, briefly introduces the SCATEAgent test-bed and summarizes plans for evaluation of the system. Finally, Section 6 concludes with a discussion of future work.

2. System Architecture

Figure 1 shows the SCATEAgent system architecture. There are two sets of agents, personal agents and shared public service agents. The personal agents include Personal Travel Agent (PTA), Personal Location Agent (PLA), and Calendar Agent. These agents belong to each individual. They maintain and manage the personal preferences and personal activities. For example, PTA is responsible for individual trip planning and monitoring and it utilizes and manages all travel related personal preferences such as preferred transportation types, route chosen history, special travel requirements due to disability or other medical situations, etc. PLA works closely with PTA. It keeps track of the traveler's location and reports it constantly to the PTA so that PTA is able to monitor the trip and propose prompt, accurate and relevant information to the traveler or prepare to re-plan the trip when it detects unexpected situation such as a delay. PLA maintains location related personal preferences such as points of interests (POI), higher-level models of location ("at home", "at work", etc.), location history, location security, and location authorization information. The location information provided by the PLA can also be used by other personal agents. For example, notification agent can consult PLA and choose appropriate notification channel according to the user's current location. On the other hand, before the calendar agent inserts an event entry in its schedule, it may check with PTA to make sure that the location of the proposed event is accessible by the personal at the proposed time slot.

For PTA to plan the trip accurately and precisely, it also needs to get information, such as maps, weather, traffic, and public transportation schedules from external services. These services are provided by shared public service agents. One great advantage of using agent technology comes from its loosely coupled architecture. For example, there can be multiple map agents; each provides map service from different source or with different granularity. This way, we can prioritize them according to factors such as reliability, cost, promptness, etc. If one service is not available for some reason, the agent architecture will automatically find the next service agent that provides the same service using a facilitator agent. This is also true for the route agents, which provide routing services to the PTA. There might be different route agents, some are simple, fast, but less accurate; some are more precise and sophisticated, but not free.

PTA may choose to use one of the route agents, or it may choose to consult more than one and then select a route among those returned by different route agents. Such decisions are made according to traveler's concerns, preferences, and the context of the trip.

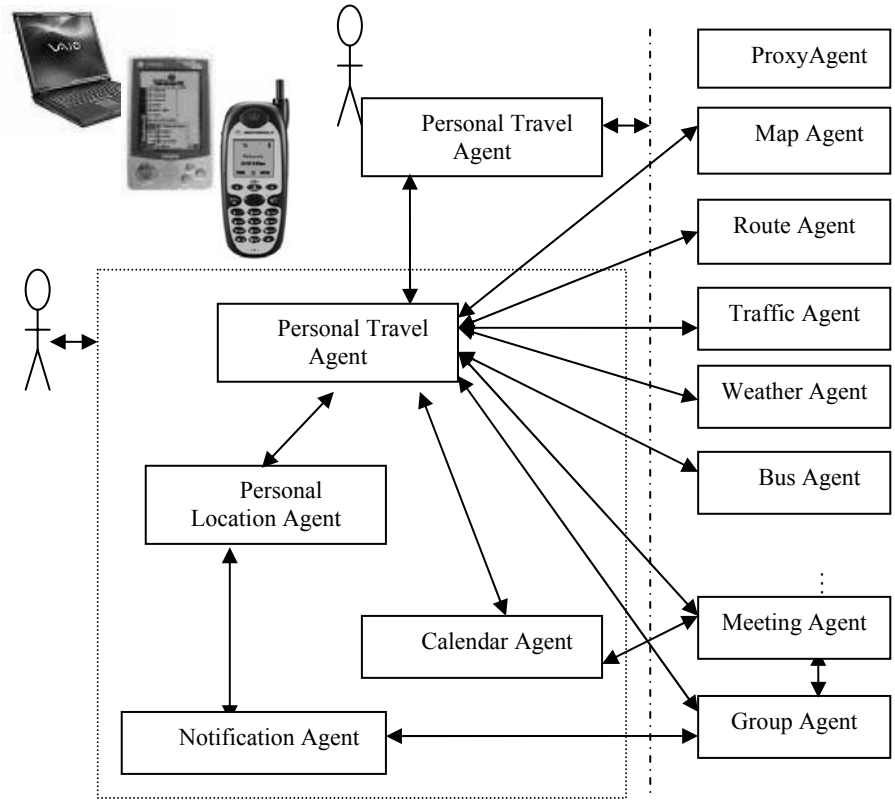


Fig. 1. SCATEAgent System Architecture

The PTA plans a trip in a goal-directed and task-driven manner. For example, when the Meeting Agent has planned a meeting that requires the physical attendance of all or part of the participants, it will send a task, or a travel request, to those participants. Upon the recipient of the request, the PTA will then check the person's current location with the PLA. Obviously there will be no need to plan a trip if the meeting will be held in the same building. But if moderate travel is required, the PTA will react and function accordingly. Similarly, a task can be sent to a group of people through the Group Agent when all the individuals in such group are supposed to travel to the same destination.

Finally, the interaction between human and agents is done through mobile or stationary devices such as cell phones, PDAs, laptops, or even a public kiosk. It is important to have a Proxy Agent to monitor and manage such human to agent interaction. Actually, since the traveler is always on the move, it is important to have a

uniform human-agent interface and a set of interaction agents to maintain a continuous seamless dialogue between the user and different devices.

In the architecture presented in Figure 1, it is essential to develop a set of ontologies understood and agreed by all agents. The user preferences, context, and tasks will be represented by corresponding models based on those ontologies. More details will be given in the following sections.

3. Ontologies and Models

In SCATEAgent, we need a coherent and effective way of creating/maintaining/ensuring consistency of ontologies, and using several preference, information and context models to drive complex decisions.

Generally, agents communicate through asynchronous message passing, using a loosely-formatted, text-based communication language. Messages include a header (to, from, etc.), an ontology, a message type, an expected message protocol, and a variable-length body. FIPA specifies a dozen or so high-level message types, such as Notify, Query, Inform, etc. In addition to requests to perform, communication languages allow agents to exchange information such as commitments, goals, beliefs, and intentions.

Different agents (Weather Agent, Bus Agent, Traffic Agent, etc) of the SCATEAgent test-bed need to access a variety of information and services, such as maps, schedules, incidents, and calendars from multiple sources, with heterogeneous interfaces. Some are proprietary applications (e.g., OutLook, Palm desktop), while others are databases, FTP accessible services (e.g., PeMS), HTTP accessible web-sites, and recently some WSDL described web services (e.g. Microsoft MapPoint). This means that shared service agents, or service wrappers will need to convert these idiosyncratic interfaces into a standardized agent-compatible format and harmonize two different view points on ontology and knowledge representation: the web services view uses XML, RDF, semantic web (JENA) and WSDL languages, while the AI/agents style uses the more logic oriented KIF, KQML/ACL, and DAML/OIL approaches. A different approach is to totally discard standard structured agent languages such as FIPA-SL and promote relatively weak ontologies through a Semantic Web standard such as RDF [Grimnes et al, 2003]. The advantages of such an approach are that it costs very little to integrate existing web services into the agent system and it is relatively easy to reuse weak ontologies elsewhere when necessary. But we suspect it will complicate the agent negotiation and learning abilities.

We believe for our SCATEAgent system, the solution lies between the two extremes. The ideal ontologies and models will leverage both sides appropriately and are tailored to meet the specific ATIS needs. After an extensive search no such ontologies and models could be found in the literature. Other desirable features include ensuring agent autonomy, providing context-awareness, controlling the flow of ACL messages between agents, and enhancing flexibility and scalability.

```
f-1 (person (first_name Kate) (last_name Smith) (gender female) (age 35))
f-2 (transportation_type (type_name private_car))
f-3 (transportation_type (type_name rental_car))
f-4 (transportation_type (type_name taxi))
f-5 (transportation_type (type_name light_trail))
f-6 (traveler (person <FACT-1>)(available_transportation_type <FACT-3> <FACT-4>))
f-7 (address (address_name SFO) (gps_latitude 37.125) (gps_longitude -122)
        (available_trans_type <FACT-2> <FACT-3> <FACT-4> <FACT-5>))
f-8 (address (address_name UCSC_Entrance)
        (gps_latitude 36.977) (gps_longitude -122.052)
        (available_trans_type <FACT-2> <FACT-3> <FACT-4>))
f-10 (individual_current_location (person <FACT-1> (current_location <FACT-7>))
f-11 (task (person Kate) (destination_address <FACT-8>) (buget MID))
```

(a) Some Travel Related Facts

```
(defrule trip_planning_by_rental
        (task (person ?X) (destination_address ?Y) (budget MID|HIGH))
        (individual_current_location (person ?X) (current_location ?Z))
        ?type_rental <- (transportation_type (type_name rental_car))
        (traveler (person ?X) (available_transportation_type $? rental_car $?))
        (not (traveler (person ?X) (available_transportation_type $? private_car $?)))
        =>
        (bind $?a (get_multislot ?Z available_transportation_type))
        (bind $?b (get_multislot ?Y available_transportation_type))
        (if (and (car_accessible $?a) (car_accessible $?b))
           then (bind $?seg (assert(trip-segment (start_point ?Z) (end_point ?Y)
                        (transportation_type_id ?type_rental))))
           (assert(trip (person ?X) (trip-segment ?seg)))
        )
)
```

(b) A Simple Trip Planning Rule

```
Class Trip  {
        Traveler  traveler;
        SCATETime start_time;
        Address start_location;
      ArrayList segments; //array of RouteSegment
}
```

```
Class RouteSegment {
        Address  end_point;
  TransType type;
  double length;
  SCATETime end_time;
}
```

(c) Corresponding Java classes

Fig. 2. A Simple Example of Trip Planning

The travel ontologies and models used in SCATEAgent are described as objects, object attributes, and object relations. For example, “POI” and “Abstract-address” are two different kinds of address. “POI” is the personalized point of interest such as home, office, school, etc, while “Abstract-address” represents one special place such as “coffee”, “bus stop”, etc. Traveler model refers to person model for the general information of the traveler. Commuters, elder-traveler, and disabled-travelers are all different kinds of travelers; each having their own set of characteristics. A trip plan can

be generated according to a travel task, the traveler's current location, the traveler's personal preferences, and available transportation tools.

SCATEAgent is constructed using the JADE platform with the JESS rule engine [JESS]. Currently the models are represented with a combination of JESS templates, facts and Java objects. There are several advantages of using JESS facts as ontology instances. First, agents can easily adjust their behaviors and demonstrate intelligence and flexibility by directly applying JESS rules to the elements of the user model and context models. Adding, deleting, or modifying JESS facts and rules is of lower cost as compared with changing the states of a state machine so both the models and the rules can evolve gradually with increasing complexity. Second, a JESS fact can be easily wrapped with an ACL message and be sent among the agents. Finally, JESS facts and Java objects can also be converted conveniently into each other so that SCATEAgent can exchange data with external services, especially web services. For example, a "trip" fact and the "trip-segment" facts it refers to can be converted into a java object of the "scate.travel.Trip" class, which is in turn compatible with other data formats, such as the SOAP route object provided by Microsoft MapPoint map and routing services, through intermediate utilities.

Figure 2 gives a very simple example of how a trip is planned according to some given facts. Figure 2 (a) is some traveler related facts, including traveler, addresses, transportation types and task. For the sake of simplicity, the time slots are omitted. Figure 2 (b) gives a simple trip planning rule using the facts given in (a). It claims that the traveler can travel by a rental car if his/her personal car is not available, he/she does not exclude rental car from the candidate transportation means, the trip budget is "mid" or "high", and both the source and the destination can be reached by car. Figure 2 (c) gives the class structure of the Java object compatible to the facts generated by the rule in Figure 2 (b). Notice that rule in Figure 2 (c) only chooses a possible transportation tools without specifying the route details. Such routing task will be then delegated to Route Agent, such as MapPoint Route Agent. Figure 2 (a) and Figure 2 (c) show how ontologies are expressed as JESS facts, JESS fact slots, and Java class fields. Other approaches, such as Protégé, are also being explored so that the ontologies developed can be generated compatibly to DAML/OIL or OWL, yet still enacted using Jess.

It can be seen that both the context (travel budget, personal car availability) and the user preferences (excluded transportation means) play an important role in the selection of transportation means. It is also relatively easy to add, delete, or modify the rules, for example, to change the budget threshold from "mid or high" to "high", etc. One or more possible routes might be returned by one or more Route Agents. SCATEAgent will then query the Traffic Agent and Weather Agent for the traffic and weather situation of the interested route segment. The results will be transferred as JESS facts and trigger relevant rules to select the best route for the current traveler under the current travel context.

The process of choosing public transportation means, including public transits, is relatively easy. Most public transportation schedules are pre-stored in the knowledge base as facts and are updated on a regular base. A route can be generated by applying rules that match both the transit address and the transit time of different public transportation schedules.

Once an initial route is generated and the traveler is on his/her way, the PTA monitors the trip using the updated location information reported by the PLA. Each new location inserts a new fact into the knowledge base and any possible delay or change of the plan will be caught by corresponding rules which will consequently trigger the process of notifying the user and generating new travel plans according to the unexpected situation.

Besides the primary travel task, other smaller tasks or tasks of lower priority can come into play via adding relevant rules. For example, tasks listed in the personal to-do list, such as picking up laundry or visiting the dentist, can affect the travel route, the time, and the transportation means.

4. Related Work and Technologies

There is considerable relevant work in the areas of software agent technology, distributed AI, context-aware computing, and applications of agent technology to traffic management systems. We will only mention a few of the most important to our work. Many of those efforts provide a base for our research, but are not directly comparable because we focus on an individually usable and personalized ATIS, while other agent-based traffic applications focus on simulation or traffic management.

Intelligent multi-agent systems are seeing increasing use in research and products associated with complex adaptive systems. Agent technology has matured considerably via standards by the Foundation for Intelligent Physical Agents (www.fipa.org), open source agent toolkits, such as JADE [JADE], large-scale agent interoperability tests (www.agentcities.org), and numerous applications of agents in research, advanced development, and products [Griss 2000]. Agents have been applied to enterprise integration and supply chain management, planning, scheduling, execution control, materials handling, and inventory management. In our work, we use the JADE system, building on a prior integration with the JBOSS J2EE server [BlueJADE; Cowan & Griss, 2002], developed to increase large-scale agent scalability and manageability [Griss & Kessler, 2002].

Applications of Agents to Traffic

Multi-agent systems are well-suited for use in intelligent transportation systems. [Burmeister et al, 1997] states that agents can work well in complex systems in which:

- the problem domain is geographically distributed,
- subsystems exist in a dynamic environment, and
- subsystems need to be adaptive and interact more flexibly.

According to Adler and Blue, efforts to provide assistance to drivers using information technology began in the 1950's [Adler & Blue, 1998]. These early systems involved both traffic surveillance and real-time dissemination of advisory notices, primarily through the use of visual displays. The development of in-vehicle route

guidance systems (IVRGS) began in the 1970's [Rosen 1970]. IVRGSs, including GPS-enabled cell-phones, have evolved to include real-time updates of traffic incidents and congestion. In the last few years, the US Department of Transportation has developed a National Intelligent Transportation System Architecture designed to provide "a common framework for planning, defining, and integrating intelligent transportation systems."[NITSA]

Today, large amounts of data on traffic, weather, dynamically annotated maps, and driving directions are made available on the Internet by Caltrains, the Department of Transport, NOAA, and many others. Research groups and product companies are collecting and consolidating this information and using it to offer real-time advice to travelers, either during trip planning or en route [FreeWay2000; PeMS; TrafficDodger; Telcontar; WestwoodOne]. Real-time updates can be sent via web, email, pager, cell-phone, or other wireless technology [MobilityTechnologies; SFBayTraffic]. Pre-trip planning can include maps showing incidents and current traffic flows or an annotated "best-route" with estimated travel time based on current congestion and incident information [TrafficDodger]. Some also offer a degree of personalization [Iteris; SFBayTraffic]. In addition to GPS-enabled in-vehicle systems, laptops, PDA's and cell-phones can now offer maps and turn-by-turn directions [ViaMoto; TeleNav]. Motorola's ViaMoto service uses GPS-enabled i88s phones to offer location-aware and navigation services to ATIS and fleet management providers. GPS-equipped public transit vehicles can send notifications to web-pages, a user's cell-phone, or PDA when the next bus is due to arrive [CoolTown; Kindberg 2002; NextBus]. All these services provide a solid foundation for SCATEAgent. Currently we have a bus agent and two route agents, more service agents will be developed as the system evolves.

Distributed agents have been used in a variety of surface and air traffic simulation and management systems [Andersson & Rnnbom, 1999; Appiani et al, 1999; Erol 1999; Fernandes & Oliverira , 1999; Hidas 2002; Katwijk 2002; Roozemond 1996; Wojcik 2000]. Compared to traditional micro-simulators such as PARAMICS, distributed agent-based simulators offer increased flexibility, the ability to model driver behavior and near real-time simulation and prediction. Agents, neural networks, and other AI techniques have been applied to incident analysis, and to simulate features and usability of an ATIS [Das 1999]. CityTraffic combines real-time sensors with a high-performance distributed agent system for real-time prediction [CityTraffic]. Intelligent Automation, Inc. has used agents for several real-world traffic simulators and controllers, allowing some experimentation with driver behavior [IAI].

The Oct-Dec 2002 issue of Transportation Research Part C was devoted to the growing use of agents as an emerging technology for transportation. While most of the papers were on simulation or ATMS coordination and analysis, Adler and Blue [Adler & Blue, 2002] described a cooperative multi-agent transportation management and route guidance system linking ATMS and some ATIS capabilities. In an earlier paper they make a compelling case for the role of AI techniques in "3rd generation Intelligent Traveler Information Systems," highlighting the need for a more personalized and adaptive ATIS [Adler & Blue, 1998]. They stressed the key role of machine learning and other AI techniques in delivering a more pleasant and effective experience to travelers while planning a trip or en route; however, they do not mention at all of agents

as a vehicle to deliver the needed capability in a dynamic, incremental and distributed fashion.

Context- and Location-Aware Systems

There are numerous approaches through GPS, wireless, RFID, infra-red and Bluetooth, and other peer-to-peer and ubiquitous computing technology to determining the physical location of appliances, and the availability and presence of other appliances and local services in the neighborhood [CoolTown].

Future cell phones will do much more than make calls and keep phone books. Perhaps the greatest potential that they hold will require sensing the environment, interacting with nearby devices, and exhibiting opportunistic behavior specific to current user goals and context. The basic idea of seeking and discovering the ability of neighboring devices or services is similar to that of SUN's peer-to-peer Jini [Jini] or UUDI/XML-based W3C web services. In Jini the advertising of abilities uses an object's API of public method interfaces, while in web-services a UDDI registry contains a higher-level WSDL description in XML.

Currently we are using the GPS-enabled cell phones to determine the traveler's location. Future work will introduce other location-aware devices, most probably infra-red and Bluetooth, and use rule-based goals and abilities to enable the matching and negotiation to occur in a much richer and more semantically meaningful way.

Agent Intelligence and Coordination

Some agent toolkits provide some useful rule-based and goal-based capabilities. For example, the Zeus agent toolkit [Zeus] has a built in problem-solver, planner, and rule-based goal/fact system, as well as a protocol-based negotiation system. Rules and workflows can be used to define how agents with different roles interact in a flexible, yet constrained manner to accomplish a collaborative task [Griss 2000; Chen et al, 2000; Sutton & Osterweil, 1997]. Others have used UML state machine, interaction diagrams, and (colored) Petri-nets to provide more precise specification and control of agent interactions. In our work, we are integrating rule-based and state-machine based approaches, where the rules provide flexible handling of transitions in the state machines, and the state machines provide structured context for the rules and goals.

There is significant work on the development, extension, and application of UML-based modeling for agent systems, AUML [Odell 2001; Bauer et al, 2001], and also the development and evaluation of complete agent system development methods, such as AOP [Shoham 1993], GAIA [Wooldridge et al, 2000], and Tropos [Guinchiglia et al, 2003]. One of the MIT Oxygen projects proposes to formalize goals as a language construct – an interesting idea, but little has yet been published [Ward et al, 2002]. The development of agent society patterns in UML [Kendall 1999] has implications for analysis and design, and leads to aspect-oriented implementation.

5. Status and Evaluation

We are in the process of constructing and revising a small multi-agent ATIS test-bed. This experimental platform will be used to tune and evaluate the improved models and rule-based technology. We have completed the first stage of constructing models and ontologies and built our test-bed to run the first prototype of SCATEAgent. The system uses the JADE agent framework [JADE] and JESS rule system [JESS], running on the BlueJADE J2EE server environment [BlueJADE]. BlueJADE wraps JADE agents as restart-able J2EE services, providing an increased measure of reliability and manageability, as well as future session-based authentication and load balancing.

Currently, the test-bed uses several Java-based GPS cell phones (Motorola i58sr), interacting with personal assistants, location and notification agents, which access local bus schedules, maps, routing services, and traffic information in the Santa Cruz area. One of the more challenging tasks was interfacing the phone with the agents. The J2ME program running on the phone will periodically poll an IP socket exposed by a specific proxy phone agent to exchange encoded messages and GPS location with other agents. Agents representing drivers and travel information services will help travelers plan a route and drive it or take public transport. The personal agents interact with other agents, with calendars and emails, and use VoiceXML, CallXML, or the phone proxy to notify the phones.

The current test-bed PTA is customized via a simple profile (name, home and office address, phone and cell phone numbers) and some simple preference rules ("I prefer to take the bus, unless I have an urgent meeting later"; "I am willing to delay my trip to avoid traffic."). We are in the process of integrating the richer models described in this paper into the system.

Currently SCATEAgent runs under the following small test scenario:

- Start at work, and issue the command "Go home".
- A transportation method will be selected by PTA according to the user's preferences and it will be proposed to the user.
- Upon the confirmation of the user, the PTA will select the appropriate driving route or bus route.
- The notification agent will notify the user to leave when the target bus is approaching.
- The PLA will monitor the trip, notify the user of the next transit or possible bus delay

A future target test scenario incorporating results of the proposed research is something like:

- Start at work, and issue the command "Go home".
- Some users will get advice on which bus to take, from which stop, and when. Others will be reminded where they parked their car.
- Directions will be offered for the recommended route, adjusting for anticipated traffic at that time.
- Some users will be reminded that they should pick up dinner before coming home, or that they should pick up the kids at ballet, and the route and directions adjusted for that activity.

- The agents will display some vigilance related to congestion, incidents, or changes in schedule or plans en route. ("e.g., the stop lights at Mission are out, take the alternate route; or location of the meeting with friend just changed").

The results of the first scenario test are promising and we are enhancing the system gradually. An initial evaluation will be done by the small group of researchers using a test harness for the test-bed. Simulation components will feed in events to simulate new routes, new personal goals or preferences, load and incidents, and weather. As the system becomes tuned and new components are added, we will incrementally replace the simulated components, and have the researchers or several undergraduates play the roles of travelers. We are expecting the system to assist the traveler accomplish travel task more smoothly with less time and cost.

6. Conclusions and Future work

In this paper we describe the plan and progress of our research on building SCATEAgent, an agent-based context-aware ATIS environment for multi-modal travel. The autonomic, collaborative, and intelligent nature of software agents helps to bring the very much desired and needed dynamic, flexible, and highly personalized characters to current ATIS systems, which are relatively simple and isolated. One key issue is to build consistent ontologies over different agents and external services and models amiable to manipulation through rule-based systems, state machines, machine learning tools, or other intelligent elements.

A challenge is how much intelligence is needed to get a useful amount of benefits from adaptive, flexible goal-directed processing [VanHilst & Griss, 2003]. A key strategy is to initially avoid very deep AI and complex natural language understanding, and look for less intelligent agents. Currently we are combining rule-based techniques (JESS) with our state machine extensions to JADE [Griss et al, 2003] to make it easier to handle complex negotiation protocols. Currently, the state machines manage the protocols used in the negotiation between agents, such as agreeing on a common time and location for several travelers to meet. Other negotiations involve (re-)checking with the traveler should the priorities, plans, or schedules change.

We are investigating mechanisms to further integrate a combination of rule-based, machine learning, information retrieval and data mining techniques into a coherent "intelligence" toolkit, consistent with the models described above, and convenient for use inside the JADE agents. Our overall approach is to structure knowledge and rules within each agent to support a simplified form of goal-directed computation. Goals are decomposed into smaller goals and assigned to specific components in the system. Each system component agent performs behaviors in pursuit of its goals. Large systems with complex behaviors are formed by collections of narrowly focused agents, each acting in its own self-interest.

Another major challenge is to develop a uniform agent human interface. Since the traveler is always on the move and will encounter different devices and/or different interaction modalities (speech, key-stroke, stylus, etc) during the trip, we need to

develop a framework that helps human agent interface to roam smoothly and seamlessly from one device to another, and/or one modality to another. We have designed such a framework where dialogue agent, device agent, proxy agent, and other utility agents collaborate to maintain a continuous dialogue and manage or trim the dialogue according to the interaction context, task, and user preferences.
Finally, we plan to scale the system to a larger pilot test, used by some of the researchers on a regular basis.

Acknowledgements

This work was produced with research grant no. NSF CMS-0339251 SC 20030921 funded by NSF and the Department of Transportation Federal Highway Administration, and with support from HP Laboratories.

References

[Adler & Blue, 1998] Adler, J. & Blue, V., Toward the design of Intelligent Traveler Information Systems, cooperative multi-agent transportation management and route guidance system, Transportation Research, 6C(3), June 1998, 157-172.

[Adler & Blue, 2002] Adler, J. & Blue, V., A cooperative multi-agent transportation management and route guidance system, Transportation Research, 10C(5-6), Oct-Dec 2002, 433-454.

[Andersson & Rnnbom, 1999] Andersson, L. & Rnnbom, A., Intelligent Agents - A New Technology for Future Distributed Sensor System Informatics 1999. http://citeseer.nj.nec.com/andersson99intelligent.html

[Appiani et al, 1999] Appiani, E., Martelli, Maurizio, and Mascardi, Viviana, A Multi-Agent Approach to Vehicle Monitoring in Motorway, Technical Report DISI TR-00-13, 2000, . http://www.disi.unige.it/research/Logic_programming/agent_pubs.html

[Bauer et al, 2001] Bauer, B. Müller, P., Odell, J. Agent UML: A Formalism for Specifying Multi-agent Interaction in Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, 2001, 91-103

[Blechschmitt & Strodecke, 2002] Blechschmitt, E & Strodecke, C., An architecture to provide adaptive, synchronized and multimodal human computer interaction, Proceedings of the Tenth ACM International Conference on Multimedia,Juan-les-Pins,France, December,2002

[BlueJADE]http://www.blueJADE.net/public_resources/director/

[Burmeister et al, 1997] B. Burmeister, A. Haddadi, and G. Matylis. Application of multi-agent systems in traffic and transportation.In IEE Proceedings of Software Engineering 97,51-60.

[Cellphone] Facts about cellphone. http://www.insweb.com/learningcenter/special-reports/cellphones/facts.htm

[Chen et al, 2000] Chen, Q., Hsu, M., Dayal, U., Griss, M.: Multi-Agent Cooperation, Dynamic Workflow and XML for E-Commerce Automation, Autonomous Agents 2000, Barcelona, June (2000), 255-256.

[CityTraffic] CityTraffic – An integrated system for traffic planning, management and information in high-density urban areas http://www.citytraffic.de/ct_eng.html See also http://www.ercim.org/publication/Ercim_News/enw46/muehlen.html

[CoolTown] CoolTown – HP Laboratories-developed, open source technology for context-aware web-based applications. http://www.cooltown.hp.com/cooltownhome/index.asp

[Cowan & Griss, 2002] Cowan, D. and Griss, M., Making Software Agent Technology Available to Enterprise Applications, 1st International Workshop on "Challenges in Open Agent Systems", AAMAS'02, Bologna, Italy, July 2002

[Das 1999] Das, S., Bowles, BA. Houghland, CR., Hunn, SJ., Zhang, Y. A knowledge based model of traffic behavior in freeways, Proceedings of the 1999 ACM Symposium on Applied Computing, , San Antonio, Texas, United States, 1999, 14-18

[Erol 1999] Erol, K., Levy, R., & Wentworth, J., Application of Agent Technology to Traffic Simulation, Federal Highway Adminstration, http://www.tfhrc.gov/advanc/agent.htm

[Fernandes & Oliverira, 1999] Fernandes, JM. & Oliveira, E., TraMas: Traffic Control through Behaviour-based Multi-Agent System, Proceedings of the Fourth International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM99), London, April 1999. 457-458, See http://www.ieeta.pt/~jfernan/tramas/

[FreeWay2000] http://www.freeway2000.com/

[Goni et al, 2001] Goni, A., A. Illarramendi, A., E. Mena, E., Y. Villate, Y., & J. Rodriguez, J. Antarctica: A multi-agent system for internet data services in a wireless computing framework. In NSF Workshop on an Infrastructure for Mobile and Wireless Systems Scottsdale, Arizona (USA), October 2001.

[Grimnes et al, 2003] Grimnes, G. A. A., Charlmers, S., Edwards, P., Preece, A., GraniteNights – A Multi-Agent Visit Scheduler Utilizing Semantic Web Technology, http://www.csd.abdn.ac.uk/~ggrimnes/pubs/GraniteNights.pdf

[Griss & Kessler, 2002] Griss, M. and Kessler, RR., Achieving the Promise of Reuse with Agent Components. First International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, ICSE, May 2002, 1-9.

[Griss 2000] Griss, M., My Agent Will Call Your Agent, But Will It Respond, Software Development Magazine, Feb, 2000.

[Griss et al, 2002] Griss, M., Letsinger, R., Cowan, D. Sayers, C., VanHilst, M., Kessler, R., CoolAgent: Intelligent Digital Assistants for Mobile Professionals – Phase 1 Retrospecitve. HP Laboratories technical report HPL-2002-55(R), July 2002.

[Griss et al, 2003]Griss, M., Fonseca,S., Cowan, D., and Kessler, R., Using UML State Machines Models for More Precise and Flexible JADE Agent Behaviors, AAMAS AOSE workshop, Bologna, Italy, July 2002.

[Guinchiglia et al, 2003] Guinchiglia, F., Myopoulos, J. and Perini, A., The Tropos Software Development Methodology: Process, Models and Diagrams. Agent-Oriented Software Engineering III, Bologna, Springer LNCS 2585. July 2002, 162-173.

[Hidas 2002] Hidas, P., Vehicle Interactions by Intelligent Agents, University of New South Wales, June 2002. See http://www.civeng.unsw.edu.au/staff/hidas.p/sitras.htm

[IAI] Intelligent Automation, Inc., agent-based traffic simulator using OpenCybele and AASIM. http://www.i-a-i.com

[Iteris] Iteris offers real time maps, has a concept of Personalized Traveler Information , is one of two companies selected to develop a national ITS architecture. http://www.iteris.com/.

[JADE] Bellifemine, F., Poggi, A., Rimassi, G.: JADE: A FIPA-Compliant agent framework, Proc. Practical Applications of Intelligent Agents and Multi-Agents, April (1999), 97-108;

[JESS] Jess™ A forward and backward chaining rule system, written in JAVA. http://herzberg.ca.sandia.gov/jess/

[Jini] Jini Network technology - http://wwws.sun.com/software/jini/

[Katwijk 2002] van Katwijk, R., and van Konigbruggen, P., Coordination of traffic management instruments using agent technology, Transportation Research Part C 10 (2002), Elsevier, 455-471

[Kendall 1999] Kendall, EA. Role Model Designs and Implementations with Aspect-oriented Programming, in Proc. of OOPSLA 99, Oct, Denver, Co., ACM SIGPLAN, 353- 369.

[Kindberg 2002] Kindberg, T., et al., People, Places, Things: Web Presence for the Real World, HP Labs tech report, HPL-2001-279, 2001,. In proceedings WMCSA2000. ACM MONET (Mobile Networks & Applications) Vol. 7, No. 5 (October 2002).

[Lapin 2000] Lapin, J., Advanced Traveller Information Service(ATiS): Who are ATIS customers, US DOT, Jan 2000, #12885

[Lapin 2002a] Lapin, J., Advanced Traveller Information Service(ATiS): What do ATIS Customers Want, US DOT, Jan 2000, #12884

[MobilityTechnologies] MobilityTechnologies owns http://www.traffic.com - provide real-time traffic information via TrafficPulse, digital sensors, real-time updates on web, email.

[NextBus] Nextbus – GPS enabled busses in San Francisco http://www.nextbus.com

[NITSA] US Department of Transport, National ITS Architecture, http://www.its.dot.gov/arch/arch.htm

[Odell 2001] Odell, J., Van Dyke Parunak, H., Bauer, B. Representing Agent Interaction Protocols in UML, in Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, 2001, 121-140

[PeMS] Freeway Performance Measurement Project Web Site, funded by PATH - http://pems.eecs.berkeley.edu/login.phtml

[Roozemond 1996] Roozemond, D.A., Intelligent traffic management and urban traffic control based on autonomous objects; in Proceedings of the sixth Artificial intelligence, simulation, and planning in high autonomy systems international conference), IEEE, 1996. (ftp://cci.ct.tudelft.nl/DR_AISP.pdf)

[Rosen 1970] Rosen, DA., Mammano, FJ., & Favort, R. An Electronic Route Guidance System for Highway Vehicles, IEEE Transactions on Vehicular Technology, VT-19, 1970.

[SFBayTraffic] A personalized alerts service can relay important traffic information to you on your cellphone, pager (and any device that can receive email) when you are on the road, or about to get on it, using a profile. http://www.sfbaytraffic.com

[Shoham 1993] Shoham, Y., Agent-Oriented Programming, Artificial Intelligence, Vol. 60, No. 1, 1993, 139-159

[Sutton & Osterweil, 1997] Sutton, SS. And Osterweil, LJ., The design of a next generation process programming language, Proceedings of ESAC-6 and FSE-5, Springer Verlag, 1997.

[Telcontar] TelContar http://www.telcontar.com/ is a traffic manager product, written in Java, uses Teleatlas SmartRoute and TrafficCast.

[TeleNav] A Nextel service that uses GPS enabled cell-phones to provide-turn-by-turn navigation information in a cell-phone. Is voice enabled, does automatic reroute updates http://www.telenav.net/telenavnextel

[TrafficDodger] Astartup that provides a MapQuest like interface, over TeleAtlas data, with some incident, event and time travel annotations. http://www.trafficdodger.com/ . Will annotate best route, give estimated travel time, etc.

[VanHilst & Griss, 2003] VanHilst, M. & Griss, M. Goal-Directed Design for Proactive and Intelligent Device Collaboration., KES 2003: 841-848.

[ViaMoto] Motorola's location-ware and traffic services. http://www.motorola.com/mobileinternet/

[Ward et al, 2002] Ward, A., Terman, C., Saif, U., Goal Oriented System Semantics, March 2002 http://cag.www.lcs.mit.edu/~umar/publications/Goals_abstract.pdf

[WestwoodOne] Westwood One (Metro Networks, SmartRoute). SmartRoute Systems provides the very latest in local traffic, news, sports and weather information to wireless, Internet, in-vehicle navigation systems and voice portal customers, in multiple formatshttp://www.metronetworks.com/

[Wojcik 2000] Wojcik, LA., Campbell, K.C., Cooper W.W., & Greenbaum, DP, Modeling Distributed Human Decision-Making in Traffic Flow Manageent Operations, Mitre, June 2000. Describes CAASD agent-simulator for advanced air traffic management, in a complex, adaptive environment.

[Wooldridge et al. 2000] Wooldridge, M., Jennings, NR., and Kinney, D., The Gaia methodology for agent-oriented analysis and design, Autonomous Agents and Multi-Agent Systems 3, Kluwer, 2000, 285-312.

[Zeus] Nwana, H., Nduma, D., Lee, L., Collis, J.: ZEUS: a toolkit for building distributed multi-agent systems, in Artificial Intelligence Journal, Vol. 13, No. 1, (1999) 129-186.

Min Yin and Martin Griss

Computer Science Department
University of California at Santa Cruz
USA
myin@cse.ucsc.edu
griss@soe.ucsc.edu

Reducing the Effects of the Braess Paradox with Information Manipulation

Ana L. C. Bazzan and Franziska Klügl

Abstract. The Braess Paradox is a well known phenomenon in transportation engineering: adding a new road to a traffic network may not reduce the total travel time in it. In fact, some road users may be better off but they contribute to an increase in travel time for other users. This situation happens because drivers do not face the true social cost of an action. Previous works have shown that in commuting scenarios, where people use the same traffic network routinely, a continuous learning and adaptation process is a realistic scenario: road users can adapt to the traffic conditions and will eventually learn to avoid the situation in which the cost is higher. However, this learning process can take a long time. Moreover, because the process is very sensible to the cost function and to the number of agents using the network, a more efficient approach to distribute agents in the network is to let the traffic control center to acquire and process data regarding the occupancy of the available roads and compute the optimal distribution (from the point of view of the whole system). With this information, manipulated information can be passed to the road users. The interesting point is what happens when drivers simultaneously receive this kind of information and are involved in learning processes. Thus this paper reports results obtained after simulations of several situations related to the Braess scenario: only uninformed agents using the network; with different shares of uninformed agents; drivers adapting to the traffic conditions under different learning probabilities; drivers receiving forecast; and drivers receiving manipulated information.

1. Introduction

The Braess Paradox was originally presented by Braess in 1968 [4]. This "paradox" consists of a phenomenon which contradicts the common sense: in a traffic network, when a new link connecting two points (e.g. origin and destination) is constructed,

Author partially supported by CNPq.

it is possible that there is no reduction regarding the time necessary to commute from the origin to the destination. Actually, frequently this time increases and so the costs for the commuters.

Several authors have investigate the original problem ([1, 2, 7, 8, 10] among others) as it will be detailed in the next section.

The present work describes a model already partially used by Tumer and Wolpert [10] who have investigated the use of a multi-agent system to control routing of packages in a computer network using the so-called Collective Intelligence (COIN) formalism. The authors conclude that the network is also sensible to the paradox in some cases. It happens because the agents, by trying to reduce their individual routing times in a greedy way, end up increasing the global time.

We depart from Tumer and Wolpert work in what concerns the scenario and the determination of private goals for each agent. We use the classical scenario proposed by Braess, i.e. road traffic, in which a new link added to the network is very attractive to the users since the commuting time there is low. Besides, instead of using the COIN formalism, we introduce a kind of "manipulation" of the information given to agents about the state of the system in order to distribute the agents more evenly.

The discussion on the Braess Paradox and previous approaches to it is presented in the next section. Section 3 describes the approach based on providing information manipulation to the road users and all the situations we simulated. The results of these simulations are reported in Section 4 while Section 5 discusses the conclusions and future work.

2. The Braess Paradox and Some Previous Approaches

2.1. Basic Braess Scenario

In the scenario proposed by Braess [4], drivers or agents can select between two or three available paths to commute from origin **O** to destination **D**, as depicted in Figure 1, items (a) and (b) respectively.

In the network, commuting times are computed by means of the functions depicted for each link. They are all function of the flow or number of vehicles (e.g. $T_{OQ} = f * 10$), where f is the number of vehicles in that particular edge.

In the configuration depicted in Figure 1, item (a), it is clear that if we have 6 vehicles (the original number in Braess paper), the equilibrium occurs when routes OQD and OPD carry 3 vehicles each. No one would be better off changing route.

In order to understand the paradox which occurs in the configuration depicted in Figure 1, item (b), let us consider an increasing number of vehicles starting with only one. For a single vehicle, route OQPD is much cheaper: it takes only $10+11+10 = 31$ time units to commute from O to D. Taking OPQ or OQD would take 61 time units. It is easy to see that for more than 2.58 drivers, the new route is not advantageous anymore, so drivers would not use OQPD at all.

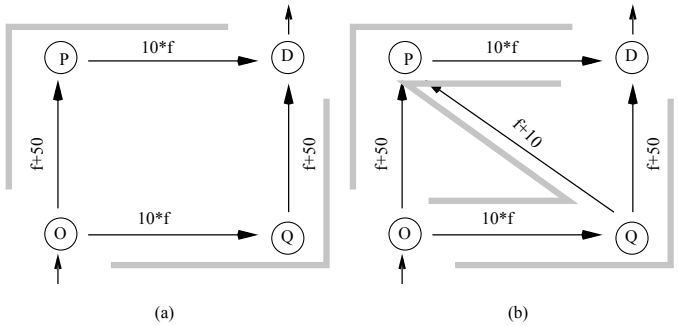


FIGURE 1. Two configurations of the network in the classical scenario of the Braess Paradox: (a) two possible routes: OPD and OQD (four-link network); (b) with additional route OQPD (five-link network)

Table 1 shows the costs for each of the five links, as well as the cost over all links, for different distribution of drivers among the three routes. The paradox is clear: when the six drivers are distribute in the two original routes, then the total cost is 176 units. However, when they find themselves two in each route, then the total cost increases to 196. This is so because the last case is the user equilibrium: each route takes 92 units so there is no incentive to change route. In the former case, routes OQD and OPD cost 83 each while route OQPD costs 70 thus being an incentive for drivers to change to OQPD. The problem is that when too many change, then the cost can be very high (see for instance the last line in the table, where the total cost is 236 because all six drivers use OQPD).

Nb. of Drivers			Cost of Link					Cost
OP	QD	QP	OP	QD	QP	OQ	PD	Total
3	3	0	53	53	10	30	30	176
2	2	2	52	52	12	40	40	196
1	2	3	51	52	13	50	40	206
1	3	2	51	53	12	50	30	196
2	1	3	52	51	13	40	50	206
2	3	1	52	53	11	40	30	186
3	1	2	53	51	12	30	50	196
3	2	1	53	52	11	30	40	186
6	0	0	56	50	10	0	60	176
0	6	0	50	56	10	60	0	176
0	0	6	50	50	16	60	60	236

TABLE 1. Cost for link and total cost for different distributions of drivers

2.2. Related Work

Following the seminal work by Braess [4], many authors have been proposing applications and modifications on the formulation of the problem in order to avoid the paradox. Smith's paper [9] aims to show how, in a particular simple case, total journey time varies with the travel time along an uncongested link. This result is of particular relevance to towns with a good bypass or, better, a good outer ring road.

From the perspective of the economics of traffic, Arnott and Small [2] analyse three paradoxes in which the usual measure for alleviating traffic congestion, i.e. expanding the road system, is ineffective. The resolution of these paradoxes — among them the Braess Paradox — employs the economic concept of externalities (when a person does not face the true social cost of an action) to identify and account for the difference between personal and social costs of using a particular road. For example, drivers do not pay for the time loss they impose on others, so they make socially-inefficient choices. This is a well-studied phenomenon more generally known as The Tragedy of the Commons [5].

Regarding the Braess Paradox, in the scenario analysed by Arnott and Small the travel time for each of the two original routes is 20 minutes, while after the addition of the new route the travel time for the equilibrium situation rises to 22.5 minutes for each route.

Returning to traffic engineering and physics communities, in Penchina [8] the "simplest anti-symmetric" two-path network is described which exhibits the Braess paradox. A discussion of the good effects (non-paradoxical) of a bridge (especially a two-way bridge) is also included. Their Minimal Critical Network and graphical solution technique gives a clear understanding of the paradox for this network. They are also especially useful for analysis of sensitivity to such extensions as, e.g. changes in parameters, elastic demand, general non-linear (even non-continuous) cost functions, two-way bridges, tolls and other methods to control the paradox, and diverse populations of users. It is shown that the paradox occurs in a simpler network than previously noted, and with a larger Braess penalty than previously noticed.

Yang and Bell [11] work deals with network design via the Braess paradox and show how this capacity paradox can be avoided by introducing the concept of network reserve capacity into network design problems. Pas and Principio [7] examine properties of the paradox and show that whether the paradox does or does not occur depends on the conditions of the problem (link congestion function, parameters and the demand for travel). Akamatsu [1] explores the properties of dynamic flow patterns on two symmetrical networks.

Although the literature from the transportation branch also proposes some ways of avoiding the paradox, all of them concentrate on the parameters of the network, not on the driver itself, thus relegating the human component which plays an important role. One can argue that even if the "right" network is constructed (i.e. one which avoids the paradox from the point of view of the mathematics of the

problem as it was proposed in the literature), drivers will always seek to maximize their individual payoffs which frequently leads to sub-optimal global distribution of traffic in the roads of the network. Besides, even if the "right" parameters are taken into account in the design of the network, the increasing demand for mobility in our society leads to a rapid obsolescence of the network. Thus we argue that it is not correct to talk about "right" parameters.

2.3. Multi-agent Approaches

Multi-agent systems approaches are interesting in the Braess scenario since one can focus on the issues related to the driver and its decision-making process. Tumer and Wolpert [10] have investigated the use of a multi-agent system to control routing of packages in a computer network, as well as the sensitivity of the network to the Braess paradox using the Collective Intelligence (COIN) formalism. Performance using COIN is compared to the case in which each agent in a set of agents estimates the "shortest path" to its destination. Since each agent decision (which is based on this estimation) ignores the effects of the decision of other agents on the overall traffic, performances based on estimation of the shortest path are badly sub-optimal. The authors conclude that the network is also sensible to the paradox in some cases. It happens because the agents, by trying to reduce their individual routing times in a greedy way, end up increasing the global time.

In the COIN approach, the new goals are tailored so that if they are collectively met the system maximizes throughput. The world utility, $G(\zeta)$, is an arbitrary function of the state of all agents across all time. The utility for an agent is given by the difference between the total cost accrued by all agents in the network and the cost accrued by agents when all agents sharing the same destination are "erased".

Bazzan and Klügl [3] present a learning heuristic and depart from Tumer and Wolpert work in what concerns the scenario and the determination of private goals for each agent. The classical scenario proposed by Braess, i.e. road traffic, is used, in which a new link added to the network is very attractive to the users since the commuting time there is low. However, since it overlaps (see Figure 1) with the existing paths, the global commuting time increases.

The approach in [3] is based on learning and adaptation by means of an heuristic capable of minimizing both the global and local performance losses. Moreover, the heuristic was developed with the multi-agent paradigm in mind requiring very little processing of each simulation agent making it very scalable. Besides, it is important to notice that agents do not need to explicitly communicate in order to coordinate their choices. Thus, the learning heuristic proposed – called A2B (Adapt to Braess) – improves the performances because it implicitly includes some factors of the global performance in the individual ones. Comparing the situation in which the agents use this heuristic to the situation in which the selection of routes happens at random, the performance greatly improves, i.e. the total commuting time decreases.

In short, A2B is based on reinforcement learning. Agents form tactics for selecting a route according to the reward obtained in that route in the past. Each tactic is associated with a learning rule.

Be i the index of an agent, and j the index of an action available to agent i. In a set of actions $A_i = (a_1,\ldots,a_j)$, a learning rule is a rule which specifies the probabilities $P_i = (p_{i,t}(a_1),\ldots,p_{i,t}(a_j))$ as a function of the rewards obtained by selecting actions in the past. In the future, each action is selected according to its probability.

In the Braess scenario, this means that each agent remembers how many times s/he used each route and the total amount of time spent traveling on each of them. With this information in mind, the agent then calculates the average travel time taken at each route and selects the one with the shortest time. The average travel time in each route is actually computed by means of a discount factor δ, in order to allow us to play with the fact that it is not desirable that agents remember *all the past* with the same weight they remember the more recent outcomes. Thus, the last time measured at a given route is multiplied by δ while the averaged past time is multiplied by $(1-\delta)$.

Hence, p_r – the probability of selecting each route r – is updated according to Equation 1, where r is the index of each available route, $\sum_r$ is the sum over all routes, and $\sum_t T_{OD,r}$ the sum of travel time for commuting from O to D using route r along time t.

$$p_r = \frac{\frac{1}{\sum_t T_{OD,r}}}{\sum_r (\frac{1}{\sum_t T_{OD,r}})} \tag{1}$$

3. Approaches and Scenarios

Although this scenario is not one of binary decision, the basic conditions from the *iterated route choice* (IRC) scenario [6] are valid. This is a model for adaptive choice in which each agent has no information about other agents. They decide which alternative to select based on a local inference about the costs or rewards of each available action from the action set. This inference is based on the update of the probability according to which an agent selects each alternative action.

In the adaptive scenario the agent i updates these probabilities with a certain periodicity according to the rewards it has obtained selecting alternative r up to that point. The update of the heuristic is done via reinforcement according to the following formula:

$$heuristic\ (i) = \frac{\sum_t reward_r(i)}{\sum_t reward(i)} \tag{2}$$

This basic scenario can be extended to give agents forecast information. Now, the decision-making process was performed in two phases. First, agents make their initial selection (first decision) based on the adaptation process introduced above. Based on this information collected from agents, a control center computes the reward for every agent and sends this information back: the agents receive a forecast information regarding the potential reward they would have if they would keep their first decision. Then, they have a second chance to actually take their first choice or change it (second decision). Finally the actual selections are made yielding the actual rewards.

The third situation occurs when "manipulated information" is given to agents to try to force an equilibrium distribution between the alternatives. For example: if the current distribution is $number_{OQPD} = 500$ and $number_{OPD} = number_{OQD} = 500$, it is clear that the system could do better in terms of *global sum* of travel times. Also, the rewards of agents are sub-optimal (not necessarily for each individual). When the control system perceives such a situation, it tries to induce agents to come closer to the global optimum by given manipulated forecast to them. In the particular case above, the control system can give a bad forecast for route $OQPD$ to try to divert drivers to other routes.

In our scenario, drivers or agents can select between the three available paths to commute from origin **O** to destination **D**, as depicted in Figure 1, item (b). In this network, commuting times are computed by the means of the cost functions depicted for each link. They are all function of the flow or number of vehicles as already explained in Section 2. Moreover, since this is a commuting scenario, agents perform this selection repeatedly. We then measure the number of drivers selecting each of the available routes, as well as the total time of commuting.

Thus, in some situations we simulate, agents just select a route at random, while in others they consider their past experience regarding performance measured by commuting time [1].

The implementation was done using SeSAm, a shell for developing agent-based simulation[2]. In the implementation, agents and vehicles can be consider a single unit. The situations we have simulated are:

- I Drivers are uninformed i.e. they all randomly select one of the 3 routes
- II Drivers select one of the 3 routes according to the algorithm proposed in [3] (A2B) which includes adaptation based on the past performance
- III Different proportions of uninformed drivers select randomly while the rest use the A2B learning algorithm
- IV All drivers receive forecast information about the occupancy of the road it has selected and either keep their decision or reselect a route
- V Drivers receive manipulated information regarding the forecast

[1] Up to now we use only this measure which is also standard in the field. However we do not completely agree that *commuting time* is the only issue drivers consider, although it is certainly the most important. We are investigating how to include driver comfort, knowledge of the network, willingness to change route, and other issues to the performance function.

[2] available for download at www.simsesam.de

Simulations of situation I is clear: N drivers randomly select between routes OPD, OQD, and OQPD. If $\vec{P}$ is the array of probabilities for selecting routes, then the initial distribution is $\vec{P_0} = (1/3, 1/3, 1/3)$. One can argue that this initial distribution does not reflect the commuting times. However, since the cost of commuting is influenced by the number of drivers at any of the routes (which is unknown), each agent actually has no preference at the beginning. Therefore, it puts equal probability to each of the routes.

Situation II was simulated taking into account a behaviour by agents which follows these ideas: drivers use the selection strategy A2B. This keeps track of choices in the past, sums the performances regarding each of the three possible route, and updates an array of probabilities for selecting each route. The performances depend on the choices of all other agents, what configures a typical problem of coordination.

While in situation II all drivers are informed, in situation III we play with the parameter *rate of random drivers*, i.e. the rate of drivers who are new in the network so they do not have enough information to use the learn-and-adapt algorithm. Thus they just select a route randomly. We use rates of 0 (which is situation II), 25, 50, 75, and 100% (situation I).

In situation IV drivers receive forecast information about the selected route, as explained above when we introduced the IRC model plus its extensions. Also here we played with the number of random drivers so when this rate is 0% everyone gets the information and adapt. When it is 100% everyone ignores the forecast. The difference in this case is that the forecast information may influence the decisions of the agents since not all will keep their first intention. It is expected that the noise increases and that the third route is used more frequently.

Finally, in situation V, the traffic control center gives "false" forecast to prevent drivers from using a route which is expected to be increase the global costs. An interesting question here is that agents can learn that the information cannot be trusted, as they not only receive this kind of information but rely on their learn-and-adapt heuristic. The situation regarding false or manipulated forecast happens frequently with radio broadcast of traffic messages. When it brings no real payoff, drivers tend to have a critical view of it.

4. Description of the Experiments and Results

In this section we discuss the results regarding the five situations presented above. In all cases, the N agents interact and have to implicitly coordinate since they are using shared resources. For the experiments, we use $N = 1500$ agents, $\delta = 0.8$, and the learning probability is 0.2. Cases with less agents and in particular with $N = 6$ are reported in [3].

As for the cost functions to compute the commuting time, they are shown in Table 2. For $N = 1500$, the paradox happens with parameters shown in Table 3.

Links	Function
OQ	$T_{OQ} = \beta_1 * f_{OQ} + \alpha_1$
PD	$T_{PD} = \beta_1 * f_{OQ} + \alpha_1$
OP	$T_{OP} = \beta_2 * f_{OP} + \alpha_2$
QD	$T_{QD} = \beta_2 * f_{QD} + \alpha_2$
QP	$T_{QP} = \beta_4 * f_{QP} + \alpha_4$

TABLE 2. Functions to compute the time to traverse each route in the network.

Parameter	Values
α_1	70
α_2	500
α_4	5
β_1	0.2
β_2	0.05
β_4	0.5

TABLE 3. Values for parameters when $N = 1500$.

With these parameters, the global costs can be computed for the particular situations of interest:

1. when the $N = 1500$ drivers are distributed equally among the 3 routes, the global cost is 795 for $OQPD$, OQD, and OPD; this is also the average cost per driver, while the global cost is 1845 (over all sub paths).
2. when all drivers avoid the $OQPD$ route and 750 use each remaining route, the total cost is 445 for $OQPD$ and 757.5 for the other two, thus adding up 1520.

In the simulations, the dimensions measured were mainly distribution of drivers in each route, and the probability of selecting each route (averaged over the N agents). The cost is a direct function of the distribution of drivers so we do not plot costs here.

In situation I the paradox can be observed: since the selection of the 3 routes is random, some drivers do select route $OQPD$ and this increases the global commuting time.

The case of situation II for $N = 1500$ agents was discussed in [3]. The majority of the agents tended to avoid the third route, although this does not happen completely due to the inertia in the learning process. The distribution of agents between the two remaining routes tended to be equal, as it maximizes the average performance of the agents. The learning process is slow and sensible to the cost parameters.

The simulations of situation III show that the increase in the number of random drivers increases the usage of the route $OQPD$ thus increasing the global cost.

As for situation IV, the forecast given to the drivers does not help when it is the correct one, that means, when it is the information actually computed from the first intention of each driver. This happens because, in the majority of the cases, the forecast is worse than the expected value a driver computes for the selected route. Thus, the driver rejects his first selection and perform a second selection randomly. As above, situations in which too many drivers choose randomly brings to an usage of $OQPD$ and therefore, to an increase in costs. Even if drivers have a tolerance to the forecast (i.e. they do not simply reject their first selection but to[illegible]e be[illegible].

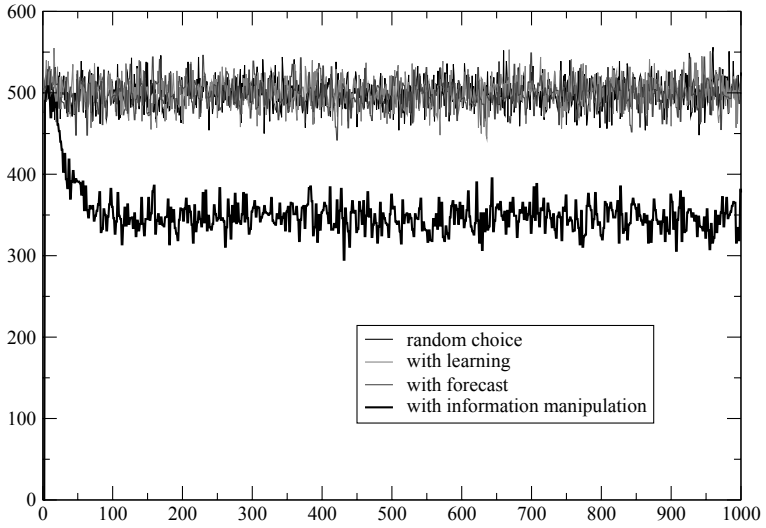


FIGURE 2. Number of drivers selecting route $OQPD$ along time, under different conditions (only random selection, with learning, with forecast, and under information manipulation ($N = 1500$).

Therefore, the best results were achieved with situation V, i.e. with information manipulation. The gain in performance is summarized in Figure 2. We plot the number of drivers who select route $OQPD$ along time, for the situations above: with only random selection, with the learn-and-adapt process, with forecast (and no random driver), and under manipulation of information (and no random driver). One sees that only the last case brings some of the number of drivers to avoid the route $OQPD$. In the other cases, around 1/3 of the drivers still select this route (curves overlap).

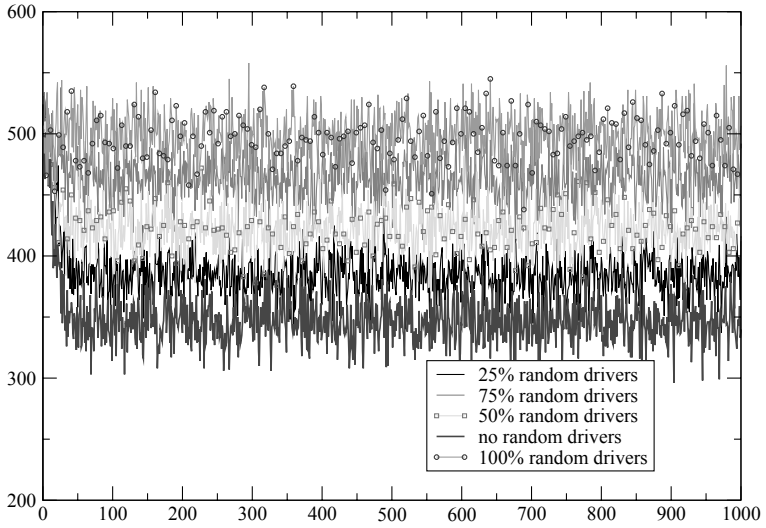


FIGURE 3. Number of drivers selecting routes $OQPD$ along time, under manipulation and with different rates of random selection ($N = 1500$ agents).

Now, it is interesting to investigate what happens if not all drivers are informed, i.e. when random drivers perform the route selection together with drivers who learn and get manipulated forecast. We have simulated this situation with different shares of uninformed drivers: 0, 25, 50, 75, and 100% and these results are depicted in Figure 3).

The case with the lowest number of drivers selecting $OQPD$ is the one in which everybody is informed (thus, the same curve which appeared in Figure 2). This shows that the role of information is significant and, moreover, that the role of manipulated information is even more important than the correct forecast since the latter can introduce noisy in the learning process of the drivers, especially when too many drivers are unaware of this information. For higher shares of uninformed drivers, the trend is that the route $OQPD$ is selected more often. Of course, in case 100% of drivers do not get information, the manipulation has no effect and thus, the selection is random and inefficient both for each driver and for the whole system.

5. Conclusion and Future Work

In the classical scenarios on the Braess Paradox reported in the literature, if drivers or agents of any kind, act to maximize their own profits, the global performance of the system may decrease. This happens because the global goal opposes the individual goals in most cases.

While studying the Braess Paradox, we noticed that this could be an interesting scenario for investigating issues related to learning and adaptation as well as information manipulation in order to minimize both the global and local performance losses. Moreover, the heuristic was developed with the multi-agent paradigm in mind requiring very little processing of each simulation agent making it very scalable. Besides, it is important to notice that agents do not need to explicitly communicate in order to coordinate their choices.

In this paper we have studied the effect of information manipulation in the Braess scenario. The simulations show that it is useful to manipulate the forecast information given to the agents. Doing so, the control system is able to divert them to the more convenient alternative, from the point of view of both the overall system and the individual agents (as this is the situation in which individual rewards are the highest). This holds in higher or lower degrees for different shares of uninformed drivers.

The main conclusion is that having agents provided with the most accurate information (information about the *actual* state of the system) is not necessarily good.

This work is based on a series of assumptions that may not be bearable in every real world application. First, we assume a global control component that is able to compute the *exact* utility of the agent decisions for producing the forecast information. Although this degree of exactness in forecast might not be necessary, under which circumstances the existence of such a forecast system is realistic in general?

Second, and more interesting from the perspective of mechanism design, is the assumption that the central component acts in the interest of the highest system performance. However, when we have other interests involved (for instance when there are several competing "central" controller components), some of them may act primarily to disturb the others.

Therefore, the next directions of this research are twofold: the system may be able to learn the share of uninformed and account for this share when giving manipulated information; and designing of more complex agent architectures to accommodate competition and self-interest.

References

[1] Akamatsu, T. dynamic traffic equilibrium assignment paradox. *Transportation Research B 6* (2000), 515–531.

[2] Arnott, R., and Small, K. The economics of traffic congestion. *American Scientist 82* (1994), 446–455.

[3] Bazzan, A. L. C., and Klügl, F. Learning to behave socially and avoid the braess paradox in a commuting scenario. In *5th Workshop on Decision Theoretic and Game Theoretic Agents* (July 2003), S. Parsons and P. Gmytrasiewicz, Eds., pp. 107–112. held together with AAMAS 2003.

[4] Braess, D. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung 12* (1968), 258.

[5] Hardin, G. The tragedy of the commons. *Science 162* (1968), 1243–1248.

[6] Klügl, F., and Bazzan, A. L. C. Simulation of adaptive agents: Learning heuristics for route choice in a commuter scenario. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (Bologna, Italy, July 2002), vol. 1, ACM Press, pp. 217–218. extended abstract.

[7] Pas, E. I., and Principio, S. L. Braess' paradox: Some new insights. *Transportation Research B 31*, 3 (1997), 265–276.

[8] Penchina, C. M. Braess paradox: maximum penalty in a minimal critical network. *Transportation Research A 31*, 5 (1997), 379–388.

[9] Smith, M. J. In a road network, increasing delay locally can reduce delay globally. *Transportation Research 12*, 6 (1978), 419–422.

[10] Tumer, K., and Wolpert, D. Collective intelligence and braess' paradox. In *Proceedings of the AAAI* (2000), AAAI Press.

[11] Yang, H., and Bell, M. G. H. A capacity paradox in network design and how to avoid it. *Transportation Research A 32*, 7 (1998), 539–545.

Ana L. C. Bazzan
Instituto de Informática
UFRGS
Caixa Postal 15064
91.501-970 Porto Alegre, RS, Brazil
e-mail: `bazzan@inf.ufrgs.br`

Franziska Klügl
Dep. of Artificial Intelligence
University of Würzburg
Am Hubland
97074 Würzburg, Germany
e-mail: `kluegl@informatik.uni-wuerzburg.de`

Analysis of the Effect of Route Information Sharing on Reduction of Traffic Congestion

Tomohisa Yamashita, Kiyoshi Izumi and Koichi Kurumatani

Abstract. This research is intended to increase drivers' utility by reducing traffic congestion. To attain our purpose, we propose a simple route guidance mechanism based on route information sharing (RIS). Drivers using the RIS give planned route details to the route information server, which then sends accumulated traffic information based on those routes to drivers. Multiagent simulation in a lattice network and a radial and ring network confirmed that: i) as the drivers using the RIS increased, the travel time of both the drivers using the RIS and the other drivers decreased; ii) the travel times of drivers using the RIS were substantially shorter than those of drivers using other mechanisms.

1. Introduction

Recently, ubiquitous computing environments in road transportation systems have developed rapidly. Car navigation systems have spread widely; moreover, the accuracy of Global Positioning System (GPS) is constantly advancing. The vehicle information and communication system (VICS) was started in 1996. Since that time, the range over which the VICS can provide congestion information has been extended [9]. [1] These advances were realized through the development of in-vehicle communication devices, sensors, and processors. An important information service in road transportation systems is navigation from an origin to a destination. Based on such developments, many researchers have been trying to develop navigation systems and to examine what kind of traffic information is available [1]. Generally, the purpose of a navigation system is to maximize the efficiency of an individual user by providing routes that satisfy their intentions, e.g., by reducing travel

[1]The VICS Center [9] collects, processes, and edits information about traffic congestion, road control, and other traffic information in real time; It then provides text and graphic traffic information through communications and broadcast media (FM multiplex broadcasting and beacons). Drivers can receive traffic information via navigation devices installed in vehicles.

distance or travel time. However, previous research efforts have revealed that individually optimizing performance by merely providing information about traffic congestion is difficult [4, 6, 8].

Usually, navigation systems recommend routes for decreasing travel time based on the current state of traffic congestion. To decrease travel time, a driver using a navigation system chooses a recommended route that is not congested. However, if other drivers simultaneously choose the same recommended route, traffic can be concentrated on that route. Consequently leading to congestion. As a result, traveling the recommended route can take longer even though navigation systems recommended it to decrease travel time. To make matters worse, traffic congestion increases exponentially as more drivers choose the recommended route. Because car navigation systems have spread rapidly, this is a potentially serious problem.

Our work introduces the concept of mass user support [2, 3] to avoid unintentional traffic congestion caused by use of navigation systems. Mass user support provides information services to users, groups, and the general populace that goes beyond conventional personal services. Conventional information services only consider utility for individual users, but mass user support considers not only the sum of personal services provided to individual users, but also interactions among users. Mass user support is intended to increase utility for both individuals and society, but not by sacrificing that for individuals.

That is to say, mass user support requires some kind of social coordination that engenders mutual concessions among users [2]. Moreover, a service based on mass user support is required to provide not only a higher utility, but also fairness among users, service stability, transparency of the mechanism, and robustness against disturbance because mass user support serves the needs of many users simultaneously.

Our research aims to increase both the individual's utility and the social benefits of road transportation systems by reducing traffic congestion. To develop a route guidance mechanism that promotes mutual concessions among users autonomously, we introduce route information sharing through the route information server. Drivers using our proposed mechanism inform the route information server of routes that they will take in the future. The server calculates accumulated and prospective traffic state based on collected routes, and then sends this information to the drivers using our proposed mechanism. Each driver reconsiders and changes the route based on accumulated traffic information from the server.

Using multiagent simulation, we examine the effect of our proposed mechanism from the following two points of view, individual incentive and social acceptability. Individual incentive means the incentive of each driver to change from others to our proposed mechanism. The following feature is required for individual incentive: traffic efficiency, e.g., travel time, of drivers using our proposed mechanism should always be higher than that of drivers using other mechanisms. Social acceptability means the degree of acceptability with our proposed mechanism

spreads. The following is required for social acceptability: as the drivers using our proposed mechanism increase, their traffic efficiency should be improved.

2. Multiagent Modeling

2.1. Traffic Model

Our work constructed a simple traffic flow model to examine the interdependence of traffic congestion and route choice behaviors. Therefore, we did not consider the following factors: traffic signals (e.g., stopping at a red lights), waiting for oncoming cars when turning at intersections, turn lanes, multiple lanes, passing, blind allies; and U-turn in lanes, not at intersections.

Our traffic flow model designates a road between intersections as a link. It is divided into several blocks. The block length is equal to the distance that a vehicle runs at the free flow speed of V_f of the link during one simulation step. After link division, an order is assigned to each block from downstream to upstream. Concerning the block assigned to be the i-th, we define K_i as the density of block i, L_i as the length of block i, N_i as the number of the vehicles in block i, and V_i as the feasible speed of vehicles in block i. K_i is the division of N_i by L_i. In block i, V_i is revised based on Greenshield's V-K relationship described as

$$V_i = V_f(1 - \tfrac{K_i}{K_{jam}}), \tag{1}$$

where K_{jam} is the traffic jam density. The density signifies the minimum density that prevents vehicles in a traffic jam from moving. In our simulation, we set these coefficients as $V_f = 13.89$ and $K_{jam} = 0.14$.

The process of the flow calculation between neighboring blocks i and $i+1$ is as follows: At every step, the speed of vehicles in each block is revised according to the V-K relationship. Then vehicles move forward based on this speed. The vehicles' movement is processed from downstream to upstream, as shown in Fig. 1. Based on V_i, vehicle j can move forward. When vehicle j moves into block i from block $i+1$, its speed changes to V_i from V_{i+1}. If K_i is exceeds jam density K_{jam}, no vehicles can move into block i from block $i+1$. After movement of vehicle j_1 in front of vehicle j_2, if vehicle j_1 is within a distance that allows vehicle j_2 to move forward at the speed of vehicles V_i, vehicle j_2 approaches vehicle j_1 to the minimum distance between two cars. Although vehicle j_2 has sufficient speed to advance, it must remain behind vehicle j_1. At the next step in block i, when V_i is revised based on K_i, vehicles can accelerate or slow down to V_i immediately, regardless of the speed in the last step.

In terms of applying the Q-K or V-K relationship to each block, the method of flow calculation in our model is similar to that of the Hybrid Block Density Method (HBDM) [7]. Although the HBDM treats traffic flow as a continuum, i.e., with a continuous approximation of vehicles, our method treats each individual vehicle discretely using a multiagent approach. In terms of treating each individual vehicle discretely, our model is similar to the Nagel-Schrekenberg model [5]. However,

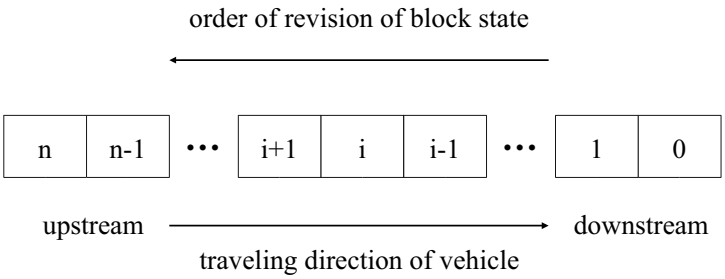


FIGURE 1. Direction of movement of vehicles and revision of blocks

using that model to address large-scale road transportation systems is difficult. The number of blocks increases as the scale of systems expands because the block length is fixed to one vehicle. Accordingly, the computational cost of calculating the state of all blocks increases markedly. In contrast, in our model, computational cost can be controlled by extending the block length because the length of one block can be variable. Therefore, our model is suitable for addressing with phenomena caused by route choice behaviors in large-scale traffic transportation systems.

2.2. Route Choice Mechanism

We prepared three types of route choice mechanisms for examining our proposed mechanism. The first and second types are well known and easy to understand because they seek routes minimizing travel distance or travel time. The third type is our proposed mechanism with route information sharing.

2.2.1. Shortest Distance Route. Drivers using the shortest distance route (SD) select a route based on their knowledge of a map without information about traffic congestion. Drivers using the SD simply choose the shortest route from their respective origin to their destination. They only consider the distance of a link and search for the shortest route for that distance.

2.2.2. Shortest Time Route. Drivers using the shortest time route (ST) decide a route based on their knowledge of a map and information about current traffic congestion. Drivers using the ST represent dynamic route choice based not only on map information, but also on current congestion information about the entire network at anytime, as obtained from a traffic information center (e.g., a VICS Center) via vehicle equipment. They search for the route that has the shortest travel time from their origin to their destination, and revise their route whenever they approach an intersection.

A driver using the ST is assumed to receive information about the current traffic density of all blocks from a traffic information center via vehicle equipment. Drivers consider the expected travel time of a link as calculated based on the current density of each block as follows. First, the speed on block i is calculated based on the V-K relationship with density K_i. Next, the passage time of block

i is calculated based on length L_i of block i and speed V_i on block i. Finally, the passage time of a link is calculated by summing the passage time of all blocks on the link. We define expected travel time ETT_l as the summation of the passage times of all blocks on link l. Drivers using the ST search for the shortest route based on the expected travel time from their current position to their destination at every intersection.

2.2.3. Shortest Time Route with Route Information Sharing. Drivers using the shortest time route with route information sharing (RIS) choose a route based on their knowledge of a map, information regarding current traffic congestion, and accumulated information about the routes of other drivers using the RIS. The drivers using the RIS also search for their routes and revise them whenever they approach an intersection. The difference between the RIS and ST is that the drivers using the RIS share their route information from their respective origins to destinations. This sharing can be achieved easily through communication with a route information server via devices, e.g., cellular phone.

The outline of route information sharing mechanism is the following: First, drivers using the RIS search for the shortest route from their origins to their destinations in the expected travel time. Next, the drivers notify the route information server of their routes. The route information server collects the routes of all drivers using the RIS and assigns passage weight of a driver to a link based on their routes, current positions, and destinations. Passage weight means the degree of assurance that a driver using the RIS will pass through a link in the future. Based on passage weight of each link, the route information server provides the total passage weight of each link for drivers using the RIS. Total passage weight means the accumulation of passage weight of all drivers using the RIS. Finally, drivers using the RIS calculate the prospective traffic volume of each link based on the total passage weight and current traffic state of each link; They search for the shortest route in the prospective traffic volume from their current positions to their destinations.

We define passage weight $PW_{l,j}$ of driver j to link l as follows. If a route passes through p links from driver's current position to a destination, the route information server assigns an order to links from the destination to the driver's current position on the route. Next, the route information server divides the order by p and regards it as the weight of a link. For example, $1/p$ is assigned to a link including a destination, 1 ($=p/p$) is assigned to a link including a current position.

Furthermore, we define total passage weight TPW_l as the sum of the passage weights of all drivers to link l. The route information server calculates TPW_l as

$$TPW_l = \Sigma^k_{k \in RIS}\, PW_{l,k}, \tag{2}$$

where RIS is the set of drivers using the RIS. Finally, we define the prospective traffic volume PTV_l of link l. The route information server calculates PTV_l as

$$PTV_l = ETT_l \times (TPW_l + \alpha), \tag{3}$$

where α is a positive constant. In our simulation, we set α at 1.0. The route information server sends the prospective traffic volume of all links to drivers using

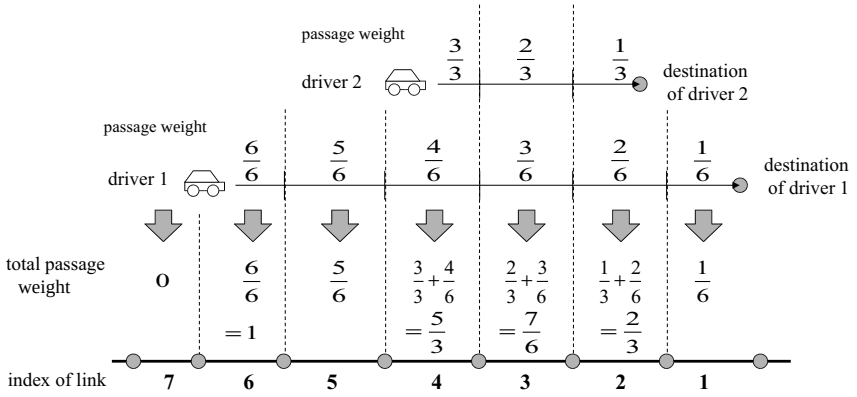


FIGURE 2. Sample calculation of total passage weight

the RIS. Drivers using the RIS seek the shortest route in the prospective traffic volume from their current positions to their destinations.

Figure 2 shows an example of calculating total passage weight. Driver 1 has the route through six links 6, 5, 4, 3, 2, 1 from a current position on link 6 to a destination on link 1. Based on the current position, destination, and route of driver 1, passage weights for links 1 to 7 of driver 1 are:

$$\begin{aligned} &PW_{1,1} = 1/6, PW_{1,2} = 2/6, PW_{1,3} = 3/6, \\ &PW_{1,4} = 4/6, PW_{1,5} = 5/6, PW_{1,6} = 6/6, \\ &PW_{1,7} = 0. \end{aligned} \tag{4}$$

Driver 2 has the route through three links 4, 3, 2 from a current position on link 4 to a destination on link 2. Similarly, passage weights of link 1 to 7 of driver 2 are:

$$\begin{aligned} &PW_{2,2} = 1/3, PW_{2,3} = 2/3, PW_{2,4} = 3/3, \\ &PW_{2,1} = PW_{2,5} = PW_{2,6} = PW_{2,7} = 0. \end{aligned} \tag{5}$$

According to the passage weights for links 1 to 7 of drivers 1 and 2, the total passage weights of link 1 to 7 are:

$$\begin{aligned} &TPW_1 = 1/6, TPW_2 = 2/3, TPW_3 = 7/6, \\ &TPW_4 = 5/3, TPW_5 = 5/6, TPW_6 = 1, \\ &TPW_7 = 0. \end{aligned} \tag{6}$$

By this mechanism, the route information server must know the route from a current position to a destination for each driver using the RIS. However, the route information server does not need to know which drivers pass through the route. Accordingly, this mechanism offers advantages in terms of anonymity. Furthermore, communication load between the drivers using the RIS and the route information server is light because the drivers using the RIS provide the routes to the route information server; Subsequently the route information server broadcasts the prospective traffic volume to them. The route information server does

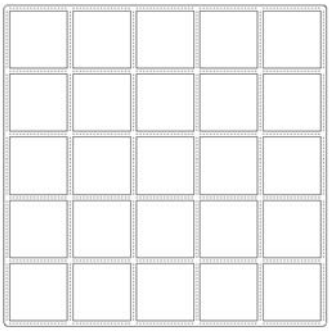


FIGURE 3. Lattice network

TABLE 1. Details of the lattice network and the radial and ring network

	lattice	radial and ring
Number of nodes	36	32
Number of links	60 × 2	56 × 2
Number of blocks	23,160	23,144
Total length (km)	169.1	169.2

not exert great computational effort because each driver using the RIS decides the route autonomously based on the prospective traffic volume. Therefore, this mechanism offers advantages in terms of simplicity.

3. Computer Simulation

3.1. Simulation Settings

In order to evaluate route information sharing mechanism, we performed simulations of three route choice types for which the ratio of the drivers using the SD is fixed at 0.2, and the ratio of the drivers using the ST and RIS are altered from ST:RIS = 0.8:0 to ST:RIS = 0:0.8. This setting is based on an estimation of the future use of traffic information systems. Traffic information will probably become more easily accessible for many drivers because of the propagation of navigation systems. Furthermore, we used two kinds of road networks to evaluate the effectiveness of our proposed mechanism: the lattice network and the radial and ring network. Details of those two networks are shown in Figures 3 and 4, and described in Table 1. In these road networks, all links have the same capacity.

The travel time of each driver was normalized against the ideal travel time to compare our simulation results of the different road networks and the different sets of origins and destinations of vehicles. The ideal travel time is the time required from an origin to a destination when a driver passes through the shortest route

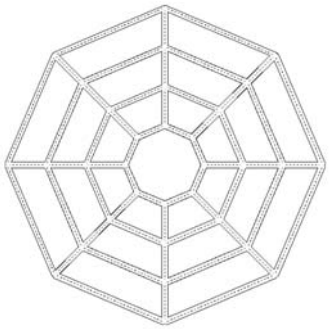


FIGURE 4. Radial and ring network

for the distance at free flow speed. The travel time is defined as the ratio of the actual travel time and the ideal travel time.

The origin and destination of a driver were assigned randomly to any block on any link in Figs. 3 and 4. After reaching its destination, the vehicle is removed from the network. At every simulation step, vehicles were generated until the amount of vehicles reaches 25,000. The number of vehicles generated at one step N_{gen} is decided as follows:

Preliminary simulations under the condition that the ratio of the drivers using the SD and ST is 0.2:0.8 confirmed the following. In the lattice network, shown in Figure 3, at $N_{gen} \leq 35$, traffic congestion was not caused: the actual travel time of all drivers using any mechanism is equal to the ideal travel time. At $N_{gen} \geq 55$, a gridlock occurred. In the radial and ring network shown in Figure 4, traffic congestion was not caused at $N_{gen} \leq 25$, and a gridlock occurred at $N_{gen} \geq 40$. Our traffic simulation defines a gridlock as a situation in which vehicles on a link cannot move into a neighboring link because all are filled completely by other vehicles. Furthermore, the vehicles in those neighboring and subsequent links also could not move into the following links for the same reason. Once vehicles are caught in a gridlock, it is impossible to escape. Because traffic congestion with a gridlock is not observed usually in actual traffic flow, we use a situation in the middle of vacant state (i.e., all vehicles can drive at free flow speed) and gridlock in the entire network. Therefore, we set two kinds of N_{gen} in each network: $N_{gen} = 40$ and $N_{gen} = 45$ in the lattice network, and $N_{gen} = 30$ and $N_{gen} = 35$ in the radial and ring network.

3.2. Simulation Results

In estimating the results of simulations, we particularly addressed a comparison of the average travel time of drivers using the SD, ST, and RIS. Results of our simulation based on an average of 10 trials are shown in Figures. 5 to 8. In these graphs, the horizontal axis is the ratio of the drivers using the RIS; the vertical axis is the average travel time. The ratio of the drivers using the RIS among all

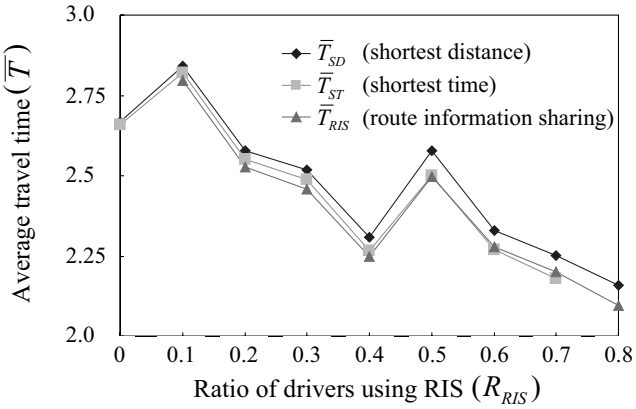


FIGURE 5. Average travel time in the case of $N_{gen} = 40$ in the lattice network

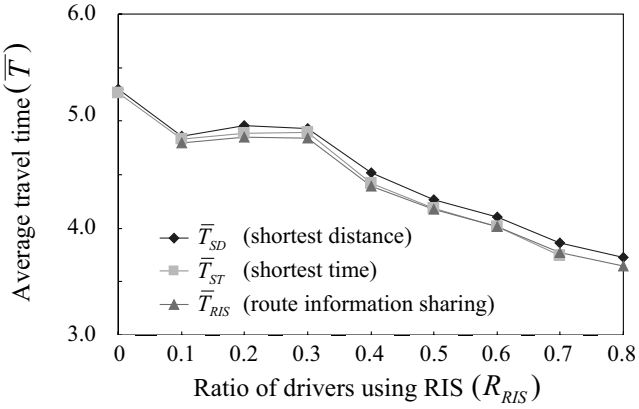


FIGURE 6. Average travel time in the case of $N_{gen} = 45$ in the lattice network

drivers is described as R_{RIS}. The average travel time of drivers using the SD is described as $\overline{T}_{SD}$. Similarly, the average travel time of drivers using the ST and that of drivers using the RIS are described as $\overline{T}_{ST}$ and $\overline{T}_{RIS}$.

Figures 5 and 6 show results in the lattice network. Figure 5 shows that the average travel time in the case of $N_{gen} = 40$. $\overline{T}_{SD}$ and $\overline{T}_{ST}$ at $R_{RIS} = 0.1$ were worse than these at $R_{RIS} = 0$. Until $R_{RIS} = 0.4$, $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$ decreased as R_{RIS} increased. $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$ increased rapidly at $R_{RIS} = 0.5$. After $R_{RIS} = 0.6$, all three decreased as R_{RIS} increased. Except at $R_{RIS} = 0.7$, the

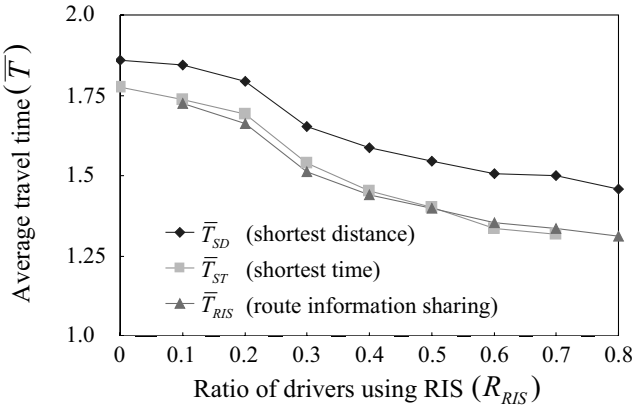


FIGURE 7. Average travel time in the case of $N_{gen} = 30$ in the radial and ring network

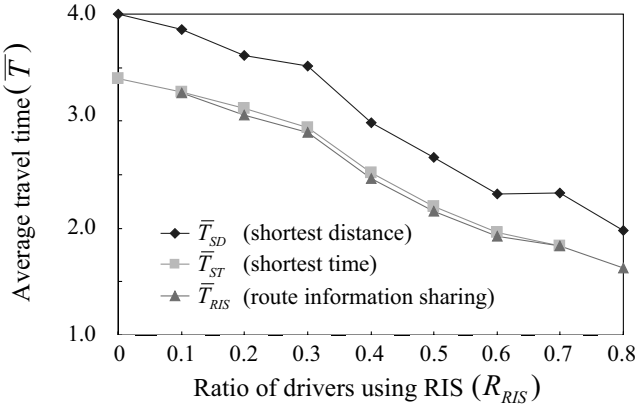


FIGURE 8. Average travel time in the case of $N_{gen} = 35$ in the radial and ring network

average travel time of the three types were ranked in ascending order as $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$. Only at $R_{RIS} = 0.7$, $\overline{T}_{ST}$ was better than $\overline{T}_{RIS}$.

Figure 6 shows the average travel time in the case of $N_{gen} = 45$. From $R_{RIS} = 0.1$ to $R_{RIS} = 0.3$, $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$ slightly increased. Subsequently, all three decreased as R_{RIS} increased. the average travel times of three types were always ranked in ascending order as $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$. In all cases, there was only marginal difference among them.

Figures 7 and 8 show results of the radial and ring network. Figure 7 shows the average travel time in the case of $N_{gen} = 30$. $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$ decreased

as R_{RIS} increased. In all cases, there was always no difference between $\overline{T}_{ST}$ and $\overline{T}_{RIS}$. $\overline{T}_{ST}$ and $\overline{T}_{RIS}$ were always superior to $\overline{T}_{SD}$. The difference between $\overline{T}_{RIS}$ and $\overline{T}_{SD}$ was from 0.08 to 0.17. Until $R_{RIS} = 0.4$, $\overline{T}_{RIS}$ was better than $\overline{T}_{ST}$. At $R_{RIS} = 0.5$, $\overline{T}_{RIS}$ was equal to $\overline{T}_{ST}$. Subsequently, $\overline{T}_{ST}$ was superior to than $\overline{T}_{RIS}$.

Figure 8 shows the average travel times in the case of $N_{gen} = 35$. Similarly to the case of $N_{gen} = 30$, $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$ decreased as R_{RIS} increased. The average travel times of three types were always ranked in ascending order as $\overline{T}_{SD}$, $\overline{T}_{ST}$, and $\overline{T}_{RIS}$. $\overline{T}_{RIS}$ was always marginally better than $\overline{T}_{ST}$. The difference between $\overline{T}_{RIS}$ and $\overline{T}_{SD}$ was from 0.35 to 0.62.

4. Discussion

This section presents discussions about the effect of our proposed mechanism based on the results of our simulation. First, we discuss the effect of route information sharing mechanism from the point of whether it satisfies individual incentive and social acceptability. In the lattice network, individual incentive was satisfied in both cases of $N_{gen} = 40$ and $N_{gen} = 45$ because $\overline{T}_{RIS}$ was always slightly better than $\overline{T}_{SD}$ and $\overline{T}_{ST}$. Social acceptability was not satisfied completely. $\overline{T}_{RIS}$ did not decrease uniformly as the drivers using the RIS increased, but $\overline{T}_{RIS}$ did not continue to increase markedly. In the case of $N_{gen} = 40$, $\overline{T}_{RIS}$ at $R_{RIS} = 0.1$ were worse than $\overline{T}_{SD}$ and $\overline{T}_{ST}$ at $R_{RIS} = 0$; $\overline{T}_{RIS}$ increased rapidly at $R_{RIS} = 0.5$. In the case of $N_{gen} = 45$, $\overline{T}_{RIS}$ increased slightly from $R_{RIS} = 0.1$ to $R_{RIS} = 0.3$. For those reasons, in the lattice network, the RIS mechanism is more effective when the links are relatively congested.

In the radial and ring network, individual incentive was satisfied in both cases of $N_{gen} = 30$ and $N_{gen} = 35$, for the most part, because $\overline{T}_{RIS}$ was always slightly better than $\overline{T}_{SD}$ and $\overline{T}_{ST}$. Not with standing, in the case of $N_{gen} = 30$, $\overline{T}_{ST}$ was better than $\overline{T}_{RIS}$ at $R_{RIS} \geq 0.6$; the difference between them was slight. Social acceptability was satisfied completely because $\overline{T}_{RIS}$ decreased uniformly as the drivers using the RIS increased. Therefore, the RIS mechanism is always effective in the radial and ring network.

Next, we discuss the different effect of the RIS in the lattice network and the radial and ring network. In the radial and ring network, a driver had only one or two shortest routes in the distance. Because these shortest routes statistically tended to pass through the innermost ring, drivers using the SD tended to concentrate at the innermost ring. Drivers using the ST and RIS could avoid traffic congestion that was caused in the innermost ring by drivers using the SD. However, drivers using the ST often caused traffic congestion in the second and third innermost ring by avoiding the innermost ring and concentrating vacant links. On the other hand, drivers using the RIS could prevent causing traffic congestion in the second and third innermost rings. Therefore, in the radial and ring network, the travel

time of drivers using the RIS was able to satisfy individual incentive and social acceptability.

In the lattice network, drivers using the SD did not concentrate on central links seriously because drivers using the SD had some shortest distance routes and chose one of them randomly. Because traffic congestion tended to occur at links other than central links, drivers using the ST and RIS had more difficulty in preliminarily avoiding congested links. Once traffic congestion occurred at a link in the lattice network, traffic congestion in other links often occurred because drivers using the ST concentrated on near and uncrowded links to avoid traffic congestion. And then, drivers using the RIS were often involved in traffic congestion. These phenomena often occurred regardless the ratio of drivers using the RIS when the ratio of drivers using the RIS was not high. Accordingly, although drivers using the RIS increased, the travel time of drivers using the RIS did not decrease consistently. The difference of travel times among the SD, ST and RIS was small.

Finally, we discuss the practical application of route information sharing. In our simulation, the lattice network we used was very symmetrical and ideal: all links had the same capacity, and there were no one-way streets and traffic regulation. In actual road networks, there are not many shortest distance routes from an origin to a destination. Moreover, links in which the number of passing vehicles exceeds traffic capacity, i.e., bottlenecks, exist certainly. Therefore, actual lattice networks may show similar characteristics to the radial and ring networks for which traffic congestion tends to occur in certain bottlenecks. In such networks, it is important to prevent two types of traffic congestion. One is traffic congestion occurring in bottlenecks; the other occurs in other links as drivers avoid the first type of traffic congestion and concentrate in vacant alternative routes. Because route information sharing is regarded as an effective mechanism for preventing two types of traffic congestion in the radial and ring network, the RIS mechanism is effective on improving traffic efficiency on actual traffic flow.

5. Conclusion

This study proposed a route guidance mechanism with route information sharing. Our proposal is intended on improving traffic efficiency for individual drivers and a whole system. Using multiagent modeling, we constructed a simple traffic flow model based on the V-K relationships. Three types of route choice behavior were prepared: the Shortest Distance (SD), the Shortest Time (ST), and the Shortest Time with Route Information Sharing (RIS). Drivers using the RIS notified the route information server of their route. The server processed accumulated information based on their routes and then returned it to them. The effectiveness of this mechanism was examined in a lattice network and a radial and ring network. Results of simulation confirmed that the RIS mechanism is effective to improve the traffic efficiency for drivers using the RIS.

References

[1] Klugl, F., Bazzan, A.L.C., Wahle, J.: Selection of Information Types Based on Personal Utility: A Testbed for Traffic Information Markets. In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent systems (2003) 377-384

[2] Kurumatani, K.: Mass User Support by Social Coordination Among Citizens in a Real Environment. In Multiagent for Mass User Support, LNAI 3012, Springer (2004), 1-19

[3] Kurumatani, K.: Social Coordination with Architecture for Ubiquitous Agents: CONSORTS. In Proceedings of International Conference on Intelligent Agents, Web Technologies and Internet Commerce 2003 (CD-ROM) (2003)

[4] Mahmassani, H. S., Jayakrishnan, R.: System Performance and User Response Under Real-Time Information in a Congested Traffic Corridor. Transportation Research 25A(5) (1991) 293-307

[5] Nagel, K., Schreckenberg, M. A.: A Cellular Automaton Model for Freeway Traffic. Journal de Physique I (2) (1992) 2221-2229

[6] Tanahashi, I., Kitaoka, H., Baba, M., H. Mori, H., Terada, S., Teramoto, E.: NETSTREAM, a Traffic Simulator for Large-scale Road Networks, R & D Review of Toyota CRDL, 37(2) (2002) 47-53 (in Japanese)

[7] Teramoto, E., Baba, M., Mori, H., Asano, Y., Morita, H.: NETSTREAM: Traffic Simulator for Evaluating Traffic Information Systems. In Proceedings of the IEEE International Conference on Intelligent Transportation Systems '97 (CD-ROM) (1997)

[8] Yoshii, T., Akahane, H., Kuwahara, M.: Impacts of the Accuracy of Traffic Information in Dynamic Route Guidance Systems. In Proceedings of The 3rd Annual World Congress on Intelligent Transport Systems (CD-ROM) (1996)

[9] http://www.vics.or.jp

Tomohisa Yamashita, Kiyoshi Izumi and Koichi Kurumatani
Cyber Assist Research Center (CARC)
National Institute of Advanced Industrial Science and Technology (AIST)
2-41-6, Aomi, Koto-ku, Tokyo 135-0064
Japan
e-mail: tomohisa@carc.aist.go.jp
kiyoshi@ni.aist.go.jp
k.kurumatani@aist.go.jp

A Test Bed for Multi-Agent Control Systems in Road Traffic Management

R.T. van Katwijk, P. van Koningsbruggen, B. De Schutter and J. Hellendoorn

Abstract. In this paper we present a test bed for multi-agent control systems in road traffic management. In literature no consensus exists about the best configuration of the traffic managing multi-agent system and how the activities of the agents that comprise the multi-agent system should be coordinated. The system should be capable of managing different levels of complexity, a diversity of policy goals, and different forms of traffic problems. The test bed aids in-depth research in this field, which we demonstrate by means of two example scenarios we have implemented.

1. Introduction

Throughout the world, traffic congestion forms a daily recurring problem. In many countries, more and more instruments are deployed to stimulate a smooth and safe flow of traffic on highways and urban road networks. Examples are ramp metering installations, which are used to regulate the inflow of traffic at on-ramps, variable message signs, which are amongst others used to inform road users about the current road conditions ahead or to present them with the current speed limit, and traffic signal installations, used to control traffic at intersections. Research is being conducted in the field of automatic coordination of these dynamic traffic management instruments. In [3, 6, 7, 11, 12, 14, 22, 23] it is argued that the communication capabilities of multi-agent systems can be used to accomplish this coordination.

No consensus exists about the best configuration of the traffic managing multi-agent system. The system should be capable of managing different levels of complexity, a diversity of policy goals, and different forms of traffic problems.

Research funded by the TNO spearhead program "Sustainable Mobility Intelligent Transport Systems (SUMMITS)" and the TU Delft spearhead program "Mobility of People and Transportation of Goods (TRM)"..

To be able to experiment with different strategies for the application of multi-agent systems for dynamic traffic management and to examine their applicability we need a test bed. Such a test bed facilitates the development of multi-agent systems for dynamic traffic management. The main requirements of the test bed are, that

1. The multi-agent traffic management system can be configured easily.
2. The business logic of traffic engineers can be easily implemented, if possible by the traffic engineers themselves.
3. The multi-agent traffic management system can be evaluated in a realistic simulated traffic environment.
4. The multi-agent traffic management system can be easily transfered to a real-world application

This paper presents a test bed that satisfies the above requirements, and is organized as follows. In Section 2 an overview is given of current, different approaches to decentralized traffic control. The test bed is described in Section 3. Two example scenarios are also described in Section 4, which are meant to demonstrate the flexibility of the test bed in configuring a multi-agent system. We conclude this paper in Section 5 with our future research.

For clarity, a distinction can be made between vehicle-oriented and measure-oriented traffic control:

1. *Vehicle-oriented traffic control* focuses on controlling traffic through vehicle-based (internal) signals, like advanced driver assistance systems.
2. *Road-oriented traffic control* focuses on controlling traffic through roadside-based (external) measures at fixed locations, like traffic signals and variable message signs. This paper focuses on this form of control.

2. Decentralized Traffic Control Concepts

In literature many examples exist where the answer to the dynamic traffic control problem is sought in the form of a traffic control center that monitors the traffic network and performs a global, or network-wide, optimization to set up new parameters for its local controllers. Much of this past work has focused on centralized, and typically predictive, control. Although this approach is very appealing it just is not always possible to do this efficiently and effectively.

Therefore, a partial solution to network-wide traffic control is sought in problem distribution or decentralization. This section discusses the different approaches taken to traffic controller coordination in decentralized control in literature. We look at *hierarchical controller coordination*, *inter-controller coordination* and *intra-controller coordination*.

2.1. Hierarchical Controller Coordination

In order to maintain a network manager's overall control objective, given that part of this control is delegated to local controllers, many authors make use of a

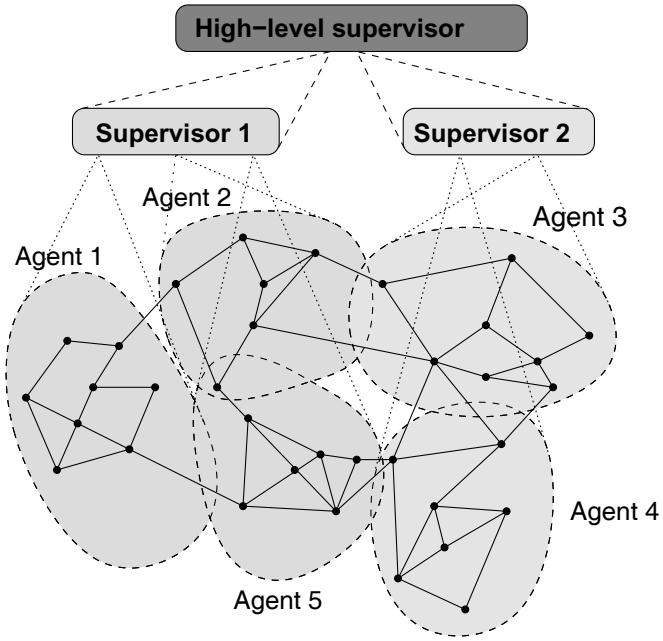


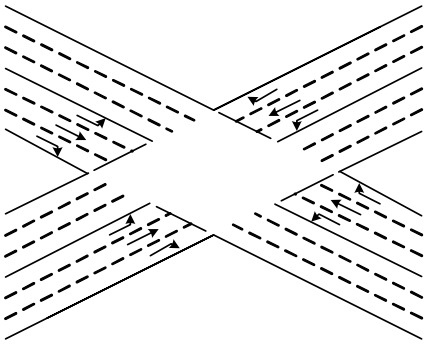
FIGURE 1. Schematic representation of hierarchical controller coordination

hierarchical structure in which higher-level agents are able to monitor lower level agents and are able to intervene when necessary. An example of such a structure is given in Fig. 1.

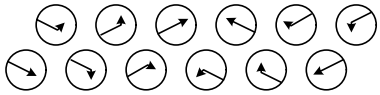
In the *TRYS* model developed by Hernández, Cuena and Molina [10] so-called '*problem areas*' are defined in a particular traffic situation. Each problem area has an agent assigned to it. The agents formulate actions to be performed and propose them to a 'coordinator', who makes a final decision in case of conflicting plans. Choy, Cheu, Srinivasan and Logi [3] introduce middle-level *zone controller agents* and highest-level *region controller agents* to coordinate the actions of the *intersection controller agents* that are present at the lowest level. In both approaches their is no communication between agents at the same level. In [3] France and Ghorbani use only two hierarchical levels, the lower level consisting of *local traffic agents* and a higher level consisting of *coordinator traffic agents*.

2.2. Inter-Controller Coordination

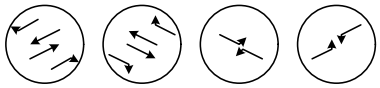
Many applications of intelligent agents in dynamic traffic control aim to make the local controllers more intelligent. The added intelligence aims to make the local controller more susceptible to the interest of the network as a whole. In principle, a local controller works on the basis of local information and can therefore perform only local optimization. In literature many different approaches are taken to overcome this shortcoming.



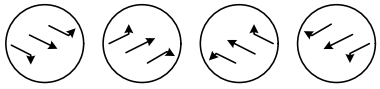
(a) The controlled intersection



(b) Each signal group is represented by an agent



(c) Each phase is represented by an agent



(d) Each approach arm is represented by an agent

FIGURE 2. Intra-controller coordination

The most common approach is to share information among controllers. Another approach is to make the road-infrastructure responsible for controller-coordination. It is after all in the interest of the road segment that connects two intersections that the control actions at these intersections are coordinated. This is the approach we take in [22].

In most cases information is only shared upstream. In fewer cases [6, 7, 14] this information is also shared downstream. The sharing of information can be done on the level of:

- operational information (often raw detector data),
- tactical information (processed, derived data), and
- strategic information (expert advice).

2.3. Intra-Controller Coordination

The task of controlling a single, isolated intersection is often perceived to be one, undividable, centralized control problem, but numerous examples exist in which the control of a single intersection is stated as the result of a negotiation process between multiple intelligent agents, each having their own control objective. These agents can either represent the individual signal groups [11, 12], phases [23] or the arms of the intersection as shown in Fig. 2.

To date, no literature can be found that compares the merits and downsides of each of the chosen approaches to traffic controller coordination. The test bed will enable us to make this comparison.

3. Components of the Test Bed

In traffic control, two processes can be distinguished. First of all, there is the *traffic process*. This process can be observed by means of monitoring equipment (e.g. induction loop detectors and floating-car data) and influenced by traffic control instruments (e.g. variable message signs, ramp metering installations, and traffic signals), which form the working material for the traffic control process. The second process, namely the *control process*, is, in the case of our test bed, comprised of multiple interacting intelligent agents.

This section discusses how the test bed is set up. The test bed consists of an *interaction model*, *intelligence models*, and a *world model*. These models are presented in the next subsections. For a more detailed discussion of the material, we refer to [19].

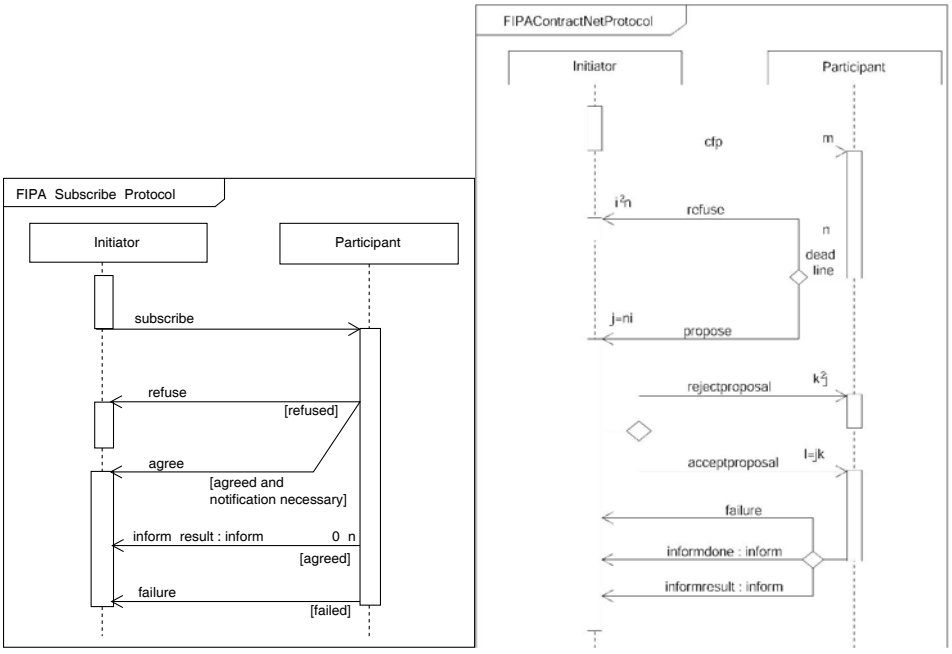
3.1. Interaction Model

The interaction model is used to model the interactions between the agents. All communications in our test bed conform to the specifications as set by the Foundation for Intelligent Physical Agents (FIPA) [8], an approach also taken in [2] for a video-based traffic monitoring system. The core message of FIPA is that through a combination of speech acts, predicate logic and public ontologies, standard ways can be offered of interpreting communication between agents in a way that respects the intended meaning of the communication. Ontologies provide the vocabulary for representing and communicating knowledge about a topic and a set of relationships and properties that hold for the entities denoted by that vocabulary.

The FIPA standards require that each agent publishes the services it provides to a Directory Facilitator. This Directory Facilitator is a component of the multi-agent system that provides a yellow-pages directory service to agents. At least one directory facilitator must be present in the multi-agent system. The presence of a directory facilitator enables a dynamic configuration of the agent system. This way the location of a service an agent needs for its own operation does not have to be hard coded in the agent, but can be found at run-time though means of the Directory Facilitator.

FIPA's standard interaction protocols and communicative acts are currently sufficient for our purposes. Examples of these are the subscription interaction protocol (Fig. 3(a)), iterated contract net interaction protocol(Fig. 3(b)) (for negotiations), the propose interaction protocol and the request interaction protocol, all of which we need for our cooperating traffic agents. For this we rely on the JADE agent development environment [1].

Currently there is no ontology available at FIPA that relates to the domain of traffic control. Our aim is to construct a broadly applicable ontology for traffic control through our use of the test bed for different cooperative traffic control problems.



(a) FIPA subscription interaction protocol (b) FIPA contract net interaction protocol

FIGURE 3. FIPA interaction protocols

3.2. Intelligence Model

The intelligence models are used to model an agent's intelligence. A fundamental decision in defining a problem is deciding how to model it. Sometimes experience is available to aid in choosing the best paradigm. Often a paradigm is selected on the basis of the applicant's familiarity with it. This is why conventional programming paradigms are often considered first. The test bed we have designed allows programming the intelligence model using a conventional programming paradigm using the C(++) and Java languages, but is not limited to these languages. The dynamic traffic management domain has always been open to unconventional approaches from the field of artificial intelligence, including evolutionary algorithms, knowledge-based systems, neural networks and multi-agent systems [5, 13, 15, 17, 18, 24].

Currently the test bed supports the following approaches:

1. *Rule-based inference*
   For our test bed we developed a generic rule-based agent using *JESS*, a rule-based reasoning engine [9]. Incoming messages are converted to facts and asserted into its working memory. Derived facts describing messages to be sent are translated into corresponding FIPA messages, after which JADE

Business rule
If there is a route named *?route* that has an alternative named *?route-alt* for which the quality of traffic flow is higher **Then** direct traffic from the former route to the latter route using the following message ....

JESS rule

```
(defrule take-action
  (RouteAlternative
     (route ?route)
     (alternative ?route_alt))
  (Route
     (name ?route)
     (quality ?quality))
  (Route
    (name ?route_alt)
    (quality ?quality_alt&:(> ?quality_alt ?quality))
=>
  (assert
  (VMSSignal
    (text "Congestion on route:     " ?route
          "Please take alternative: " ?route_alt)))
)
```

TABLE 1. Rule-based intelligence

takes care of their delivery. The rule-based agent is typically used to program the expertise of a human expert and is as such an ideal prototyping and training tool for traffic managers [21]. Decision rules in an expert system take the form of simple IF-THEN statements. A simplified example of an IF-THEN statements as used for network traffic management is given in Table 1.

2. *Bayesian inference*
   For our test bed we have developed a generic Bayes agent using *JavaBayes*, a set of tools for the creation and manipulation of Bayesian networks [4]. Bayesian inference is an approach to statistics in which all forms of uncertainty are expressed in terms of probability. Traffic control is guided by uncertainties, namely uncertainties regarding the current traffic state (due to limits in the amount and quality of the available monitoring data), uncertainties regarding the progression of the current traffic state (due to limits in traffic forecasting models), and uncertainties about the effects of a control

action on traffic flow. Bayes' law describes how events from the past influence the probability of those events that are still about to happen (or not). Because the theorem explicitly describes the relation between events, causality and their context, it is an excellent mathematical foundation for automated traffic control [20].

Incoming messages are converted and assigned to variables in the network. The set of variables that has assigned values is called evidence. The resulting expectations corresponding to messages to be sent are translated into corresponding FIPA messages, after which JADE takes care of their delivery.

3.3. Virtual World

We use the microscopic traffic simulation package *Paramics* developed by Quadstone [16] as our traffic model. Fig. 4 contains a screen-shot of the model. Paramics simulates traffic at the level of individual vehicles. Our prime motivation for choosing for Paramics for our test bed is that it can be programmatically extended through an application programming interface. The application programming interface allows the agent-controllers to retrieve information from and to enter control actions into the simulation environment. Paramics was one of the first models providing this capability. The test bed can however easily be modified to also use other traffic models that provide a suitable application programming interface. Traffic simulation models often employ a fixed-time step based method to simulation. Paramics is no exception. It is possible to retrieve detector data and modify the actuators in-between these time steps, which is shown schematically in Fig. 5. The agents in a multi-agent system in contrast operate in continuous time. In order to bridge this gap, the Paramics-World Interaction Agent stores the request and subscriptions from other agents until it is time to continue to the next time step. The decision to go to the next time step depends on the type of synchronization one wishes to apply. Since the traffic system is simulated using a single simulation process, there is only *one* agent that handles *all* outside world requests and subscriptions. In the real world each detector and actuator could in principle be represented in the multi-agent system by a specific agent. This is however a theoretical deployment scenario, which will be difficult to attain in practice. Traffic control centers are often equipped with different control applications, each representing a group of detectors or actuators from one manufacturer. A more realistic deployment scenario is that these applications are retrofitted with an agent wrapper.

Simulation is used to test various real-world application scenarios for multi-agent systems. In order to test whether the configured multi-agent system will function under real-time conditions synchronization can be performed by slowing down the simulation such that simulation time equals wall-clock time. However, the test bed will typically be used to test the performance of a configured multi-agent system with regard to traffic flow. In that case it is required that the multi-agent

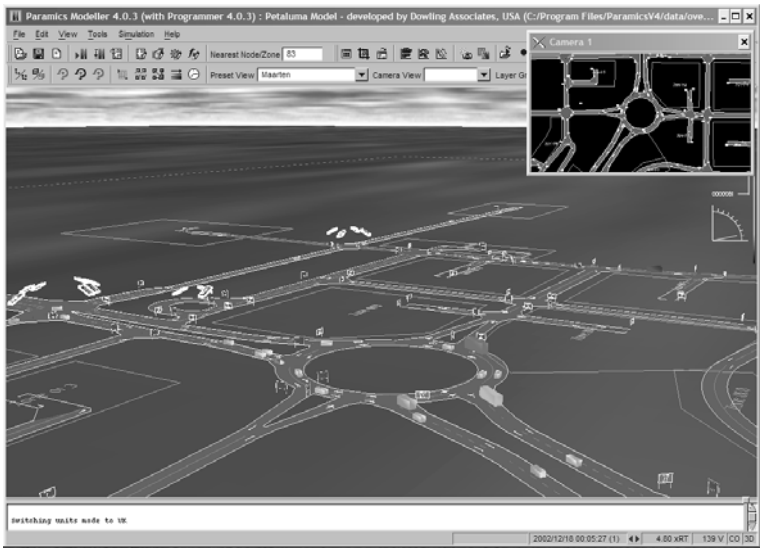


FIGURE 4. Screen-shot of Paramics

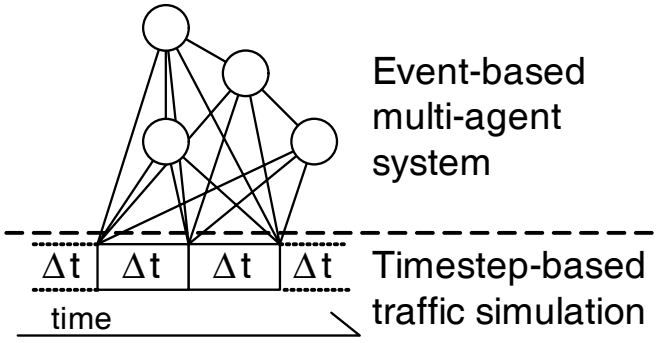


FIGURE 5. Interaction in discretized time

system gets sufficient time to formulate the control actions and that the simulations are repeatable.

In order to guarantee that the multi-agent system is given sufficient time to formulate the control actions, it has to be determined when the agents in the multi-agent system have finished formulating their control actions. This is done using a special purpose agent, that when present, requires an agent to report when it wants to change its state from busy to idle. This special purpose agent is named MAI, short for Maintainer of Agent Information. When all agents have reported to be idle, and thus all information on the basis of which control actions can be

formulated has been processed by the multi-agent system, is the simulation allowed to continue.

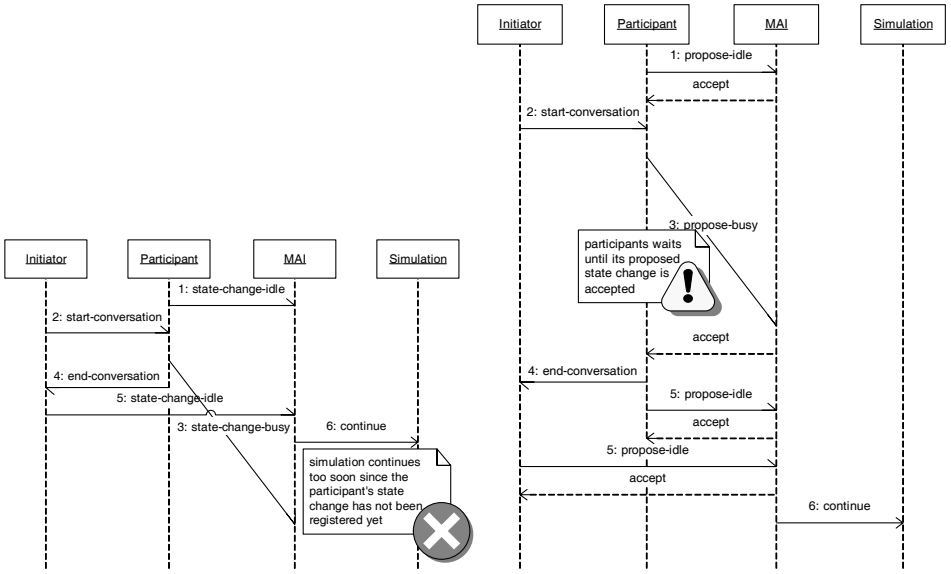
The FIPA propose interaction protocol is used to convey an intended state change from an agent to the maintainer of agent information. This protocol mandates that proposals are explicitly accepted or refused. In this case the latter will never be the case. The explicit acceptation is needed since there is no way to guarantee that messages from the agents informing about an intended state change arrive in the order they are sent. Without explicit acceptation the simulation can sometimes be allowed to continue before the agents are done formulating their control actions. An example of this is shown in Fig. 6(a). In this figure an initiator initiates a conversation (message 2) with a participant which has previously registered itself as being idle (message 1). In order to participate in the conversation, the participant has to register itself as being busy (message 3). If this message is delayed it is possible that the message arrives at the maintainer of agent information at a moment when the initiated conversation has already ended (message 4) and the initiator's has registered itself as being idle (message 5). In this case the simulation is allowed to continue while the participant might still be busy formulating its control actions. Fig. 6(b) shows the same communication trace where each intended state change is explicitly accepted. In this case it is guaranteed that the simulation continues only when all agents are done formulating their control actions.

When agents are mandated by the maintainer of agent information to communicate intended state changes, all agents operate following the higher level state chart as shown in Fig. 7. In this figure the busy state is a composite state which encompasses the regular state charts of the agent when it is operating in unsynchronized mode. When an agent intends to change state from idle to busy or vice versa it communicates this intent to the maintainer of agent information though means of a propose-message. When the propose-message is sent, the agent enters an intermediate pre-idle or pre-busy state. The agent remains in this intermediate state until it receives an accept-proposal message from the maintainer of agent information in reply to the proposed state change. Only then is the agent allowed to change its state to either the busy or the idle state.

4. Application Scenarios

From the perspective of the multi-agent system developer, the following three phases can be distinguished:

1. Configuration phase. The user creates the traffic situation to be simulated and the interacting multi-agent system.
2. Simulation phase. The simulation is started, simulation data is collected and control actions are derived and communicated by the multi-agent system.
3. Analysis phase. The simulation data is analysed using the tools provided for this purpose by the traffic simulation model.



(a) State changes are not confirmed: Unsuccessful synchronization

(b) State changes are confirmed: Successful synchronization

FIGURE 6. Synchronization sequences

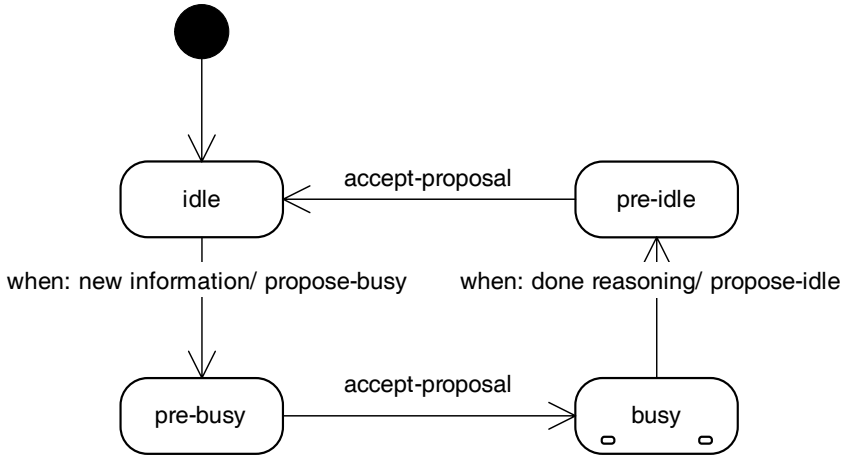


FIGURE 7. States and transitions needed to ensure synchronized operation

A special XML file lists the agents that make up the multi-agent system. Adding an agent is done by adding a line containing its name and the location of the agent-specific knowledge. The next step is to actually construct the JESS files containing the knowledge. When the simulation is started, the XML file is parsed, the listed agents are created in JADE, and they are provided with the corresponding agent-specific knowledge.

To demonstrate the flexibility of the test bed in configuring a multi-agent system, we implemented two alternative multi-agent system configurations for the two examples of [22] describing:

1. The coordination of traffic management instruments on a freeway corridor, and
2. The coordination of traffic management instruments in a network.

For both examples traffic control was at first delegated to agents representing each part of the road infrastructure. In this configuration information is exchanged between agents representing neighboring infrastructure elements. From a functional perspective it seems natural to have the agents directly represent the road infrastructure. However, from a more practical perspective it is more natural to have the agents represent the traffic control instruments present in the network, since the technical infrastructure for this kind of delegation is largely already there. We therefore subsequently converted the multi-agent traffic management system into a system in which traffic control was delegated to agents representing the traffic control instruments. In that case information is exchanged between traffic control instruments that are in relative close proximity of oneanother.

The directory facilitator in combination with the XML file allowed us to easily configure the infrastructure-based and controller-based agent implementations once the agent-intelligence was defined. The files containing the agents' knowledge were assigned to different agents depending on the configuration chosen. The agents' knowledge itself could remain unchanged. The ease with which the two agent configurations were configured, showed that the system enables users to create and experiment with different multi-agent control structures for dynamic traffic management. The two alternative multi-agent configurations for each of the two examples of [22] are described below.

4.1. Scenario 1: Coordination on a Freeway

To alleviate traffic congestion due to recurrent congestion, ramp metering has been applied throughout the world. Ramp metering aims to limit the number of vehicles entering the freeway so that freeway flow in the ramp metering installation's area of influence can be maintained at the desired quality level. Excess demand is forced to wait at the entrance ramp. The intention of ramp metering is, therefore, to maintain uninterrupted, non-congested flow on the freeway as long as possible by transferring delay from the freeway to the entrance ramp.

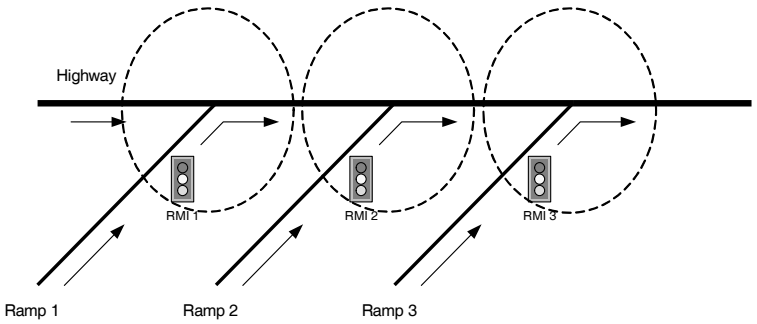
In Fig. 8(a) a corridor is represented. In this representation traffic drives from the left side of the picture to the right side. Traffic enters the freeway represented

in the figure from the freeway upstream (left of the picture) and the three on-ramps (numbered 1, 2 and 3). Each on-ramp is metered by a ramp metering installation. Each ramp metering installation acts upon the information it receives from the measurement loops directly up- or downstream of the on-ramp (depending on the chosen implementation). As such, no matter what the objective of a ramp metering installation may be, it will always be local.

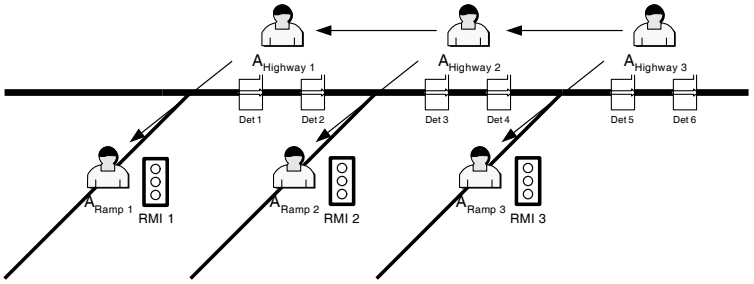
Coordinated ramp metering refers to the application of ramp control to a series of ramps where the interdependency of ramp operations is taken into account. The primary objective of integrated ramp control is to prevent or reduce the occurrence of congestion on a longer stretch of freeway. Therefore, the control of each ramp is based on the demand-capacity considerations for the whole stretch rather than on the demand-capacity constraint at each individual ramp. Whenever traffic demand on the freeway near on-ramp 3 approaches the capacity of the road in the bottleneck, the ramp-metering installation at on-ramp 3 will maximally delay traffic on the on-ramp. In case traffic demand on the freeway near on-ramp 3 amply exceeds the capacity of the road in the bottleneck then a queue will start forming. This queue will rapidly build up to on-ramp 2. As soon as this happens the metering installation will want to maximally delay traffic on its on-ramp. Too late, since the queue has already formed and has probably reached on-ramp 1. The situation illustrated above could have been delayed as long as possible or could have been prevented altogether when the actions of the ramp-metering installations would have been coordinated.

As soon as the ramp-metering installation at on-ramp 3 notices that the traffic demand on the freeway near on-ramp 3 is going to approach the capacity of the road in the bottleneck, it could have called upon the ramp-metering installation at on-ramp 2. The ramp-metering installation at on-ramp 3 can have a pretty good idea of how much traffic is going to arrive based upon the information it can get from the ramp-metering installations upstream and it's own experience. Instead of waiting for the problem to occur (the queue forming at the bottleneck) the problem can thus be pro-actively combated.Whenever a ramp-metering installation calls upon a ramp-metering installation upstream, the ramp-metering installation that is called upon can try to meet this request as good as it can. If it fails to fully meet the request it can then itself call upon a ramp-metering installation further upstream.

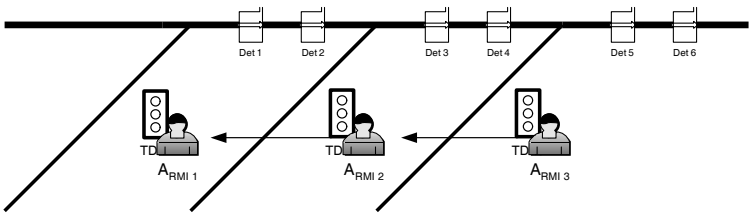
Although the example above is certainly an improvement compared to the example without co-ordination it still isn't exactly ideal. Traffic that wants to enter the freeway using on-ramp 3 for example is put at a disadvantage compared to the traffic that wants to enter the freeway using on-ramp 2. Traffic that wants to enter the freeway using an on-ramp further upstream is generally put at a disadvantage in this example.By allowing the individual ramp-metering installations to negotiate about how much traffic they plan to let through, they can also take care that the delay imposed at the traffic at the on-ramps is evenly distributed.



(a) Traffic network used in the scenario



(b) Coordination through infrastructure agents



(c) Coordination through controller agents

FIGURE 8. Different multi-agent control system structures

Using the test bed, we have developed a multi-agent system, capable of communicating about the required amount of metering. As a result of this communication, the ramp metering installations are better aware of future congestion. In this way, they can react in advance and solve the congestion in a more timely way, resulting in a 5 percent improvement in freeway travel time [19].

In the agent implementation of Fig. 8(b) traffic control is delegated to the infrastructure. Information is exchanged between neighboring infrastructure agents. Another, and more practical way of delegation is to delegate traffic control to the ramp-metering installations (the control instruments) as shown in Fig. 8(c). In

that case information is exchanged between the ramp-metering controller agents. The arrows indicate the communication possibilities of each agent. In this way the knowledge about the current traffic situation propagates upstream through the network.

4.2. Scenario 2: Coordination on a Network

Variable Message Signs (VMS) are programmable traffic control devices that display messages composed of letters, symbols or both. They are used to provide information about changing conditions in order to improve operations, reduce accidents, and inform travelers. They may advise or urge drivers to change travel speed, change lanes, divert to a different route, or simply to be aware of a change in current or future traffic conditions.

A VMS system can directly benefit:

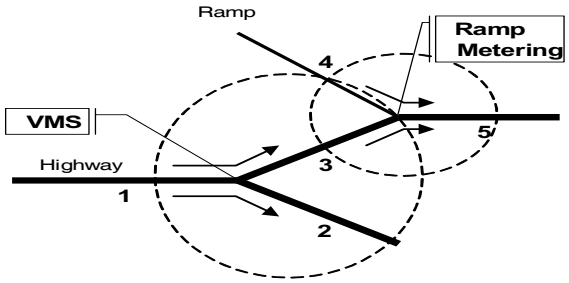
- Routing choice, saving vehicle traveled miles and hours, and
- Congestion reduction.

The objective of a VMS depends on the application, for this example we'll focus on the application of a VMS for route guidance.

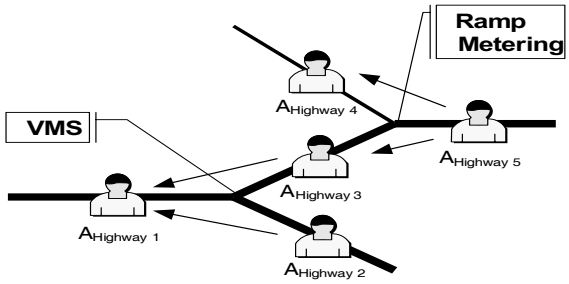
In Fig. 9(a), a small freeway network is represented. The example given below focuses on the traffic that travels from the left side to the right side of the picture. The main arteries (the arteries denoted with the numbers 1, 2, 3 and 5) are drawn fatter than the regular arteries (artery number 4). Traffic enters the network by means of a main artery (1) and a regular artery (4). Main artery 1 branches off into main arteries 2 and 3. Road users are informed just before this branching point about the traffic conditions on both branches by means of a VMS. A ramp-metering installation meters the traffic that wants to enter the main artery (5) using the regular artery (4).

In case a queue starts forming caused on freeway 5 the ramp-metering installation will only start metering when the end of the queue approaches the junction of arteries 3,4, and 5. This will somewhat slow down the growth of the queue on main artery 3. The road user will be informed by the VMS at the junction of arteries 1, 2, and 3 about the worsening traffic conditions at main artery 3. This will slow down the growth of the queue even more. When the end of the queue reaches the junction of arteries 1,2, and 3, traffic will be faced with the consequences of the incident that took place on freeway 5 that never planned to take a route including freeway 5 in the first place. The queue will then start to grow rapidly since road users wanting to follow freeway 2 will be blocked passage by road users wanting to follow freeway 3.

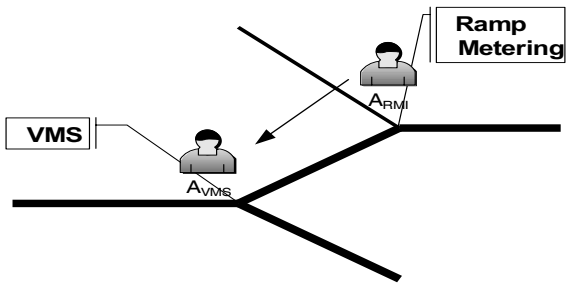
Using the test bed, we have developed a multi-agent system, capable of coordinating the actions of the ramp-metering installation and the VMS. As a result of this coordination, we were able to further delay the situation illustrated above and even prevent the situation from occuring altogether. As soon as the density of traffic on the main artery near the ramp-metering installation approaches a level that entails that traffic entering the main artery from artery 4 will have to be



(a) Traffic network used in the scenario



(b) Coordination through infrastructure agents



(c) Coordination through controller agents

FIGURE 9. Different multi-agent control system structures

significantly delayed, the ramp-metering installation calls upon the VMS to start rerouting traffic via freeway 2. By combining the efforts of both the ramp-metering installation and the VMS the growth of the queue can be delayed significantly. By delaying the growth of the queue it will take longer for the end of the queue to reach the junction of arteries 1,2, and 3. The queue might even dissolve before it reaches the junction. This way, road users that want to follow freeway 2 can do so as long as possible without being blocked passage.

In the agent implementation of Fig. 9(b) traffic control is delegated to the infrastructure. Information is exchanged between neighboring infrastructure agents. Another, and more practical way of delegation is to delegate traffic control to the traffic control instruments as shown in Fig. 9(c). In that case information is exchanged between the ramp-metering controller agents. The arrows indicate the communication possibilities of each agent. In this way the knowledge about the current traffic situation propagates upstream through the network.

5. Conclusions and Future Research

Our contribution to the research in the road-traffic management domain is the development of a software environment for rapid development of multi-agent control systems. In this paper a test bed for agent-based road traffic management is presented. The organization of the software is discussed, as well as the role of rule-based and Bayesian reasoning. The implementation of two multi-agent systems with the test bed is described. The implementations show that multi-agent systems can be created easily.

The presented test bed will be of great value for developments in traffic management. The developed system still has opportunities for further extension. A graphical user interface can be developed in which agents can be created and the the multi-agent system can be configured with only a few mouse clicks. This would further accelerate the implementation of the desired multi-agent system. Extending the number of available intelligence models could be another improvement.

With the test bed, a tool has been developed to study the possibilities of applying multi-agent systems in dynamic traffic management. It is a starting point for our further research in decentralized traffic control. E.g. the different decentralized control concepts described in Section 2 can be implemented using the test bed. Another line of research regards the scalability properties of multi-agent based decentralized control versus centralized control.

References

1. Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa, *Jade programmer's guide*, TILab S.p.A., July 2002, URL: http://sharon.cselt.it/projects/jade/.
2. Luís Botelho, *A control structure for agent interaction*, Proc. IEEE Intelligent Vehicles Symposium (IVHS'00), October 2000, pp. 398–403.
3. Min Chee Choy, Ruey Long Cheu, Dipti Srinivasan, and Filippo Logi, *Real-time coordinated signal control using agents with online reinforcement learning*, Proc. 83rd Annual Meeting of the Tranportation Research Board (TRB'03), November 2003.
4. Fabio Gagliardi Cozman, *Bayesian networks in java*, URL: http://www-2.cs.cmu.edu/~javabayes, May 2003.
5. J. Cuena, J. Hernández, and M. Molina, *Knowledge-based models for adaptive traffic management systems*, Transportation Research Part C **3** (1995), no. 5, 311–337.

6. Enrique D. Ferreira, Eswaran Subrahmanian, and Dietrich Manstetten, *Intelligent agents in decentralized traffic control*, Proc. IEEE Intelligent Transportation Systems (ITSC'01) (Oakland (CA), USA), IEEE, August 2001, pp. 707–711.
7. Nicholas V. Findler, Sudeep Surender, and Ş. Catavra, *A semi-autonomous decentralized system for controlling street traffic signals*, Proc. IEEE Autonomous Decentralized Systems, April 1995, pp. 377–383.
8. FIPA, *The foundation for intelligent physical agents*, URL: http://www.fipa.org, 2003.
9. Ernest Friedman-Hill, *Jess, the expert system shell for the java platform*, URL: http://herzberg.ca.sandia.gov/jess/, May 2003.
10. Josefa Hernández, José Cuena, and Martín Molina, *Real-time traffic management through knowledge-based models: The TRYS approach*, ERUDIT Tutorial on Intelligent Traffic Management Models (Helsinki, Finland) (ERUDIT, ed.), 1999.
11. Sandy Irani and Vitus Leung, *Scheduling with conflicts, and applications to traffic signal control*, Proc. ACM-SIAM Symposium on Discrete Algorithms, January 1996, pp. 85–94.
12. Iisaki Kosonen, *Multi-agent fuzzy signal control based on real-time simulation*, Transportation Research Part C **11** (2003), 389–403.
13. B. Krause and C. von Altrock, *A complete fuzzy logic control approach for existing traffic control systems*, Mobility for Everyone, Proceedings of the 4th World Congress on Intelligent Transportation Systems (Berlin, Germany), October 1997, Paper no. 2045.
14. Jia Lei and Ü. Özgümer, *Decentralized hybrid intersection control*, Proc. IEEE Decentralized Hybrid Intersection Control, vol. 2, December 2001, pp. 1237–1242.
15. M. Molina, J. Hernández, and J. Cuena, *A structure of problem-solving methods for real-time decision support in traffic control*, International Journal of Human-Computer Studies **49** (1998), no. 4, 577–600.
16. Quadstone Ltd., *Paramics: Microscopic traffic simulation*, URL: http://www.paramics-online.com/, May 2003.
17. S.G. Ritchie, *A knowledge-based decision support architecture for advanced traffic management*, Transportation Research Part A **24** (1990), no. 1, 27–37.
18. A.W. Sadek, M.J. Demetsky, and B.L. Smith, *Case-based reasoning for real-time traffic flow management*, Computer-Aided Civil and Infrastructure Engineering **14** (1999), no. 5, 347–356.
19. Alexander Th. van den Bosch and Maarten R. Menken, *Multi-agentsystemen en verkeersbeheersing – De ontwikkeling en demonstratie van een testbedomgeving*, Tech. Report 03-7N-109, TNO, Delft, May 2003, MSC Thesis (in Dutch), Vrije Universiteit, Amsterdam, The Netherlands.
20. Sicco Pier van Gosliga, Paul van Koningsbruggen, and Ronald van Katwijk, *Bundling the available knowledge on local traffic situations in network wide traffic management*, Submitted to the 11th World Congres on Intelligent Transportation Systems (ITS'04), 2004.
21. Ronald van Katwijk, Paul van Koningsbruggen, and Marcel Westerman, *On the application of an expert system for network traffic management*, Submitted to the 11th World Congres on Intelligent Transportation Systems (ITS'04), 2004.

22. Ronald T. van Katwijk and Paul van Koningsbruggen, *Coordination of traffic management instruments using agent technology*, Transportation Research Part C: Emerging Technologies **10** (2002), no. 5–6, 455–471.
23. Liu Xiaoming and Gong Xiaoyan, *Study of intersection traffic flow control on the basis of agents*, Proc. IEEE Intelligent Transportation Systems (ITSC'03), vol. 2, October 2003, pp. 1755–1758.
24. H. Zhang and S.G. Ritchie, *Real-time decision-support system for freeway management and control*, Journal of Computing in Civil Engineering **8** (1994), no. 1, 35–51.

R.T. van Katwijk
P. van Koningsbruggen
Netherlands Organization for Applied Scientific Research (TNO)
2600 JA Delft
The Netherlands
e-mail: {r.vankatwijk, p.vankoningsbruggen}@inro.tno.nl

B. De Schutter
J. Hellendoorn
Delft Center for Systems and Control (DCSC)
Delft University of Technology
Mekelweg 2
2628 CD Delft
The Netherlands
e-mail: {b.deschutter, j.hellendoorn}@dcsc.tudelft.nl

Collaborative Driving System Using Teamwork for Platoon Formations

Simon Hallé and Brahim Chaib-draa

Abstract. Collaborative driving is a growing domain of Intelligent Transportation Systems (ITS) that makes use of communications to autonomously guide cooperative vehicles on an Automated Highway System (AHS). In this paper, we address this issue by using a platoon of cars considered as more or less autonomous software agents. To achieve this, we propose a hierarchical architecture based on three layers (*Guidance* layer, *Management* layer and *Traffic Control* layer), which can be used to develop coordination models for centralized platoons (where a head vehicle-agent coordinates other vehicle-agents by applying its coordination rule) and decentralized platoons (where the platoon is considered as a team of vehicle-agents trying to maintain the platoon). The latter decentralized model mainly considers a software agent teamwork model using architectures like STEAM. These different coordination models will be compared using results on preliminary simulation scenarios, to provide arguments for and against each approach.

Mathematics Subject Classification (2000). Primary 99Z99; Secondary 00A00.

Keywords. Collaborative Driving, Intelligent Vehicles, Teamwork, Multiagent, Platoon, Intelligent Transport System.

1. Introduction

Transport systems all over the world are suffering from spreading problems regarding mainly their traffic flow and safety. To address these traffic problems, we generally build more highways, but this solution is greatly limited by the available land areas, which is running low in most cosmopolitan cities. An alternative solution which is growing in popularity is to develop techniques that increase existing roads' capacity by investing in Intelligent Transportation Systems (ITS) infrastructure [7]. It is shown that ITS may provide potential capacity improvements

This work was carried out as part of the Automobile of the 21st Century (Auto21) project supported by the Government of Canada through the Networks of Centres of Excellence (NCE).

as high as 20 percent [14]. We can cite as ITS components: advanced transportation management, advanced transportation information system, and commercial vehicle operations. Among these components, there are sub-components such as automobile collision avoidance and electronic guidance system, which are generally sustained by individual technologies as: electronic sensors, wire and wireless communications, computer software and hardware, GPS, GIS, etc. To sum up, ITS has the advantage of enhancing safety and reducing congestion, which results in a reduction of transport systems' negative environmental impacts.

Collaborative driving is an important sub-component of ITS that strives to create vehicles being able to cooperate in order to navigate through highway traffic using communications. Such a system is made possible with the collaboration of a lower layer of control system, which acts as an Adaptive Cruise Control (ACC) [11]. Thus, at its simplest implementation, collaborative driving adds a layer of communication to the present ACC, to create a Cooperative Adaptive Cruise Control (CACC) and benefit from a communication system to collaborate between vehicles' ACC [18]. Going forward to a wider level of collaboration, the vehicle platoon model, has used communications to coordinate platoon members with their platoon leader [16]. Compared with CACC, this vehicle organization adds a deliberative system in the lead vehicle, which coordinates the preceding vehicles equipped with CACC, to maintain the platoon formation. As a new approach to centralized platoon models based on the leader, we aim to incorporate the multiagent vision to the platoon architecture and coordinate the vehicles through a model of teamwork for software agents [15]. Such an approach incorporates autonomous agents in each vehicle to use their respective communication system and coordinate each others in a decentralized platoon model.

In this paper, we address in Section 2 the coordination issues for a platoon of vehicles, by first describing the collaborative driving domain and the simulator used to represent this environment. Then, Section 3 presents the hierarchical architecture we adopted as the driving system of automated vehicles. Section 4 describes the different coordination strategies we implemented and tested in the previous simulator. Section 5 reports our preliminary results using the comparison of centralized coordination models with decentralized ones, by putting the emphasis on agent teamwork. Finally, Section 6 presents a discussion, followed by the conclusion.

2. Application Domain

Collaborative driving is a research domain which aims to create automated vehicles that collaborate in order to navigate through traffic. In this sort of driving, one generally form *a platoon* [17], that is a group of vehicles whose actions on the road are coordinated by the means of communication. The first vehicle of a platoon is called the platoon leader and its role is to manage the platoon and guide it on the road, so other vehicles can simply follow it. Our work comes as part of the

Auto21 project [3] studying the automobile of the 21st century within three levels of system functionality. The first level focuses on the vehicle's longitudinal control, while the second focuses on its lateral control, both in a platoon lead by a human driver. Finally, the third level considers every vehicle (even the leader) as fully autonomous. This article focuses on the first level of functionality and its usage of communications within the different platoon driving manoeuvres.

2.1. Microsimulation of Autonomous Vehicles

The environment in which our vehicle coordination system has been tested is a Collaborative Driving System (CDS) [5] simulator developed to provide user interface and graphical results of our work. Similar to California Path's Smart AHS [1], our simulator called HESTIA, aspires to a lower level of vehicle microsimulation, as its main purpose is to create an environment for the development and testing of Intelligent Transport Systems (ITS). To do so, it simulates a highway environment, with vehicles represented as 3D shapes. These vehicles are using simulated dynamics and sensors to retrieve information from the environment, such as the vehicle's internal dynamic information and external vehicles' dynamic information. The simulator's environment is based on JAVA 3D™ 's technology that offers a 3D environment in which autonomous vehicles can evolve.

The simulated vehicles' model includes longitudinal and lateral vehicle dynamics, wheel model dynamics, engine dynamics, torque converter model, automatic gear shifting and throttle/brake actuators. The simulated sensors were developed using the 3D engine of JAVA 3D™ , and in the vehicle model presented in this paper, each following vehicle is equipped with a vehicle-based laser sensor. This sensor provides information on the front object's (a vehicle) distance and difference of velocity, for distances up to $100m$, using an abstract model of laser. The second type of sensor, used for high-level navigation, is a Global Positioning System (GPS), which gives real-time information on the vehicle's position (latitude, longitude), mapped in a two dimensions system. Finally, we simulated a radio transmitter/receiver onboard each vehicle for two ways point to multipoint communications. We will not go further on a detailed representation of the simulator's components, as it is out of scope for this paper, but the readers can refer to [8] for more information.

2.2. Simulated Driving Scenarios

The primary goal of the CDS is to maintain the platoon formation stable, and therefore, the two simulated scenarios we focus on are the two main disturbance in this formation: a vehicle splitting and a vehicle merging the platoon. These two scenarios are represented in Figure 1 and they can be detailed as follows for a better understanding:

A Vehicle splitting happens when a vehicle member of a platoon decides to leave it, thereby forming two non-empty platoons. To execute this manoeuvre, the splitting vehicle (*F2* in Figure 1) must communicate its intention of leaving the platoon, so the platoon formation modifies the distances at the front and

rear of the splitting vehicle, as shown in *step 1* (*S1*) of Figure 1. When this new formation gains stability, the splitting vehicle *F2* can change lane, while the rest of the platoon followers keep the same distance. When the splitting vehicle has safely left the platoon (*S2*), the gap created for its departure can be closed, thus forming back the precedent platoon, minus one vehicle (*S3*).

A Vehicle merging is the exact opposite of a split manoeuvre: two non-empty platoons merge together to become one. To execute this manoeuvre, the merging vehicle must be part of a platoon formed of one vehicle (itself), like vehicle *L2* in Figure 1. The merging vehicle first has to communicate to another platoon a query to join it. In the example of Figure 1, in *step 1* (*S1*), the platoon lead by vehicle *L1* accepts *L2*'s query. Moving from *S1* to *S2*, *L1*'s platoon reacts by creating a safe space and communicates the dynamic position of this space to the merging vehicle. The merging vehicle then modifies its velocity to join the meeting point, verifies if it is safe to merge and changes lane to enter the platoon formation and leave *S2* to meet *S3*. Once the merged vehicle has stabilized its inter-vehicle distance, the platoon can reach its precedent formation plus one vehicle, by diminishing the distances with the new vehicle. Although the steps of the merge manoeuvre may differ from one coordination model to another, this represents the general pattern of the merge manoeuvre.

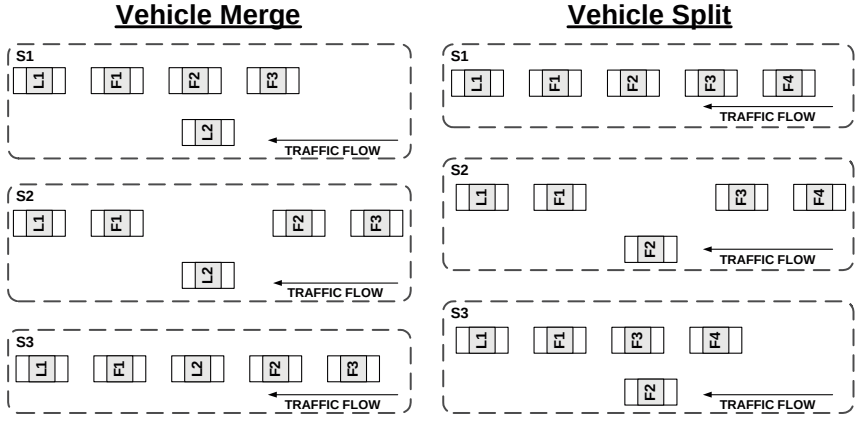


FIGURE 1. The three steps of the removal (split) and insertion (merge) of a vehicle in the platoon.

As it has been shown, the merging and splitting manoeuvres of a vehicle are the most problematic cases of the platoon formation and this is why the different coordination models presented in Section 4 focus on the communications involved during these two tasks. Since we put the emphasis on the communication and coordination of the platoon, the vehicles' lateral automated control is simulated to enable us to perform the lane changes involved in the merge and split manoeuvres. The lateral guidance system of our current system could then be seen as the

simulation of the human driver's steering behavior or as the first phase of the lateral guidance system. This subject being out of scope for this paper, which focuses on communications, we will not detail the lateral controller.

3. Hierarchical Architecture for Collaborative Driving

The architecture we adopted for our driving system is based on a hierarchical approach. This model uses a more reactive system as the bottom of the architecture and moves forward to a more deliberative system as it raises to the upper levels. Finally, as we related to other collaborative driving models, our hierarchical architecture was also inspired by concepts coming mainly from the PATH project [12]. As indicated in Figure 2, the resulting architecture has three major layers: *Guidance* layer, *Management* layer and *Traffic Control* layer.

The *Guidance* layer has the function of sensing the conditions and states ahead and around the vehicle and activating the longitudinal or the lateral actuators. For the *Intelligent Sensing* sub-layer, the inputs come from sensors for speed, acceleration, raw rate, machine vision, etc. The *Guidance* layer outputs sensing data and vehicles state variables to the vehicle *Management* layer, which in turn sends back "desired state" queries in the form of steering and vehicle velocity queries to the *Guidance* layer. These queries are finally applied by the *Vehicle Control* sub-layer, which includes lateral controllers and the longitudinal controllers developed by our partners at Sherbrooke University [10].

The *Management* layer determines the movement of each vehicle under the cooperative driving constraints using data from: (a) the *Guidance* layer; (b) vehicles coordination constraints through the inter-vehicle communication; and (c) the *Traffic Control* layer through the road-vehicle communication. To determine the movement of each vehicle under the cooperative constraints, the *Management* layer needs to reason on the place of the vehicle in its platoon when the vehicle stays in the same lane (intra-platoon coordination), and its place in a new platoon when the vehicle should change lane (inter-platoons coordination). The first type of coordination is handled by the *Networking* module and the second by the *Linking* module, together forming the *Coordination* sub-layer. Generally, the task of the *Linking* module is to communicate with the *Traffic Control* layer to receive suggestions on actions to perform. Resulting from these suggestions, the architecture's *Linking* module reasons about the place of its vehicle on the highway and it coordinates lane change actions with neighboring vehicles (inter-platoons coordination). Following the synchronization of neighbors' lane change requests, the vehicles executing their lane change have to coordinate the actions involved in the lane change. For example, if a vehicle leaves a platoon to enter a new one during its lane change, it has to coordinate both a split and a merge manoeuvre. This coordination task is handled by the *Networking* module, which is responsible of the intra-platoon coordination and thus, the platoon formation's stability. The *Networking* module coordinates driving actions with other driving agents involved

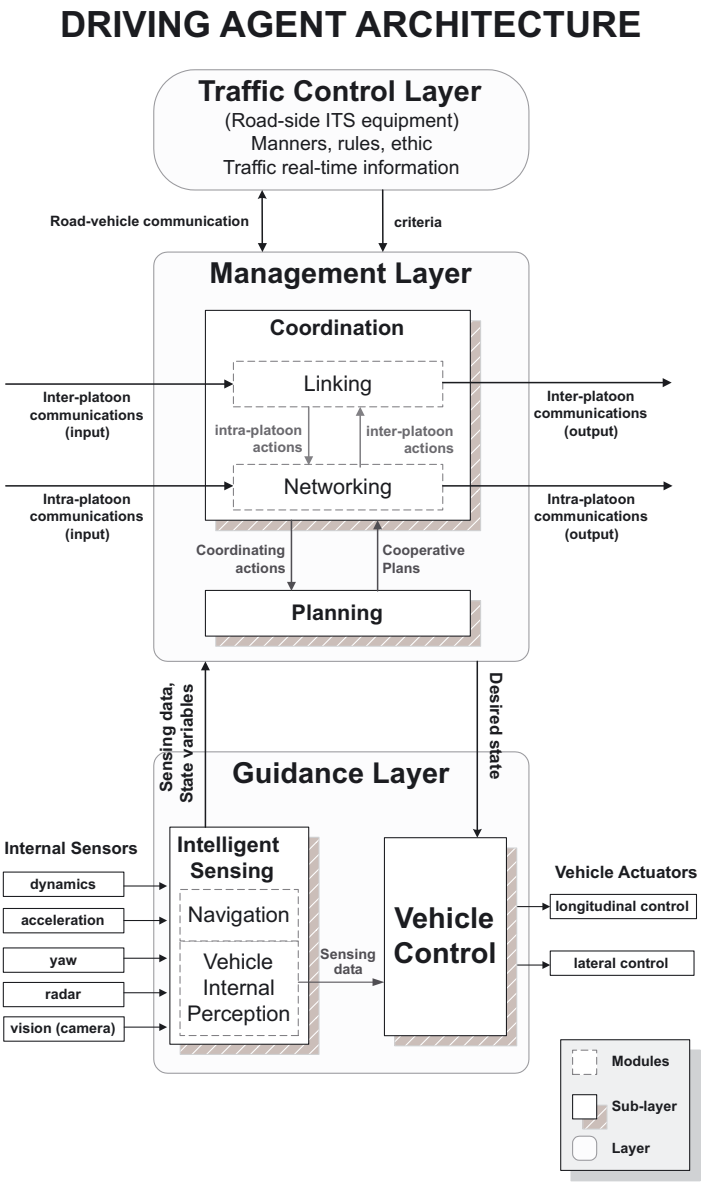


FIGURE 2. Hierarchical driving architecture.

in the manoeuvre (split or merge) to finally plan a series of local actions using the

Planning sub-layer. This sub-layer schedules driving actions (in time or according to events) that are locally executed by sending "desired state" queries to the *Guidance* layer.

The *Traffic Control* layer is a road-side system composed of infrastructure equipments like sign boards, traffic signals and the road-vehicle communications as well as a logical part including: social laws, social rules, weather-manners and other ethics (more specific to Canada), etc.

4. Communication and Coordination Methodologies

Communication has shown its value in Collaborative Driving Systems (CDS), by providing faster response time, more efficiency and by enhancing safety [18], but we must define the most efficient way of using it, in order to take full advantage of this technology. The different possible communication methodologies for the platoon of vehicles are implemented in the *coordination* sub-layer of Figure 2. For the coordination of the platoon and its two main manoeuvres: split, merge, we describe four models and outline their differences in Section 5. The models we decided to compare were taken in part from projects as PATH [17], which have mostly used platoon architectures coordinated by a master entity (the leader), although some decentralized coordination models were also proposed [4]. However, in our application, we bring this decentralization even forward, to finally come with a novel approach to inter-vehicle coordination in CDS: a teamwork coordination for driving agents. In this section, the centralized, decentralized and teamwork models are described by focusing on the teamwork model, which is more complex and represents the most promising research avenue for our application. Note that in each of our coordination models, the guidance and control systems are decentralized for every vehicle involved [10].

Figure 3 presents the four coordination models we compared inside a common CDS framework. This figure illustrates the differences in the communicative behavior of each model by showing which vehicles are involved in the split and merge manoeuvres, and wether each vehicle "surely" communicates or not. Accordingly, Figure 3 outlines the fact that every communication in the centralized model involves the leader, while the communications in the decentralized model only involve two vehicles. Moreover, the communications in the teamwork model involve most of the platoon members, but they do not necessarily communicate during the manoeuvre, as it is shown in Section 4.3.

4.1. Centralized Platoon Coordination

In a centralized platoon coordination model, the communications are centered on one "master vehicle" giving orders to the rest of the platoon: the leader. In this case the leader is the head vehicle of the platoon, and as mentioned earlier, this vehicle is driven by a human (simulated) in our first phase of development. To maintain the platoon formation, the leader is the only entity that can give orders, in which case the followers only apply requested changes.

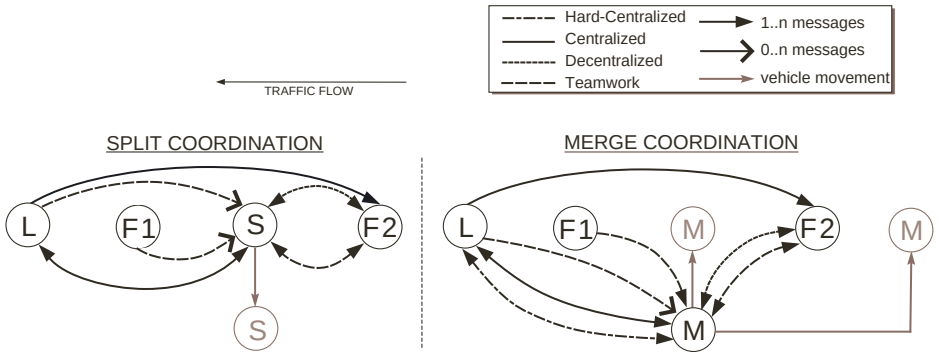


FIGURE 3. Four coordination models of the merge and split task.

During a split manoeuvre, three vehicles are involved: the leader, the splitting vehicle, the vehicle following the splitting vehicle (vehicle *F3* in Figure 3). During a merge, the same configuration of vehicles is involved: the leader, the merger, the vehicle which will follow the merged vehicle after its lane change (vehicle *F2* in Figure 3). For both of these manoeuvres, the merger or splitter first communicates its need to do a manoeuvre, and then, the leader communicates requests for inter-vehicle distance, change of lane, meeting point or velocity to involved vehicles.

For the merge manoeuvre, we have defined two sub-models: hard-centralized and centralized. The hard-centralized model simplifies the task and only requires two vehicles to communicate, by requesting the merging vehicle to always merge at the end of the platoon. In the centralized model, the leader specifies the optimal in-platoon merging position, considering the merging vehicle's position (parallel to the platoon). Thus, the centralized model requires three vehicles to execute a merge (leader, merger and gap creator), while the hard-centralized model requires only two (leader and merger).

4.2. Decentralized Platoon Coordination

In a decentralized platoon coordination model, the leader is still the platoon representative, but this is only for inter-platoon coordination. Thus, every platoon member has a knowledge of the platoon formation and is able to react autonomously, communicating directly with each others. An agent's common knowledge is initialized when it enters the platoon and is updated using the broadcasted information about new vehicles' merge or split (done at the end of such manoeuvres).

This model represents the simplest decentralization approach and does not rely on any existing framework or complex distributed plans, but tries to lower the communications as much as possible. In this model, the leader is only in charge of maintaining the platoon safety by notifying others of any emergencies, similarly to the centralized approach. For the split manoeuvre, only two vehicles are involved: the splitter and the vehicle following the splitter (vehicle *F2* in Figure 3). For

the merge, once the merging vehicle has chosen a platoon, only two vehicles are involved as well: the merger, the vehicle which will follow the merged vehicle after its lane change (vehicle *F2* in Figure 3). For these two manoeuvres, we eliminate the intermediate, i.e., the leader in the centralized model, because every platoon member has the knowledge of its platoon configuration. To replace the leader and coordinate platoon members without relying on a single vehicle, we used a set of social laws, which specify the state in which an agent can take part in a manoeuvre. This way, the actions of creating a safe gap for the split and merge tasks can be handled by a platoon member without being assigned by the leader. For instance, the intention of creating a safe gap emerges (through social laws) from the vehicle at the right distance from the merging vehicle, while the other platoon members determines that it is not their task.

4.3. Teamwork for Platoons

The previous decentralized model can be improved using a better structured organization, as the teamwork for agents, a concept gaining in popularity these recent years, in the filed of multiagent systems. In this context, a teamwork architecture like STEAM [15] can be used to assign roles to platoon members within a predefined team hierarchy. Using this type of architecture, most of the communications required to coordinate team members are handled by the framework. Thus, the teamwork concept results in most vehicles of a platoon to be involved in tasks and communicate when it is necessary, as shown by the dotted lines of Figure 3, representing "possible" communication. For the Auto21 project, we adapted STEAM's communication framework (based on STEAM operators) considering our specific needs. We then defined the required CDS team structures and developed a set of driving plans as domain-level operators, which can be used inside STEAM's framework. This section presents the previous aspects starting with the Auto21 team formations and the Auto21 domain-level operators, followed by the description of STEAM's operators, which are used to ensure coherence inside our team formation.

4.3.1. Auto21 Team Formations & Operators. For the Auto21 project, we defined three major teams: (i) the "platoon formation" team; (ii) the "split task" team; and (iii) the "merge task" team. The "platoon formation" team is a persistent team using persistent roles for long-term assignments. The two latter types of team are task teams using task-specific roles, for shorter-term assignments, since these teams stop existing after the task completion. Each of these teams is formed of different agents that must fill all the roles that are specified in the team's definition. For instance, the "split task" team is defined by Figure 4's tree, in which the leaf nodes represent roles and the internal node (only one in this case) represents a sub-team (the task observers). When an agent fills a role inside a team, this role defines the actions this agent can execute, while the notion of team defines the agent's goal or intention.

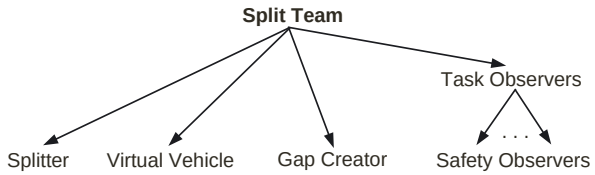


FIGURE 4. Split task team's role organization.

Following the assignation of a specific role to an agent, the STEAM framework constrains this agent to execute domain-level operators that are specific to the assigned role. Domain-level operators refer to the agent's actions (programmed as plans in our application) and they can be defined in a hierarchial tree. Such a tree is presented in Figure 5, which depicts the operators used by the merge team formation of our CDS. In the tree's hierarchy, team operators are surrounded by `[]`, while the other operators are standard individual operators. Agents evolving in the STEAM framework are able to execute two possible types of operator: domain-level; and architecture-level (STEAM operators). The operators presented in Figure 5 are from the domain-level, since they represent actions of the CDS domain, while the architecture-level operators, described in Section 4.3.2, are independent of the application's execution environment.

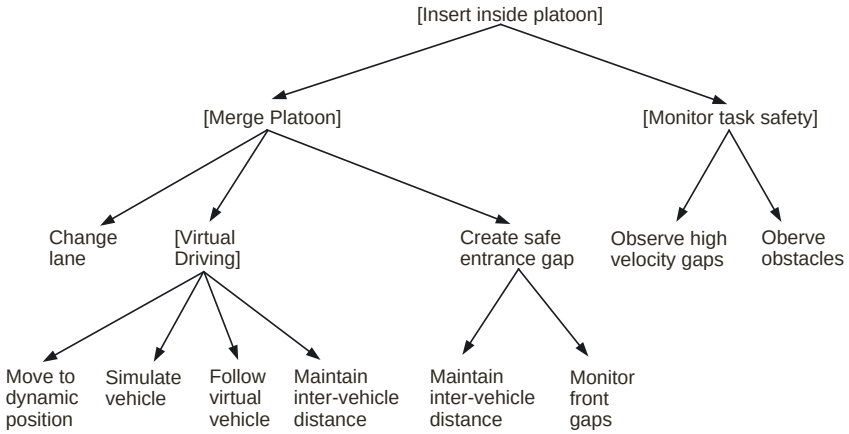


FIGURE 5. Team and individual operators hierarchy for the merge manoeuvre.

As mentioned above, we developed three types of team for the Auto21 project. The "platoon formation" team is the simplest team, where each driving agent holds the intention of maintaining a stable and safe platoon formation. At the moment, we consider that each member of the "platoon formation" has the same task, which

consists of following the front vehicle in a safe manner. Hence, this formation only requires two persistent (long-term assignments) roles:

- *Leader:* a role filled by the head vehicle, which mainly communicates with others using Selective Communication (SC) operators (defined in Section 4.3.2). Since its goal is to maintain a stable platoon, the leader can perceive an unsafe deceleration as a situation that could endanger the goal achievement, therefore influencing the leader to inform others of this fact using SC operators. The probability of this communicative act is discussed later.
- *Follower:* a role filled by all the platoon members that are not at the head. At the moment, each follower's goal consists of maintaining the safe inter-vehicle distance with the preceding vehicle, in order to maintain the platoon's stability. An agent in this role does not need to communicate any information since the automated driving system of each vehicle is capable of maintaining the platoon stable in the context of a "platoon formation" team. Thus, the task of maintaining a safe distance is realized by using the vehicle's front sensor and possible information from the leader.

The "merge task" team being similar to the "split task" team, we only depict the "merge task" team in this paper. This team is centered around the [Insert vehicle] team operator, shown in Figure 5. As shown in Figure 4, the split team (similar to the merge team) is formed of four different roles and a sub-team: the *Task-Observers* sub-team. Each role is described below by referring to the operators they use in Figure 5's tree and the place they occupy in the platoon, in Figure 1's illustration of the merge manoeuvre.

- *Merger:* a role filled by the agent which initiates the "merge task" team by broadcasting its intention to merge a platoon (vehicle *L2* in Figure 1). The merger executes the [Merge Platoon] team operator (refer to Figure 5), which restricts its local operators to the ones below [Merge Platoon], in the same tree. The Move to dynamic position operator is used by the merger when the "merge task" team has acquired a mutual belief about the merger's entry position in the platoon. The *Merger* role also uses the Follow virtual vehicle operator, which is a virtual representation of *L2*'s future preceding vehicle (*F1*). This virtual vehicle is followed by *L2* before it actually senses the real vehicle with its laser. Finally, the Change Lane operator is used here, to switch to the platoon's lane and complete the merge.
- *Gap Creator:* a role filled by the agent driving the vehicle behind the merging position, in the platoon (vehicle *F2* in Figure 1). Within this role, an agent defines the entry position for the merger, since its vehicle will be behind the merger after the lane change. The *Gap Creator* role requires its filler to execute the Maintain inter-vehicle distance operator (refer to Figure 5), which maintains a distance large enough to safely fit a vehicle. Following the execution of this operator, the *Gap Creator* has to execute the Monitor front gaps operator when the merger is changing lane. This operator monitors high gap values between the last front vehicle percept reading, to conclude

on the arrival of the merging vehicle in the platoon. At the end of the merge manoeuvre, the *Gap Creator* executes the Maintain inter-vehicle distance with a smaller distance value, in order to close the gap in the platoon.

- Virtual Vehicle: a role that was introduced to ensure a stable execution of the manoeuvre. This role helps the manoeuvre executor (splitter or merger), when it is in a different lane, to follow the vehicle that was or will be in front of it. In the split and merge manoeuvres, this role is taken by vehicle number *F1* from Figure 1. Within the [Split Platoon] team operator (not illustrated here), the *Virtual Vehicle* role applies the Simulate Vehicle operator if its own velocity is modified after the splitting vehicle has changed lane and before the split manoeuvre is over. This operator results in the communication of information about the *Virtual Vehicle*'s new velocity. Within the [Merge Platoon] team operator, the Simulate Vehicle operator is applied after vehicle *F1* has transmitted an initial representation of itself to the merging vehicle. This role thus ensures a safe entrance of the merger, which always has an up to date representation of the vehicle it cannot sense when changing lane.
- *Safety Observers:* a role filled by one or more agents. The constraint on the role fillers, is that they must be in a position ahead from the manoeuvre executor, so they can monitor dangers in advance. Using the communication selectivity presented in Section 4.3.2, agents in the *Safety Observers* role communicate their belief about dangers or unsafe deceleration to others, by taking in account the dangers of sudden movements during the execution of a manoeuvre. Agents filling this role conjointly execute the [Monitor task safety] safety team operator, therefore executing observation plans individually.

4.3.2. Team Framework Operators. To ensure a coherent execution of the domain-level operators, STEAM's framework incorporates architecture-level operators-(STEAM operators). These operators include: (i) Coherence Preserving (CP); (ii) Monitor and Repair (MR); and (iii) Selective Communication (SC). The CP and MR operators are not presented in this paper, since their use in our application is very limited, so the reader should refer to [15] for more information. In contrast, the CP operator has been very useful in order to support the "intra-team" communications of our CDS, so it is detailed below.

The Selective Communication (SC) operator's task is to synchronize mutual beliefs within the execution of team operators (domain-level operators). A SC operator simply monitors the agent's local beliefs and compares them with the team's mutual beliefs. If it considers that the difference between these two beliefs is important enough, it automatically communicates the agent local belief to other team members and makes its local belief a mutual belief. In order to determine if a new belief should be communicated, the SC operator uses the decision tree represented in Figure 6. According to this tree, the SC operator communicates the new belief considering a reward based on the following variables:

- ρ: the probability that the new belief is not known by its teammates.

- σ: the probability that the new belief opposes a threat to the execution of the current team operator.
- C_c: the cost of communication.
- C_n: the cost of nuisance.
- C_{mt}: the cost for miscoordination.
- S: a reward for synchronization of the team's belief during the execution of a team operator.

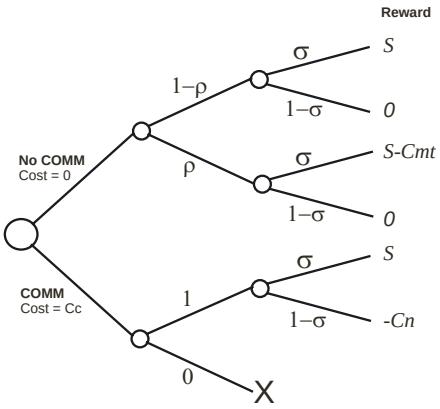


FIGURE 6. Decision tree on team communicative acts, from [15].

By using the decision tree in Figure 6, the SC operator chooses to communicate iff the reward of making a new belief "mutual" is higher than the cost of communications. With a probability of $1-\sigma$, the new belief is not a threat to the team operator, and therefore, there is no reward relating to this belief. If the agent chooses not to communicate (No COMM (NC)) and the belief is a threat, with a probability $1-\rho$, this belief was already known by its teammates so the agent receives the standard reward of S. However, with a probability ρ, this belief was not mutual, so the agent receives the standard reward of $S - C_{mt}$, where, in our application, C_{mt} depends on the difference between the local and mutual belief (if the local belief is much different from the mutual belief, C_{mt} is high). Therefore, from Figure 6, the expected utility of not communicating can be defined as $EU(NC) = \sigma * S - (\rho * \sigma * C_{mt})$.

In the situation where the agent decides to communicate (COMM (C)), a cost of communication C_c is applied to further possible rewards and this cost is fixed considering the current available communication bandwidth. Following a decision to communicate, the new belief is definitely mutual, so the second branch is irrelevant in Figure 6 (probability 0). However, the new belief can be a threat with a probability σ, in which case the agent receives a reward of S. Nevertheless, with a probability $1-\sigma$, this was not a threat and the communication of this belief

has a cost of nuisance C_n to other team members. This cost depends on the type of information that is communicated, but it is usually set to discourage agents from communicating information that may disturb other team members, when it is not necessary. According to this definition, the expected utility of communicating a new belief to team members can be summarized as $EU(C) = \sigma * S - (C_c + (1-\sigma) * C_n)$. The two previous equations on expected utility can be merged to represent the decision of the Selective Communication (SC) operator, which communicates when $EU(C) > EU(NC)$, i.e., iff:

$$\rho * \sigma * C_{mt} > (C_c + (1 - \sigma) * C_n)$$

This kind of decision-theoretic selector being part of the STEAM framework, we use it for all of the roles presented earlier. Probabilities, cost and rewards used by the communication decision tree are initialized according to common knowledge on Collaborative Driving System (CDS) and should be adapted through testing using offline learning approach on patterns of communication within the team. For the moment, the SC operators have been very useful to determine when a *Safety Observers* or a *Virtual Vehicle* should communicate its new beliefs. For instance, if the *Virtual Vehicle* (vehicle *F1* in Figure 1) has to modify its velocity during the merge, the probability ρ that this new information on *F1*'s velocity is commonly known mainly depends on the probability P(*L2*,*F1*) that the merging vehicle *L2* has *F1* in its sensor's range (if *L2* is in the platoon). Furthermore, the probability σ that this information opposes a threat to the merge manoeuvre depends on the difference between *F1*'s local belief on its velocity and the team's mutual belief. If the team is highly out of synchronization, the agent will communicate at a higher probability. Finally, cost C_n will be low for this type of belief, while cost C_{mt} will be higher, since changes on the virtual vehicle's velocity affect the platoon safety.

5. Evaluation and Results

To develop the previous theory on coordination strategies, we have used an agent development toolkit called JACK Intelligent Agents™ [2] which supports the Belief Desire Intention (BDI) agent model [13], as well as teamwork related strategies. The BDI model allows us to develop our application in such a way that each vehicle's driving agent has a database of beliefs and a set of predefined desires that rule its intentions, i.e. change lane, which result in the application of actions like merging and splitting from a platoon. In the Team Oriented Programming (TOP) vision relating to the STEAM model, a non-negligible advantage is the reusability and flexibility of the operators [15], since it contains many infrastructure rules that are not directly related to the domain level. Thus, using JACK's representation of an agent, we managed to develop collaborative driving teams that follow TOP models.

The coordination models presented in Section 4 have been implemented according to the architecture presented in Figure 2. We show as an example in Figure 7's graphic, the results we got in the average coordination of a vehicle exit (split

manoeuvre), using a centralized coordination (in the two blue lines) as opposed to the teamwork model of coordination (in the red blue lines). This graphic shows results using the splitter's (vehicle *F2* in Figure 1) data and the splitter's preceding vehicle's (vehicle *F3*) data. The preceding vehicle senses the splitting vehicle and has to adjust from its departure by keeping a *safe* gap until the splitter is safely out. The two bold lines present the difference between, the inter-vehicle distance (IVD) between this vehicle and its front vehicle (splitter), and the *safe* front distance. This *safe* distance is defined by a gap in time between vehicles that agents should respect to insure security. In addition, the two thinner dotted lines only show the inter-vehicle distance (IVD) from *F3*'s sensor, without applying a difference.

Around time $14s$ in Figure 7, vehicle *F3* has to create a larger and safer distance with the splitting vehicle, so the two bold lines drop, but are readjusted within almost $10s$. The second outlined step arises at time $30s$, and $27s$ for the teamwork, when the splitter has went out of range of vehicle *F3*'s laser. At this moment, the sensed distance raises on the IVD curves, but there is no gap considering the distance defined as *safe* (bold lines). Before the splitter has stabilized itself on the next lane (time $37s$ or $34s$), the gap creating vehicle of the centralized model does not manage to keep the *safe* distance and has a difference of $2m$ with the *safe* distance by the end. On the other hand, the splitting vehicle modelled with the teamwork coordination is using communications from vehicle *F1* through teamwork rules, to maintain his virtual representation and follow it after it has changed lane. Thus, the splitting vehicle maintains the right *safe* distance more easily in the teamwork model. This approach gives much better results, since the difference with the *safe* distance does not go higher than $0.5m$. When the splitter is stable in the other lane, the distance qualified as *safe* drops to the normal intra-platoon distance. At that moment, the vehicle is at $17m$ (length of the gap created by the vehicle that left) from the *safe* distance, reached by the end of the manoeuvre. This graphics also shows that using a teamwork model, information is exchanged faster since messages do not have to go through the leader, which results in an overall faster response time of three seconds by the end of the manoeuvre.

The implementation of four different coordination strategies leads us to conclusions concerning their respective advantages and disadvantages. First, as mentioned in Section 4.1, a fully centralized coordination at its simplest version was developed. Second, we developed another model which was centralized on the leader, but allowing entrance anywhere inside the platoon. In the third approach, the coordination is decentralized, so the vehicle executing the manoeuvre only has to coordinate itself with the vehicle directly concerned by this manoeuvre. The fourth approach uses the previously detailed teamwork model.

Strategies on decentralized approaches are still under development and extensive simulations including multiple platoons to increase traffic density will have to be done to improve our results and the choice being done in the end. But using the preliminary results, presented in Table 1, we show each of the four models used for both a split and a merge, divided in four rows. We compared on the first

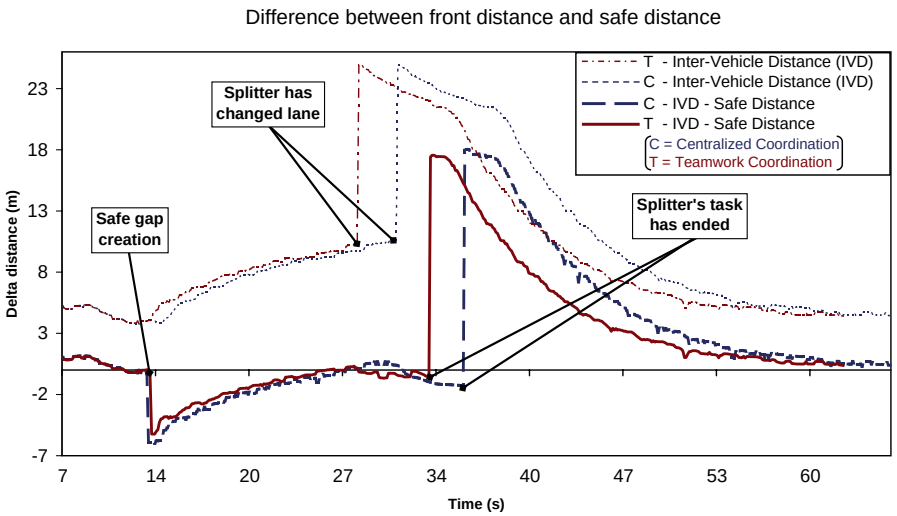


FIGURE 7. Inter-vehicle distance with the splitting vehicle.

pair of column, the average total amount of messages exchanged by vehicles during each manoeuvre. On the second pair of columns, we compare the amount of plans (JACK plans), which were required to support each coordination model, showing the flexibility of their respective framework. These JACK plans refer to the coordination plans developed in JACK Intelligent Agents™ language[1] that made it possible to support a specific manoeuvre, with a specific communication protocol.

TABLE 1. Total of messages and plans used by coordination model

Coordination	Nb Messages		Jack Plans	
	Merge	Split	Merge	Split
Hard-Centralized	7	7.5	12	12
Centralized	11.5	8	20	13
Decentralized	8	5.5	12	9
Teamwork	8.75	6.75	14	10

By using the results presented in Table 1, along with the results presented in Figure 7, each coordination model were analyzed and their respective advantage and disadvantage are summarized as follows:

1. The first centralized coordination model (*Hard-Centralized*) is very benefic on the amount of messages it exchanges. But the major disadvantage is the traffic

[1]In JACK, Plans are pre-compiled procedures based on the PRS architecture [6]. A plan can only be executed when it answers to internal or external events using *relevance* and belief *context* criterions.

density it creates, as it must reach the platoon's tail by either accelerating or decelerating (considering his position), thus creating traffic waves and diminishing the highway's capacity.

2. The second centralized coordination model suffers from the amount of messages it encounters, as the leader redirects all the messages within the platoon. Moreover, in average, more than three quarters of the messages were sent or received by the leader, creating a bottleneck for this vehicle. The centralized model does not require the followers to keep a platoon knowledge, which helps lowering communications compared to decentralized models. As the previous model, the centralized coordination uses static coordination protocols supported by the leader, which has the disadvantage of not allowing much flexibility on the coordination of unexpected situations.
3. The decentralized coordination model uses less messages, but is a lot less safer than the other models. This fact will be proven using further simulations, but since it uses only two actors, and no virtual vehicle (as mentioned for the teamwork model in 4.3), this model would have to compensate by using more sensors to attain the level of safety of the teamwork model. This model also needs to communicate to initialize and maintain common knowledge within the platoon, but since the "updates" on knowledge are done through a broadcast, it does not require much more messages.
4. In the teamwork model, we managed to use an amount of messages that is in the average of the three other models. It must be mentioned that this number varies more than in the other models, from different contextual simulations, because of the selective communications. On the other hand, the selective communications enable a faster and safer execution of our manoeuvres, as shown in Figure 7. Using a TOP model implemented with the STEAM vision in JACK, even though there are more actors (to increase safety), we do not need much more plans than the standard decentralized approach, which was not developed using generic plans. Moreover, TOP uses less plans than the decentralized model, developed in a more functional vision, in which more plans are required to handle uncertainty, compared to STEAM's vision. Compared to the decentralized model, the TOP framework is in charge of the platoon's belief (common knowledge) and manages to handle better their communications.

6. Conclusion and Future Work

Collaborative driving is emerging in the domain of ITS and it will ultimately be part of the every day vehicle's automation system, since it is the next step, following the Adaptive Cruise Control (ACC). CDS can therefore be easily included in the development plans of ITS for the upcoming years, as it will evolve until vehicle level technologies like ACC meet AHS infrastructures technologies and increase ITS benefits on safety, efficiency and environment.

This paper presented an autonomous driving system based on a strong architecture giving a wide latitude to the definition of different inter-vehicle communication models. From these models, we have presented and tested four different strategies, for which advantages and disadvantages were presented. The coordination model based on teamwork presented great avenues considering its flexibility and its ability to safely and efficiently execute manoeuvres.

We will continue improvements on the longitudinal guidance system that could enable us to lower the communication probabilities in the selective communication decisions of the Team Oriented Programming (TOP) infrastructure. The different coordination strategies will be further extended and many more scenarios involving uncertainty will be taken into account using our simulator, similarly to the approach we have taken in [9].

References

[1] M. Antoniotti, A. Deshpande, and A. Girault. Microsimulation analysis of multiple merge junctions under autonomous ahs operation. In *Proocedings of the IEEE Conference on Intelligent Transportation System (ITSC) 97*, pages 147–152, November 1997.

[2] AOS. JACK Intelligent Agents™ 4.1. Software Agents Development Framework, 2004.

[3] Auto21. [Online], April 2004. `http://www.auto21.ca/`, (accessed the 30th of April 2004).

[4] S. Vahdati Bana. Coordinating automated vehicles via communication. Ucb-its-prr-2001-20, University of California, Berkeley, September 2001.

[5] DAMAS-Auto21. [Online], April 2004. `http://www.damas.ift.ulaval.ca/projets/auto21/`, (accessed the 30th of April 2004).

[6] M.P. Georgeff and F.F. Ingrand. Real-time reasoning: The monitoring and control of spacecraft systems. In Computer Society Press, editor, *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications(CAIA 90)*, volume 1, pages 198–205, Los Alamitos, Calif., May 1990.

[7] S. Ghosh and T. Lee. *Intelligent Transportation Systems: New Principles and Architectures*. CRC Press, 2000.

[8] Simon Hallé. Automated highway systems: Platoons of vehicles viewed as a multiagent system. Master's thesis, Université Laval, Canada, January 2005.

[9] Simon Hallé and Brahim Chaib-draa. Collaborative driving system based on multiagent modelling and simulations. *Transportation Research Part C: Emerging Technologies*, 2005. Submitted.

[10] X. Huppe, J. de Lafontaine, M. Beauregard, and F. Michaud. Guidance and control of a platoon of vehicles adapted to changing environment conditions. In *Proceedings of IEEE International Conference on Systems Man and Cybernetics, 2003.*, volume 4, pages 3091–3096, 2003.

[11] Petros Ioannou and Margareta Stefanovic. Evaluation of the acc vehicles in mixed traffic: Lane change effects and sensitivity analysis. PATH Research Report UCB-ITS-PRR-2003-03, University of Southern California, 2003.

[12] John Lygeros, Datta N. Godbole, and Shankar S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43:522–539, 1998.

[13] A. S. Rao and M. P. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, pages 312–319, San Francisco, 1995. The MIT Press.

[14] R. R. Stough. *Intelligent Transportation Systems: Cases and Policies*. Edward Elgar Pub. LTD, 2001.

[15] M. Tambe and W. Zhang. Towards flexible teamwork in persistent teams: extended report. *Journal of Autonomous Agents and Multi-agent Systems, special issue on Best of ICMAS 98*, 3:159–183, 2000.

[16] S. Tsugawa, S. Kato, K. Tokuda, T. Matsui, and H. Fujii. A cooperative driving system with automated vehicles and inter-vehicle communications in demo 2000. In *Proceedings of the 2001 IEEE Intelligent Transportation Systems Conference*, pages 918–923, August 2001.

[17] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 32, 1993.

[18] Qing Xu, Karl Hedrick, Raja Sengupta, and Joel VanderWerf. Effects of vehicle-vehicle / roadside-vehicle communication on adaptive cruise controlled highway systems. In *Proceedings of IEEE Vehicular Technology Conference (VTC)*, pages 1249–1253, Vancouver, Canada, September 2002.

Simon Hallé
Département d'informatique et génie logiciel
Université Laval
Sainte-Foy, QC, Canada, G1K 7P4
e-mail: `halle@iad.ift.ulaval.ca`

Brahim Chaib-draa
Département d'informatique et génie logiciel
Université Laval
Sainte-Foy, QC, Canada, G1K 7P4
e-mail: `chaib@iad.ift.ulaval.ca`

Computation of Location Choice of Secondary Activities in Transportation Todels with Cooperative Agents

Fabrice Marchal and Kai Nagel

Abstract. Activity-based models in Transportation Science focus on the description of human trips and activities. We address the modeling of the spatial decision for so-called secondary activities: given both home and work locations, where do individuals perform activities such as shopping and leisure? The simulation of these decisions using random utility models requires a full enumeration of the possible outcomes. For large data sets, it becomes computationally unfeasible because of the combinatorial complexity. To overcome this limitation, we propose a model where agents have limited, accurate information about a small subset of the overall spatial environment. Agents are inter-connected by a social network through which they can exchange information. This approach has several advantages compared to the explicit simulation of a standard random utility model: a) it computes plausible choice sets in reasonable computing times b) it can be easily extended to integrate further empirical evidence about travel behavior and c) it provides a useful framework to study the propagation of any newly available information. The paper emphasizes the computational efficiency of the approach for real-world examples.

1. General context

Activity-based models in Transportation Science focus on the description of the organization of human activities in time and space. This organization determines the demand for travel, that is the amount of users that the various transportation systems need to accommodate. It is assumed that the demand for travel is derived from the demand for performing activities at specific locations. Obviously, individuals constantly perform some trade-off between enjoying activities that have a high reward value (for instance working at a company in the downtown area) and the time and budget it takes to get to the specific location of these activities.

Various operational models such as URBANSIM [1] are available to describe this trade-off for the choice of the home and work locations. It is essentially assumed that users perform a trade-off between rents, travel costs and wages. However, empirical evidence [2] have shown that a significant amount of traffic is generated for other purposes than commuting, often referred to as *secondary* activities: shopping, leisure, going to social events, etc.

2. Problem statement

Our work intends to model the specific process of the location choice of secondary activities in the case of high resolution data sets. The methodological constraints are that the modeling should be behaviorally sound, compatible with micro-economics foundations and computationally feasible. The temporal dimension (i.e. the scheduling of the activities) is ignored for the time being. We assume that the order of the activities called a *plan* is given (i.e. getting out from home, going to work, working for eight hours, going for shopping at lunch time, etc.). The physical environment is described by two large data sets that typically originate from Geographical Information Systems (GIS): a) the land-use data and b) the transportation system data. The land usage is a raster-type description that includes the information about the nature of each parcel of the studied area (e.g. housing density, number of shops, type of area: rural, commercial, industrial). The transportation system is a vector-type description of the various transportation modes available (e.g. car, rail, bus) as a network with nodes and links. Nowadays, these data are available at a very high resolution: typical land-use cells are 100 meter square and road networks are described down to 10 meter road sections. The long-term goal of our research is to model entire metropolitan areas microscopically by simulating the individual decision of millions of citizens. Therefore, the problem at hand can be stated as follows: how to simulate the selection of the activity locations of $A = 10^6$ **citizens** in a **grid** that has $C = 10^5$ **cells**. Note that the travel times from cell to cell have to be given by external traffic models. Multi-agents traffic assignment models such as those developed by the authors [3, 4] are now able to predict travel time patterns for large-scale data sets. Therefore, this issue will not be considered here. The generation of travel demand for these models is addressed here.

3. Micro-Economics foundation

The standard practice in Transportation Science to approach such problems is to use random utility models (RUM) borrowed from the discrete choice theory of Micro-Economics [5, 6]. These models assume that individuals are maximizing their own utility. For instance, the utility to go shopping at a mall located in cell i for a simple plan (i.e. home - shopping - work) is given by

$$U_i = R_i - C_{hi} - C_{iw} + \mu\epsilon_i = V_i + \mu\epsilon_i,$$

where R_i is the reward associated to shopping at that particular facility that depends on the availability of goods, their prices, etc.; C_{ih} is the travel cost to travel from home to cell i; C_{iw} is the travel cost to travel from cell i to *work*; μ is a scale factor and ϵ_i is a random variable that is specific to the individual. The latter random utility part captures all the hidden preferences of a specific user for location i that are not accessible to the modeler. By contrast, V_i is called the deterministic part of the utility. Under the assumption that ϵ_i are i.i.d. extreme value distribution of type I, it can be shown that the probability to choose to go shopping at cell k is given by:

$$P(k) = P(U_i < U_k \forall i \neq k) = \frac{\exp(-V_k/\mu)}{\sum_{i=1}^{C} \exp(-V_i/\mu)}$$

Since a probability greater than zero is assigned to each potential intermediary stop on a cell, this formulation requires a full enumeration of the possibilities on the spatial grid. For trips that count S intermediary stops, the complexity is $O(AC^S)$ which is not feasible in realistic cases. Initially, RUMs were intended for the description of choices between a small set of alternatives distinguishable by humans (e.g. car brands). Their application to a discretized continuum (i.e. urban space) remains behaviorally questionable. Still, we believe it is fundamental to keep some compatibility with RUMs because of the huge amount of literature that has been devoted to developing empirical techniques (e.g. surveys) to calibrate the parameters of those models.

Another drawback of RUMs is that they provide only a static representation that does not take into account the temporal dimension of the decision process. RUMs do not model explicitly the choice process but only its outcome. Our goal is to come up with a model that reflects the underlying learning process and that will eventually include the dynamics of the building of the choice set. Capturing this aspect is potentially important to study, for instance, the evolution of travel demand given some modification in the land-use patterns. Moreover, the environment itself should be time-dependent. Travel times and travel impedances to move in the network are subject to within-day and daily variability. The feedback of the travel conditions on travel choices and conversely is also often missing from static RUM-based analysis. To overcome these limitations, we propose a four phase multi-agent simulation model that includes a dynamic learning process and that can, in principle, be coupled with a dynamic mobility simulator. Agents travel, explore, learn and socialize.

4. Multi-agent based approach

We propose a multi-agent based simulation where **each agent (i.e. each simulated citizen) has only limited, accurate information about N cells** ($N \ll C$) called the "memory" of the agent. The intuition is that real humans have limited cognitive abilities and can only consider a small amount of options at the same time. The organization of these options in the mind and the human representation of space is

probably far different from "pixels". Nevertheless, we keep that trivial representation for now, as it can be replaced later by a more sophisticated one, for instance that of [7].

We assume that agents are inter-connected by a *social network* through which they can exchange information about their respective subsets. Each agent is socially connected to K **acquaintances or "kins"**. The intuitive benefit of the social network is that agents are very heterogenous yet facing similar choices. Therefore, a decision which is optimal for an agent might be close to optimal for one of his acquaintances. We show below that the diffusion of the knowledge of optimal strategies through the social network can, indeed, exploit that hidden redundancy.

The simulation is iterative and each round has four stages: evaluation, socialization, exchange and exploration.

Evaluation: each agent performs the location choice of the intermediary stops based on his own private information. The choice can be deterministic (the best cells are selected from the memory of the agent) or probabilistic (a RUM is applied to the small set corresponding to the memory of the agent). The computing load for building and storing the travel plans is $O(AN^S)$. At this point, the plans are fed in a dynamic traffic model (such as MATSIM [3] or METROPOLIS [4]). The traffic simulator computes the delays incurred due to traffic congestion, which are then used in the next round of evaluation.

Socialization: social connections are created and deleted dynamically. The deletion mechanism is a simple exponential decay. The creation mechanism is a spatial reinforcement reminiscent of pheromones in ant colonies optimization [8].

Exchange: for each of its social connections, an agent has the opportunity to exchange a piece of information. A cell is picked randomly from the agent's memory and the other agent is informed about it. The exchange is bi-directional and the outcome of the exchange is described by the learning mechanism below. More sophisticated exchange strategies could be taken into account.

Exploration: agents have the possibility to explore cells in the neighborhood of those that they visit. This stage is mainly intended to recover potential information loss in the other stages, thus relieving the implementation from checking that any cell was lost from the global knowledge of all the agents. Obviously $O(A)$ operations are required.

5. Learning mechanism

The memory of an agent is represented on Figure 1: the two first buffers contain the information about locations that are either close to home or close to work (e.g. a small circular area). The third buffer called "elite" buffer (of size E) corresponds to locations that have high score values and the last buffer contains "vague" information about cells that have poor score values. The scores corresponding to

the elite buffer are also kept in the agent's memory. When an agent informs another agent during the exchange stage, a cell is picked randomly from the total memory of the informer. At that point, the informed agent evaluates how this new cell information can potentially improve his/her plan score. This is done by comparing the poorest score of the elite buffer to the scores of all the potential plans with at least one stop at the new cell. This revision implies to replace each intermediary stop of the potential plans by the new cell: $O(SN^{(S-1)})$ operations are needed. If the score is better than the worst solution of the elite buffer, the new cell is promoted to that buffer and it is sorted: $O(\ln E)$ operations are needed. If the cell does not improve one of the elite plans, the cell information replaces a cell randomly selected from the vague buffer. This has two consequences: Firstly, agents keep information that is not relevant to themselves but that might be to others in the future, hence they adopt a *cooperative* behavior that is not supervised. Secondly, the information in the vague buffer can be erased and lost forever. The information about cells with very low utility is more likely to be lost from the collective memory. The exploration phase allows still to recover them. So far, the computation load of a single round is $O(A(\ln E + SN^{(S-1)}))$ which is feasible for reasonable assumptions ($S \leq 3, N < 50, E < 50$). However, the number of iterations is still to be determined. Note that the learning speed of the overall process depends on the *greediness* of the exchange which is a function of the ratio between the sizes of the elite and the vague buffers.

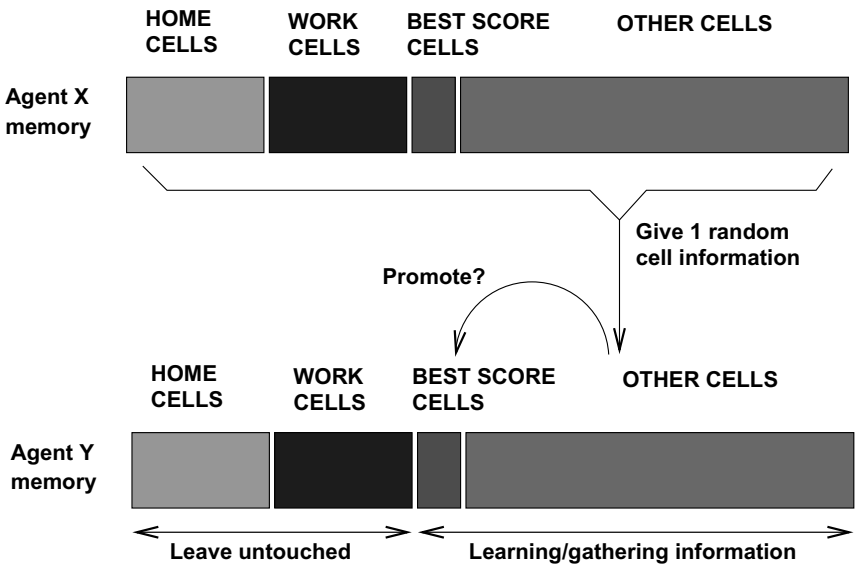


FIGURE 1. Agent memory organization and learning

6. Social network dynamics

Social links disappear following an exponential decay law at the end of each iteration. Initially, the social network is a random graph of degree K and each social connection has the same decay time. When two agents perform activities at the same location, two situations can occur: 1) if there is a social connection between them, the strength of the connection is reinforced and its decay time increases 2) if not, a new social connection is created. This task requires $O(AK)$ operations for the enumeration of the social links and $O(C)$ operations to browse the cells for detecting interactions between agents. However, the detection of the existence of a social link between two given agents would require $O(AK)$ operations for each cell to browse the connections. An alternative solution would be to store the connections of a given agent in a hash table. Both solutions are costly in memory or computation time. For these reasons, we adopt a slightly different implementation which is equivalent statistically. Every connection has the same decay time but multiple redundant connections can exist between two given agents. When an agent visits a particular cell from his own elite buffer, the cell/agent pair is stored until a second agent visits the cell. A new social connection is then created between the two visitors. After one iteration, we end up with a larger number of connections than at the beginning. All of them are of equal strength but with potential redundancy (a given pair of agents can appear several times). Identifying the redundancy would be costly. Instead, we keep the total connection strength constant in the system by deleting connections randomly so that the total number of connections is constant.

7. Implementation issues

The code of the simulation has been written in Java. Input and output files use the XML file format which is well suited to variable length content such as the description of the individual plans with multiple stops. The goal being to simulate 10^6 agents on a single CPU, some performance concerns have to be taken into account.

Cells: each grid cell is stored as an individual object. With 10^5 cells, it is not crucial to store the cell attributes as plain arrays. This allows to keep cell characteristics private and to have cell references. Since cells have to be often compared during the learning process to determine if a new cell is already known to an agent, it is far more efficient to use the identity operator == than the default Java *equals()* method which is the equivalence operator. This is valid as long as the cells are not dynamically allocated or cloned once the simulation starts. Cells have to keep references to agents that visit them (see the socialization stage). The average number of visits per cell is small ($O(SA/C)$) so that we could allow for the overhead of a dynamic container (e.g. vector). However, in the simple pairwise interaction described above,

only the information about the last visitor is needed. Therefore, it is sufficient to maintain a hash table of visited cells and visitors.

Agents: it is tempting to have a dynamic container steadily increasing in size for the agent memory. However, that would completely ruin the performances and is not compatible with our assumption that only a limited number of simultaneous options can be memorized. Each cell is referenced on average by $O(AN/C)$ agents so that the initial coverage is sufficient to ensure that there is not any information missing about the environment.

Random numbers: a typical bottleneck of this kind of simulation is the computation of random numbers. A priori, $O(AK)$ random numbers have to be computed for each single iteration of the information exchange stage. This quickly becomes prohibitive and can be avoided by using two integer random seeds at the beginning of the exchange stage. One is used to pick a cell from the informer agent, the other to replace a cell in the memory of the informed agent. These two pointers can be simply incremented from one social link to the other since there is no correlation between social links and they are accessed in a *a priori* random order.

Update sequence: The original ordering of the agents is that of the XML source and could be biased. Agents are processed sequentially during the evaluation phase: each agent uses one of his elite plans and marks the visited cells. Therefore, the probability to create a social connection between agents decreases with their distance in the list. To avoid this potential bias, the list of agents is randomized at the beginning. Social connections are also stored initially in a random list since they are processed sequentially during the learning phase. New social connections are created at the end of the list. The decay process rearranges them randomly.

8. Results

The simulation is tested on a real-world example for the Zurich region for which we have available a high resolution transportation network and a land-use raster (see Figure 5). The area covers approximately a 50x50 kilometer square area where about one million inhabitants are living. The land-use utility values R_i are generated based on census data. Random plans with 1 or 2 intermediary stops are generated for 10^6 agents that are distributed on the area according to job and housing densities. The home to work pairs are computed using an external model written by one of the authors (see [9] for the computation of the rent values presented in Figure 5). The initial social network that connects them is a random graph. Obviously, this is not realistic but we intend to evaluate only the computational feasibility in this preliminary work.

Figure 2 presents the evolution of the sum of the scores of all the agents during the iterative process. It can be seen that the process converges in a few dozens of iterations but that the choices are not optimal since the utility does not

reach the maximum value obtained with a full enumeration of the alternatives. This is due to the fact that some information is lost in the process. It can be recovered slowly through the exploration. However, some information can still be lost forever because the exploration mechanism is local: the agents cannot jump to explore a totally new area. This limitation could be easily removed. Still, the value of the plateau is high enough (more than 90% of the maximum utility) to ensure that plausible strategies have been selected. It remains to be studied how much this is compatible with empirical evidence. Note that the size of the memory of the agent only slightly affects the convergence properties.

Figure 3 illustrates a typical spatial adaptation process for a single agent with a two-stop plan: home-work-leisure-shopping-home. On the first iteration, leisure ($L1$) and shopping ($S1$) are performed at the home place because the agent ignores good locations to perform these activities. In the second iteration, he learns that the location $L2 - S2$ is a good location for one of the two activities, hence making the extra trip distance worth it. On the third iteration, the agent discovers that the area around $S3$ has a high utility for shopping. During the fourth and fifth iterations, the agent keeps shopping close to that area and only optimizes the leisure location ($L3 \rightarrow L4 \rightarrow L5$).

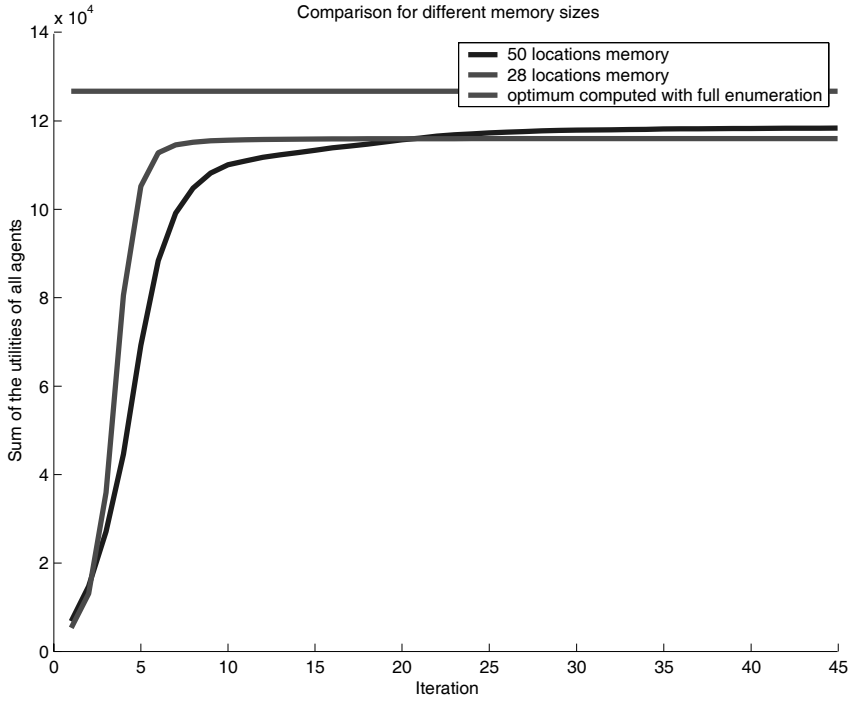


FIGURE 2. Performance of the multi-agent simulation

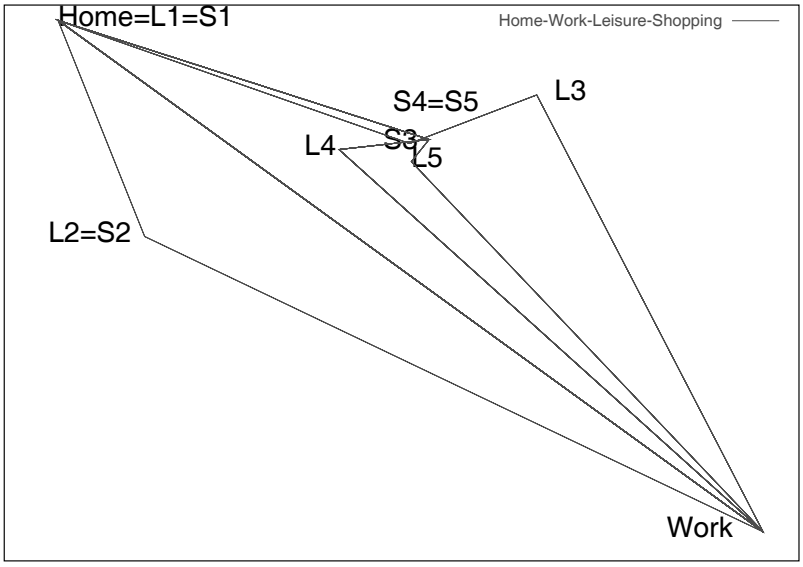


FIGURE 3. Adaptation of the location for a two-stop plan

Figure 4 shows the evolution of the distribution of social connections in the system. The initial condition is a random graph of degree $K = 20$. On the first iteration, the distribution is roughly a $N(20, 1)$ distribution. Progressively, the distributions shift to the left. In the end, we have a self-sustained distribution that can be approximated by a $N(13, 4)$ distribution. Therefore, most agents maintain between 9 and 17 connections. Note that redundant connections between a given pair of agents are not identified and are therefore counted multiple times. The decrease (from 20 to 13) in the average number of social connections indicates that the spatial interaction is not sufficient, in this case, to sustain 20 connections on average. This is dependent on the properties of the land-use data such as the concentration of high-utility areas. It also depends on the total number of agents in the system. (The area under the curve is equal to the total number of connections and thus constant.)

All the experiments were done on a computer equipped with a Intel Pentium 4 clocked at 2.5Ghz. The typical simulation performance for 100 iterations of a system with 10^6 agents is below one hour of CPU time. This is for plans that have only one or two intermediary stops. In terms of memory requirement, about 400 Mbytes of RAM are needed. Obviously, the simulation of larger systems and more sophisticated plans with more than two stops will require to distribute the workload on several computers. In particular, we would like to extend the framework to integrate other travel decisions such as the timing of the different trips.

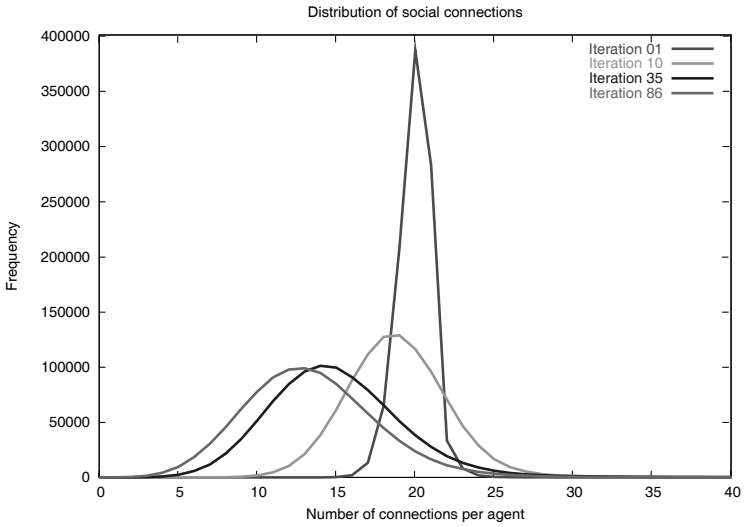


FIGURE 4. Distribution of social connections

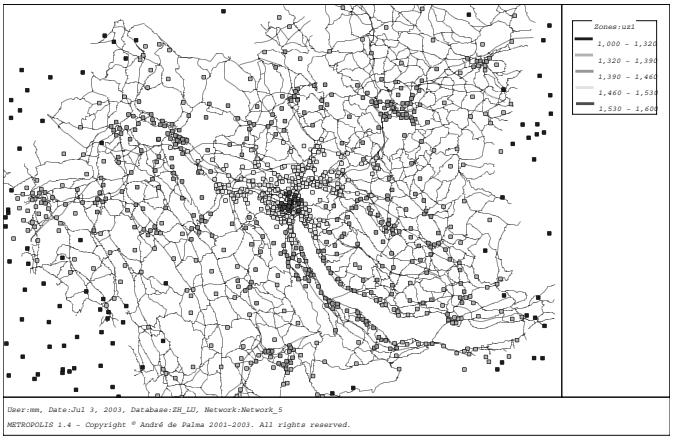


FIGURE 5. Zurich area: transportation network and rent values

9. Conclusions and perspectives

Many aspects still need to be addressed to improve the realism of the model. In particular, we plan to validate the model using the results of a recent survey [10] conducted in Germany and Switzerland. The goal was to study the factors that influence leisure trips. Respondents were asked to give the location and the frequency of the visits to their five closest friends or relatives. Empirical results suggest that the purpose of an important share of leisure trips is to visit social connections and that the number of known locations is small. These data should, in principle, allow to validate the model and to compare the spatial distribution of social connections with that of our model. This preliminary work has shown that a multi-agent based approach to the location of secondary activities is technically feasible and behaviorally plausible for high resolution data sets. The fact that agents cooperate in even some simplistic way yields an important gain in term of computation workload. This has to be compared with the standard practice in Transportation Science where it is typically assumed that users are in the situation of a non-cooperative Nash equilibrium.

10. Acknowledgments

The authors express their thanks to Prof. K. Axhausen and PTV AG for providing them with the data for the Zurich area. The authors acknowledge the extremely helpful comments of the anonymous referees on the earlier version of this article. Credits go also to A. Altenhoff who wrote a first implementation of the simulation. Computer resources for the simulations were provided by the Computational Laboratory (CoLab) at ETHZ.

References

[1] P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, G. Ulfarsson, Microsimulation of urban development and location choices: Design and implementation of UrbanSim, Networks and Spatial Economics 3 (1) (2003) 43–67.

[2] K. Axhausen, A. Zimmermann, S. Schönfelder, G. Rindsfüser, T. Haupt, Observing the rhythms of daily life: A six-week travel diary, Transportation 29 (2) (2002) 95– 124.

[3] B. Raney, N. Cetin, A. Völlmy, M. Vrtic, K. Axhausen, K. Nagel, An agent-based microsimulation model of Swiss travel: First results, Networks and Spatial Economics 3 (1) (2003) 23–41, similar version Transportation Research Board Annual Meeting 2003 paper number 03-4267.

[4] A. de Palma, F. Marchal, Real case applications of the fully dynamic METROPOLIS tool-box: an advocacy for large-scale mesoscopic transportation systems, Networks and Spatial Economics 2(4) (2002) 347–369.

[5] M. Ben-Akiva, S. R. Lerman, Discrete choice analysis, The MIT Press, Cambridge, MA, 1985.

[6] T. A. Domencich, D. McFadden, Urban travel demand, in: D. Jorgenson, J. Waelbroeck (Eds.), Urban travel demand, no. 93 in Contributions to Economic Analysis, North-Holland and American Elsevier, 1975.
[7] T. Arentze, H. Timmermans, Representing mental maps and cognitive learning in micro-simulation models of activity-travel choice dynamics, in: Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR), Lucerne, Switzerland, 2003, see http://www.ivt.baum.ethz.ch/allgemein/iatbr2003.html.
[8] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence : From Natural to Artificial Systems, Santa Fe Institute Studies on the Sciences of Complexity, Oxford University Press, 1999.
[9] F. Marchal, A trip generation method for time-dependent large-scale simulations of transport and land-use, Networks and Spatial Economics (2005) forthcoming.
[10] R. Schlich, A. Simma, K. W. Axhausen, Determinanten des Freizeitverkehrs - Modellierung und empirische Befunde, Arbeitsberichte Verkehrs- und Raumplanung 190, Institut für Verkehrsplanung und Transportsysteme (IVT),ETH Zürich, see www.ivt.baug.ethz.ch (2003).

Fabrice Marchal
Computational Laboratory
Swiss Federal Institute of Technology Zurich
Hirschengraben 84
CH-8092 Zurich
Switzerland
e-mail: marchal@inf.ethz.ch

Kai Nagel
Inst. for Land and Sea Transport Systems
Technical University Berlin Sek SG 12
Salzufer 17-19
D-10587 Berlin
Germany
e-mail: nagel@vsp.tu-berlin.de

Multi Agent System Simulation for the Generation of Individual Activity Programs

Guido Rindsfüser, Franziska Klügl and Jörg Freudenstein

Abstract. Due to advances in multi-agent simulation, realistic traffic microsimulation can be based on detailed models of traveller's behaviour. Paradigms like the activity-based approach provide solutions at the level of traffic generation or transport demand modelling. This forms a prerequisite for any realistic traffic simulation.

In this contribution, we present a behavioural agent architecture that is used to reproduce the daily activity scheduling behaviour of individuals. Depending on the agents' individual attributes like age, gender or employment status, specific tasks (activities or trips are assigned out of a set of habitual programs. This initial program has gaps and abstract tasks that are more and more concretized during the course of the day according to the agents' situational context, its interaction with other agents and also due to unforeseeable traffic conditions during travelling.

The implementation is based on empirical findings where possible and heuristics as well as stochastic assignments where necessary. Only when implementing theoretical concepts, problems related to complex behaviour models, like available data, data formats, parameter calibration, simulation speed etc. occur. Thus, new research questions arise for both, agent simulation technology and transport demand modelling.

1. Introduction

Traffic and transport modelling is done on several aggregational levels and for different purposes, e.g. aiming at the estimation of effects of transport planning measures to support decision processes. A widely and well investigated problem concerns modelling the traffic flow in combination with the simulation of route choice behaviour. In this contribution, traffic demand simulation is tackled. Models and simulations of transport demand – based on individual human decision making – form the basis for many other microscopic traffic models, as they deal with

reasons for mobility. A currently prominent approach is activity scheduling, with the objective of modelling daily activity program generation.

The importance of individuals' activities as objects of examination in transport research, the view on travel as derived from activities, and also the shift from vehicles to people (households) as the behavioural units is underscored in several reviews in the literature (e.g. [21]).

The renewed interest in activity scheduling, within the framework of the activity based approach, aims at understanding the underlying mechanisms that give rise to activity sequencing over the course of a day or a week [10]. Recent successfully deployed surveys on real-world activity scheduling brought new insides into individual scheduling behaviour [12, 30, 32]. Based on this empirical data and on first concepts of activity scheduling models, applications were developed and tested recently (e.g. [24]). It can be hoped that these new models of activity scheduling will improve transport demand modelling due to the higher behaviorual soundness.

Agent-based simulation seems to form the ideal modelling paradigm for activity scheduling models as it supports individual decision making, flexible interaction between agents and their environment and multi-level modelling and simulation. On the other side, this application area forms an interesting challenge for agent-based simulation technology as it requires an integrative view: a complex agent model for activity program planning based on local and global information and on interactions with other agents.

Interesting problems have to be solved when fixing the details: regarding not only an appropriate reactive planning mechanism, but also data management concerning thousands and more agents that are concurrently acting and interaction in an environment based on explicit, real-world maps.

Multi agent transport demand modelling is an interesting interdisciplinary research field per se. However, a combination or better, an integration with the adjacent field of traffic flow simulation would lead to a better consideration of the various existing interdependencies. Traffic flow simulation made enormous progress. Also, these models become more and more behaviourally rich (e.g. [25]) due to the fact that both, computers' calculation time is decreasing and computers' memory space is increasing. This leads to simulations of single units (vehicles) in large networks performed in real time or faster. Based on the agent concept an integration leads to several advantages: the input of traffic flow models would be improved in terms of a more realistic origin destination flow; detailed information about the start and end times, destinations and used modes of the travellers could be incorporated in the model. On the other way around, experiences gathered by the travellers during their way, like e.g. travel times, could lead to improved activity scheduling. Thus, such an integrated approach could enable a really context sensitive decision model with carrying out simulated activities and trips.

The rest of the paper is organized as follows. Section 2 comprises a brief overview of current research related to transport demand and activity scheduling. In section 3 we'll give a short introduction to the used modelling and experimenting

environment SeSAm, followed by describing the agent architecture and behaviour generation in section 4 and some short remarks on technological challenges in section 5. The conclusions in section 6 will close this paper.

2. Modelling Transport Demand

2.1. Transport Demand

Using the term *traffic generation* all processes which lead to individuals demand for location changes are summarized. This transport demand can be differentiated in terms of transport modelling into two major fields. From the aggregated economic view, transport demand is a national or regional measure, seldom spatially related and thus not differentiated in terms of origin and destination transport demand. Models for estimation of transport demand on that aggregated level are often used for long term trend predictions. From the disaggregated planning view, transport demand is always spatially related and seen as a demand of a single person or a group of persons with assumed homogenous behaviour within. The differentiation of transport demand between origin and destination (and other attributes) allows the usage of models to support decisions related to development of infrastructure and other planning decisions and their influence on individual behaviour. A standard description of transport demand is given by the following function:

$$F_{ij} = \sum_k F_{ij}^{(k)} = \sum_k \sum_p \sum_t \sum_m \sum_r F_{ijtmr}^{(kp)} \tag{1}$$

with F = trip
i = origin
j = destination
k = group of persons with similar behaviour
p = trip purpose
t = time
m = mode
r = route

Here, a trip is seen as an activity of a person to get from one location to another without carrying out another activity in between [5]. This assumption is made for a simplified handling within the model. It facilitates the treatment of trips which are definitely activities e.g. walking the dog or biking around a lake. The dependencies between some activities and trips are not negotiated. While generating a complete activity pattern it makes sense to view a trip as an activity because the whole sequence of different tasks (activities and trips) of a day is modelled (see figure 1).

Modelling transport demand also means to take into account the "classical" (normally sequential) steps of traffic generation, namely destination choice and

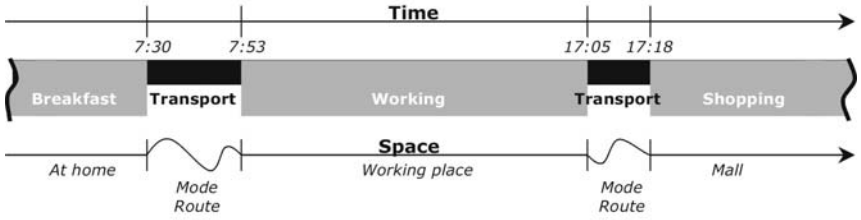


FIGURE 1. Part of an activity sequence [29]

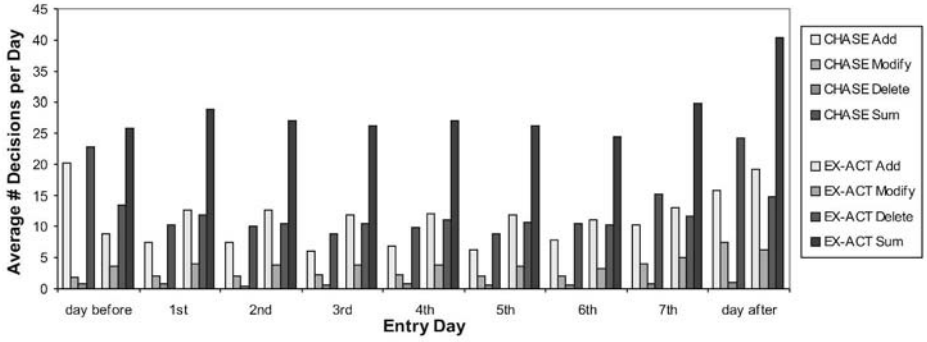


FIGURE 2. Comparison of amount of decisions during the course of a week, between CHASE and Ex-ACT [31]

mode choice. Modelling route choice and traffic flow is not viewed in detail here; as mentioned before, a number of well investigated models and simulations exists.

2.2. Activity Scheduling

As mentioned above, a relatively new field in transport demand modelling research in the framework of activity based approaches [2, 3] is the so called activity (re-)scheduling [9, 10, 4]. Based on empirical work concerning the planning and scheduling processes of individual activities (see for example [8, 30, 12]), the concept of activity (re-)scheduling aims at understanding and modelling the behavioural mechanisms underling these processes. Recent surveys provide for example information about the amount of planning decisions during the course of a week (see fig. 2) or the sequence and timely positions of decisions (fig. 3). For a comparison of the CHASE and EX-ACT surveys see [31].

Observing and tracing individual behaviour using new technologies like for example handheld devices with integrated GPS and asking respondents for the reasons behind the processes is one step towards improvement of models of activity (re-)scheduling [13]. Another major step is formed by new simulation technologies

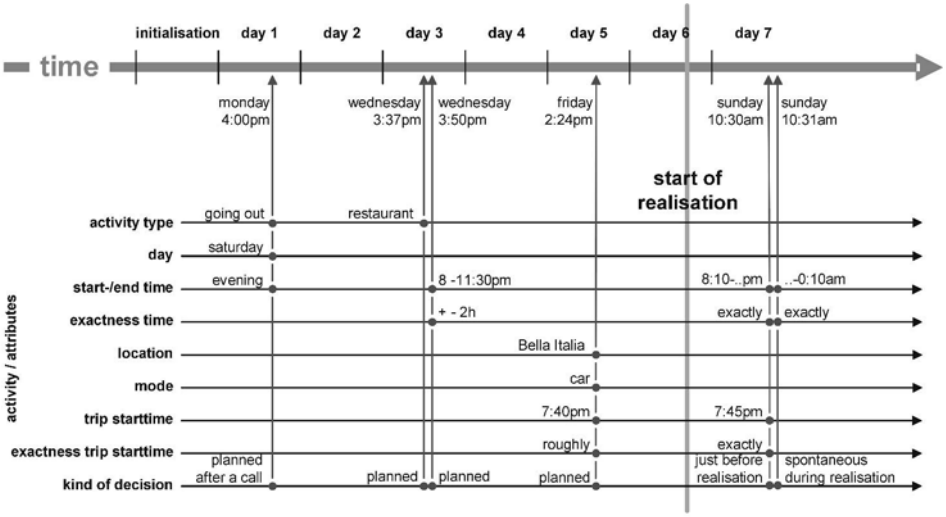


FIGURE 3. Individual decision process for planning one activity [30]

as for example the agent concept. Regarding the individuality and variations in individual behaviour, this approach seems to be straightforward [6].

So far, most models reproduce some kind of more or less complete and fixed activity patterns or at least major components of more complex patterns. The generation of individual activities and their combination to activity patterns is a relatively new approach (e.g. see [24]). The architecture described in this paper is a consequent attempt to build a context sensitive agent based model for the generation of individual activity programs. It is based on the research and findings so far incorporating the specific advantages of the agent concept.

Thus, transport demand modelling, especially the simulation of the activity scheduling processes, is a relatively new research field. Summarizing, modelling the individual demand for transport activities as well as the generation of activity patterns seems to call for an agent based simulation. This is also supported by the fact that society is more and more becoming heterogenous, individuals activities diversify as observed activity patterns do. Thus, new models for the estimation of transport demand are required. Traditional models are able to depict only a small range of variation in behaviour [18, 27]. With the multi agent approach, one should be able to simulate activity patterns with more behavioural soundness and cover a broader range of behaviour adaptations due to new transport measures.

3. SeSAm Modelling and Simulation System

The model and experiments described in this paper are implemented and conducted using the multi-agent simulation environment SeSAm. The "Shell for Simulated Agent Systems" provides a generic environment for modelling and experimenting with agent-based simulations. Its development was specially focused on providing a comfortable completely visual tool for the construction of complex agent-based models [20, 26].

The declarative nature of model representation enables modelling and simulation support far beyond "traditional" tools. However, this is connected to some specialities that influence modelling, like e.g. the usability of real-world maps [33]. Due to space limitations we have to restrain from describing the relevant aspects. More information about SeSAm can obtained from *http://www.simsesam.de*.

4. The "Scheduling Agent" Model

The "Scheduling Agent" is a complex system of models of individual behaviour related to the scheduling processes combined into an agent architecture. It aims at the generation of activity patterns of individuals. The agent model is designed using SeSAm. Here, the the first implementation and tests are described. At this level of development, it was the major objective to bring the two approaches, the transport related scheduling process approach and the IT related agent approach, together and to demonstrate the feasibility and advantages. Because of the complexity and the amount of necessary data and detailed models of individual decision making and the poor available empirical data, several used partial models are kept relatively simple according to their assumptions, heuristics, rules and stochastics, although more sophisticated approaches are known and well documented (see for example [17]).

4.1. Model Concept

The basic concept goes back to the ideas of a unified modelling framework for the scheduling process proposed by [11]. Following this suggestion, [28] processed the original idea and showed for one facet of the concept some related data analysis.

The input is a data set containing descriptions of a synthetic population, a synthetic city and some behavioural parameters. In a first modelling step the "activity repertoire" and the "habitual programs" are generated. The repertoire is a list of activities with various sets of distributions for every attribute value, as for example the start time. For different combinations of individual socio-demographic attributes a contiguous set of activity attribute value distributions can be found. The habitual program is also a set of routine/habitual activities with contiguous attributes.

A second modelling step concerns the scheduling processes. At the beginning of the simulation of the scheduling process, an initial current program is generated

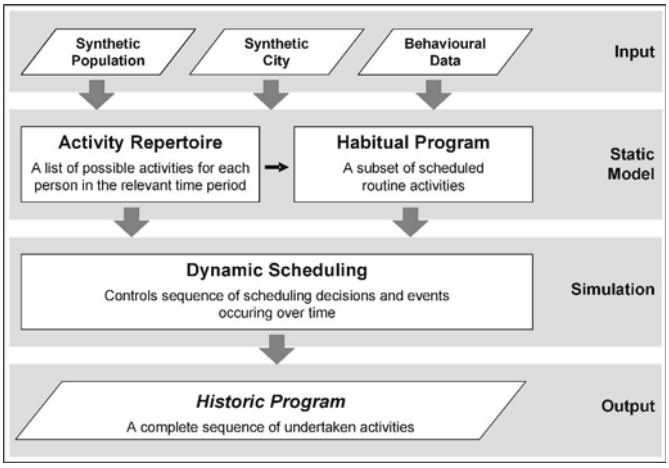


FIGURE 4. Schematic representation of the simulation concept [28]

from the habitual one adding and fixing some variable values. This current program can be seen as a skeleton. It serves as representation of habitual or routine behaviour which can be modified during scheduling. The simulated scheduling may happen at every time step during the execution of the current program, during the simulated period (a day, a week, etc). The output of the overall simulation is a "historic program", that means a sequence of all activities and trips the individual has undertaken during the simulation run.

One of the major benefits of grounding the generation of activity patterns on agents is the possibility to integrate the situational context of each individual into the model. The agent may act in reaction to a modification in his environment or depending on his own status or the status of other agents at every time step during the simulation. All actions of the simulated agents result in changes in the environment, agents' own attributes or attributes of the other agents. An example is depicted in figure 4: The agent possesses an activity repertoire and habitual programs. At the beginning of the simulated day, a skeleton from the habitual program is taken and extended. With time proceeding, activities are executed and the future program of the day may be modified. At the end of the simulated day, the program as it happened is known and thus becomes a "historic program".

4.2. Implementation

The main objective during the design process of "The Scheduling Agent" using SeSAm was to create a generic agent or agent architecture and not to design different types of agents. The individuality of the agents should be represented by the individual set of attributes of every individual agent. The meta level behaviour – scheduling and executing activities – was defined in one agent class. Therefore, a

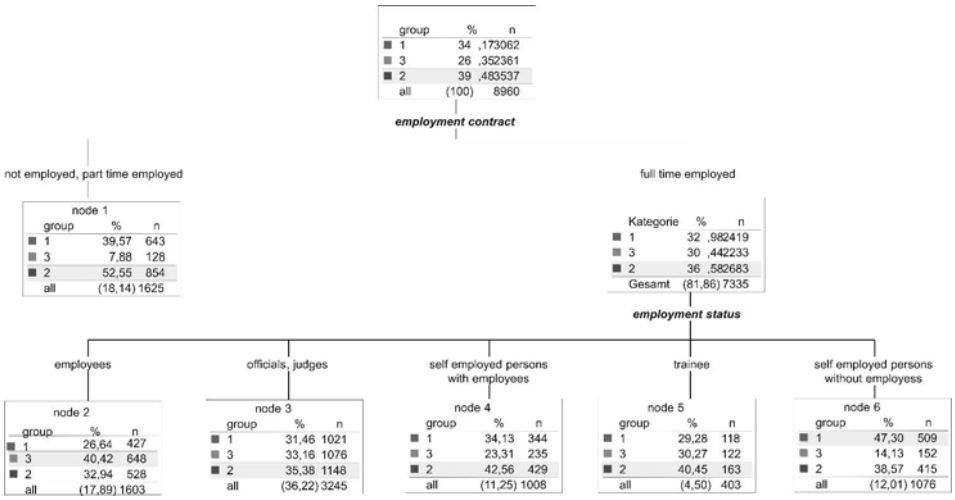


FIGURE 5. Decision tree example for determining the data set and individual attribute values for a particular simulation run

simulation situation can be generated with all different agents as instances of the designed agent. Due to this, all agents have the same generic decision making.

4.3. Generation of Artificial Population

Based on empirical data ([6, 34]) various rules to assign individual attribute values were derived from decision trees using the CHAID algorithm [2]. An example of an output of this method is displayed in figure 5.

As can be seen, the complete dataset is classified into 6 leaf nodes. Two predictors determine the output. First, all cases are split into two categories based on the variable employment contract. The category of the full time employed are further subclassified into 5 categories based on the variable employment status. The responsive variable, type of work activity, has 3 different groups (categories). The percentage of response belonging for each group is calculated.

For example, in leaf node 1, 39,57% of respondents belong to group 1, 52,55% to group 2 and 7,88% to group 3. Is a case characterized by leaf node 1 (part time employed or not employed), the person will be assigned to group 2 with probability p=0,53 and to the other groups with accordant probabilities. The results (the rules) of this procedure could be represented in rule format as illustrated with the following examples:

Rule1: IF employment contract = not employed OR part time employed THEN node=1

Rule2: IF employment contract = full time employed AND employment status = employee THEN node=5

Rule3: IF employment contract = full time employed AND employment status = officials OR judges THEN node=6

The above example is for the activity *work*. The groups and their parameters were found using a cluster algorithm:

Group 1: mean start time 2:19 pm, mean duration 176,19 minutes
Group 2: mean start time 7:39 am, mean duration 236,14 minutes
Group 3: mean start time 7:12 am, mean duration 539,13 minutes

For each activity this procedure results in some decision trees with assignment depending on various combinations of socio-demographic variables of the person and additionally stochastic assignment because of the underlying distributions of start time and duration.

4.4. Agent Behaviour

The generic behaviour of the agents is specified using a behaviour network, like an enhanced UML-Activity graph. It is depicted in figure 6. At first sight, the graph seems to be quite simple. The main information is stored in the status of an agent. An agent can be either in an activity or during a trip. Is a person in one of these states, and only then, simulation time is advanced. All other actions are only containers of more or less inner processes. They are executing during every activity or trip excluding the node "out of simulation", which is only implemented to get a defined simulation ending.

An instrument to generate some kind of real situational context is the usage of so called triggers. As asked for in the CHASE [8, 32] and EX-ACT [30] empirical data, several reasons to start of a scheduling process exist. Some of the most frequently observed reasons for scheduling (compare [22]) are used in the simulation to cause a transition from "during activity" or "during trip" to "schedule". In the following, the most important components of the behaviour graph are explained in more detail. The central components in this graph are *Current Task*, *During Activity (Trip)* and especially *Schedule*.

- *Current Task*: Depending on the current task the next actions of the agent are determined. Using the time of the world, the current task is determined out of the agents' current program as the task with $starttime <= currenttime$ and with $endtime >= currenttime$.
  - If the current task is an activity, a transition to "during activity" is done via the node "update historic program" that simple saves what happened so far.
  - Is the current task a trip, the agent's state in the next simulation step is "during trip".
  - If the current task is empty the agent adds a new activity and continues with the node "schedule".
- *During Activity/Trip*: The agent is in this state during an activity, during a trip respectively. The simulation time is advancing until the activity end time is reached and the agent leaves this behavior node. The agent is able

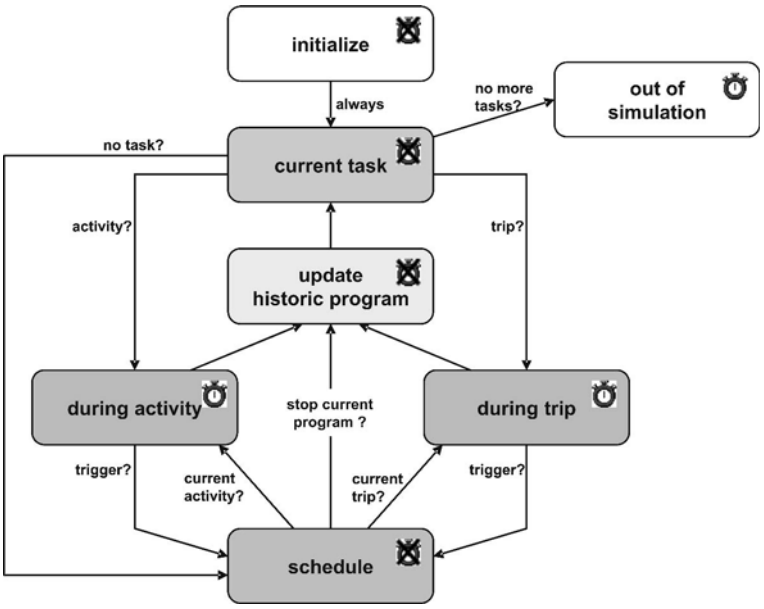


FIGURE 6. Schematic representation of the behavioural processes in an agent

to react in every simulation step on modifications of the situational context. The agent is continuously updating his variables and his knowledge of the environment with every simulation step. In the state "during trip" the agent moves in the environment to reach the destination where the next task has to be carried out. It may happen due to some triggers (interactions, perceptions, ...) that the agent needs to do some re-scheduling. If so, he leaves the node "during activity" and changes to the node "schedule". After finishing his scheduling, the agent continues with his current activity or trip. The current implementation of the agent movement is along the direct connection from one position to another with a calculated mean speed depending on the used transport mode.

- *Schedule*: This is the most complex module of the agent architecture. All scheduling related processes are simulated within this behavior node. Addition of new activities, modification or deletion of already scheduled (future) activities may happen while the agent executes this scheduling behavior. Because it is possible, that an already scheduled activity has to be modified again for some reason, re-scheduling of activities and trips can be necessary. For all these decisions a reasonable number of constraints and intentions needs to be taken into consideration (as for example location and mode choice or temporal overlapping with other, already scheduled, activities).

This node can be reached from every other node, comparable to real life, where planning and scheduling is not fixed to specific times of the day. Depending on the received triggers the agent needs to add a new task, modify or delete an already scheduled or the current task or their variables.

4.5. Experimentation and Preliminary Results

For the first test simulation runs different numbers of agents, households and facilities were used as artificial population and landscape. For test reasons we limited the number of agents to 200. We developed a SeSAm plugin that enables us to import data files containing description of agents, households and facilities in arbitrary amounts.

With the actual version of the model, a first level of agent deliberation is reached: Agents select their individual current program, fill the remaining free time windows by adding new activities out of a list of possible activities with distributions of start time and duration, do some constraint processing, like the resolution of overlapping newly scheduled tasks with existing tasks and finally try to carry out the scheduled tasks under consideration of the environmental context. As a result, complete historic programs (activity sequences) are generated. Figure 7 shows example historic programs from 5 agents processing relatively detailed activity categories, using a simple communication between the agents for meeting planning.

The results so far demonstrate that this approach is promising in terms of a better reproduction the behavioural inter- and intrapersonal variety for a sound traffic demand simulation.

5. Challenges for Multi-Agent Simulation Technology

An integrated traffic generation model – as the one presented here – leads to technological challenges. These issues mainly refer to performance of individual decision making - consisting of both, the treatment the habitual programs and activity repertoire by the individual agents as well as the actual simulation when these programs are continuously constructed, updated and executed. The decision making simulation of the agents mainly comprises of a meta-level behaviour that is the same for every individual. We implemented this decision making behaviour declaratively using the modelling language of SeSAm. The initialization of simulation runs in SeSAm contains some optimization steps from compiler construction theory. Whether these optimizations are sufficient for scaling up the simulations to thousands of agents will be tested soon. For more demanding simulations distributed approaches are currently developed.

Therefore, the main modelling effort is reduced to specifying habitual behaviour programs and assigning them to categories of agents. This can be facilitated by specialized data structures and data base connections. The agents individual status and context is responsible for the individual specific behaviour. This status

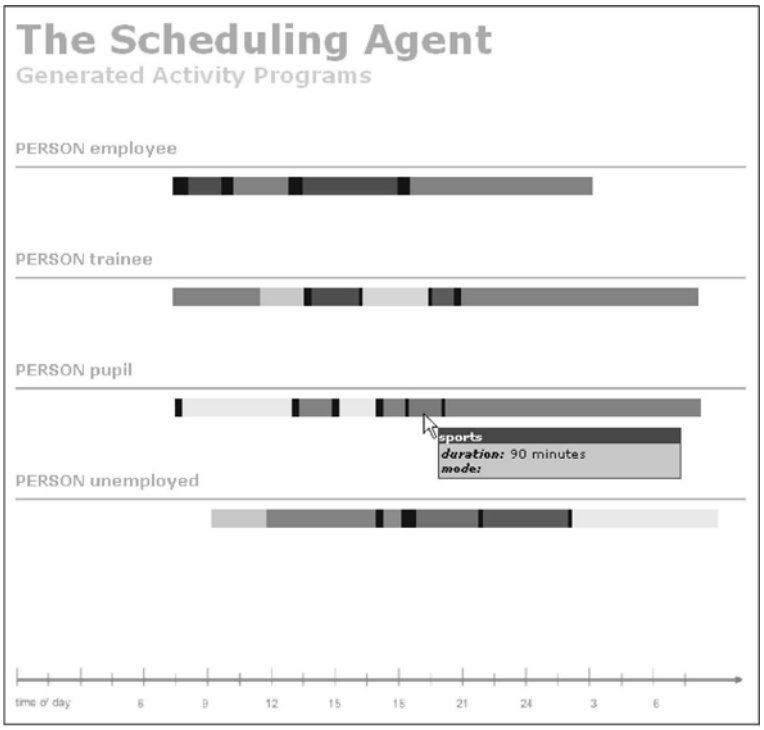


FIGURE 7. Examples of generated activity patterns

and context not only contains the locations and type of households or facilities, but also personal status, friends, etc.

The most important point for multi-agent simulation – like for every simulation – is the issue of data grounding and validity. Socio-demographic data is used for generating the synthetic population. Not only the configuration of agents, households and facilities has to be realistically set, but also the decision making processes are in need of empirical foundation. As mentioned before, data from projects acquiring individual scheduling processes was gathered recently for example in two projects (CHASE and EX-ACT) which hopefully can be used to fill the empirical gaps in this multi-agent model.

6. Conclusions

A multi agent simulation of the generation of individual activity programs was developed based on the ideas of a simulation of the scheduling processes. First tests were sketched.

It can be seen, that for such a micro simulation, other, mainly new data related to the demand as well as to the supply side of transport is needed. Data related to the scheduling processes and the related behaviour is essential as well as more detailed data about the infrastructure and especially temporal organization of infrastructure. Although only poor data were available and a great amount of behavioural data was only assumed, the results shows, that using rules based on expert knowledge, some functional descriptions and empirical data where possible, satisfying results can be obtained.

To improve the estimation of realistic activity patterns, a better understanding of mobility related to individual decision processes is needed on the one hand, an integration with state-of-the-art traffic flow micro simulation is important on the other hand. Both is possible due to emerging technologies to observe real human behaviour and due to the underlying agent concept. The current version of the agent architecture and the simulation model already demonstrates the feasibility of this approaches in transport modelling. Nevertheless, the complexity and the uncertainty in human behaviour still requires a lot of research to build a valid model of activity (re-)scheduling.

References

[1] Adler, J.L. and V.J. Blue (2002) A cooperative multi-agent transportation management and route guidance system. *Transportation Research C*, 10(5-6), 433-454.

[2] Arentze, T. and H. Timmermans (2000) ALBATROSS: A Learning - Based Transportation Oriented Simulation System. Technische Universiteit Eindhoven, EIRASS, Den Haag.

[3] Axhausen, K. W. and T. Gärling (1992) Activity-based approaches to travel analysis: conceptual frameworks, models, and research problems. *Transport Reviews*, 12(4/92), 323-341.

[4] Axhausen, K. W. (1997) Data needs of activity scheduling models. In: Ettema, D. F. and H. J.P. Timmermans (Eds.), Activity-based approaches to travel analysis Pergamon, Eindhoven, pp. 229-242.

[5] Axhausen, K. W. (1999) Definition of movement and activity for transport modelling Handbooks in Transport: Transport Modelling.

[6] Axhausen, K. W. et al. (2002) Observing the rhythms of daily life: A six-week travel diary. *Transportation*, 29(2), 95-124.

[7] Dia, H. (2002) An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research C*, 10(5-6), 331-349.

[8] Doherty, S.T. and E.J. Miller (1997) Tracing the household activity scheduling process using a one week computer-based survey. In. *8th Meeting of the International Association for Travel Behaviour Research: Challenges and Opprotunities in Travel Behaviour Research and Applications*, September 21-25, Austin, Texas.

[9] Doherty, S.T. et al. (2002) A Conceptual Model of the weekly Household Activity-Travel Scheduling Process. In: Bovy, P.,I. Salomon und E. Stern (Eds.), Travel Behaviour: Patterns, Congestion and Modelling. Edward Elgar Pub.

[10] Doherty, S.T. (2000) An activity scheduling process approach to understanding travel behaviour. In. *79th Annual Meeting of the Transportation Research Board*, January 9-13, Washington D.C.

[11] Doherty, S. T. and K. W. Axhausen (1999) The Development of a Unified Modeling Framework for the Household Activity-Travel Scheduling Process. In: Brilon, W. et al. (Eds.), Traffic and Mobility: Simulation - Economics - Environment, Springer Verlag, Berlin.

[12] Doherty, S. T. et al. (2004) Design and Assessment of the Toronto Area Computerized Household Activity Scheduling Survey. In. *83rd Annual Meeting of the Transportation Research Board*, January 11-15, Washington D.C.

[13] Fischer, J. and G. Rindsfüser (2004) A Concept for an integrated GPS Tracing and Activity Scheduling Survey. SRL, 76, Institut für Stadtbauwesen und Stadtverkehr, RWTH Aachen, Aachen.

[14] Fischer, K., H.-J. Bürckert and G. Vierke (2000) Holonic Transport Scheduling with TELETRUCK. *Journal of Applied Artificial Intelligence*, 14, 697-725.

[15] Hernández, J.Z., S. Ossowski and A. Garcia-Serrano (2002) Multiagent architectures for intelligent managment systems. *Transportation Research C*, 10(5), 473-506.

[16] Hidas, P. (2002) Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research C*, 10(5-6), 351-371.

[17] Joh, C.H., T.A. Arentze and H.J.P. Timmermans (2004) Activity-travel rescheduling decisions: Empirical estimation of the Aurora model. *To appear in Transportation Research Record*.

[18] Kitamura, R. (1996) Applications of Models of Activity Behaviour for Activity Based Demand Forecasting. In. *Proceedings of the Activity-Based Travel Forcasting Conference*, June 2-5.

[19] Klügl, F. and A.L. Bazzan (2004) Route Decision Behaviour in a Commuting Scenario: Simple Heuristics Adaptation and Effect of Traffic Forecast. *JASSS*, 7(1).

[20] Klügl, F. (2001) *Multiagentensimulation*. Addison-Wesley Verlag, München.

[21] Kurani, K.S. and M.E.H. Lee-Gosselin (1996) Synthesis of past activity analysis applications. Proceedings of the TMIP - Conference on Activity-Based Travel Forecasting, New Orleans, Louisiana.

[22] Litwin, M. S., S. T. Doherty and E. J. Miller (2004) Investigating competition patterns in household activity scheduling processes. In. *83rd Annual Meeting of the Transportation Research Board*, January 11-15, Washington D.C.

[23] Ljungberg, M. and A. Lucas (1992) The oasis air-traffic management system. *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, Seoul, Korea.

[24] Miller, E. J. and M. J. Roorda (2003) A Prototype Model of Household Activity/Travel Scheduling. In. *82nd Annual Meeting of the Transportation Research Board*, January 12-16, 2003, Washington D.C.

[25] Nagel, K. (2003) Proceedings of the IATBR, Luzern, Schweiz.

[26] Oechslein, C. et al. (2001) UML for Behavior-Oriented Multi-Agent Simulations. In Dunin-Keplicz, B. and E. Nawarecki (Eds.), *Proceedings of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS 2001)*, Sept. 26-29, Cracow, Poland.

[27] Recker, W. W., M. G. McNally and G. S. Root (1986) A model of complex travel behavior: Part I - theoretical development. *Transportation Research A*, 20A(4/86), 307-318.

[28] Rindsfüser, G. and S.T. Doherty (2000) Konzept, Module und Datenerfordernisse für SMART - Simulationsmodell des Aktivitäten-(Re)Planungsprozesses. SRL, 69, Institut für Stadtbauwesen und Stadtverkehr, RWTH Aachen, Aachen, 109-129.

[29] Rindsfüser, G. (2001) Die Verwendung zeitbezogener Daten für die Analyse von Aktivitätensequenzen im Kontext der Verkehrsnachfragemodellierung. In: Ehling, M. und J. Merz (Eds.), Zeitbudget in Deutschland - Erfahrungsberichte der Wissenschaft Spektrum Bundesstatistik, Bd. 17, Statistisches Bundesamt, Wiesbaden, S. 58-77.

[30] Rindsfüser, G. et al. (2003) Tracing the planning and executing attributes of activities and travel - Design and application of a hand-held scheduling process survey. In. *10th International Conference on Travel Behaviour Research* August 2003, Lucerne.

[31] Rindsfüser, G., Sean T. Doherty and Heike Mühlhans (2004) Quality Assessment of Scheduling Survey Data In. *Conference on Progress in Activity-Based Analysis* May 2004, Maastricht.

[32] Roorda, M. and E. Miller (2004) Strategies for Resolving Activity Scheduling Conflicts: An Empirical Analysis. In. *Conference on Progress in Activity-Based Analysis* May 2004, Maastricht.

[33] Schüle, M., Herrler, R. and F. Klügl (2004) Coupling GIS and Multi-Agent Simulation – Towards Infrastructure for Realistic Simulation, In: G. Lindemann et al. (eds) Multiagent System Technologies, Proceedings of the Second German Conference MATES 2004, LNAI 3187, pp. 228-242

[34] Statistisches Bundesamt(1997) Wo bleibt die Zeit? Zeitverwendung der Bevölkerung in Deutschland. Scientific Use File der Zeitbudgeterhebung 91/92. Version 1.1, Wiesbaden.

Guido Rindsfüser
Emch+Berger AG Bern
Gartenstrasse 1, CH - 3001 Bern, Switzerland
e-mail: `Guido.Rindsfueser@EmchBerger.ch`

Franziska Klügl
Universität Würzburg
Institut für Informatik
Am Hubland, D-97074 Würzburg, Germany
e-mail: `kluegl@informatik.uni-wuerzburg.de`

Jörg Freudenstein
Rheinisch-Westfälische Technische Hochschule Aachen
Institut für Stadtbauwesen und Stadtverkehr
Mies-van-der-Rohe-Str. 1, D- 52076 Aachen, Germany
e-mail: joerg.freudenstein@epost.de

A Dynamic Network Simulation Model Based on Multi-Agent Systems

Rosaldo J. F. Rossetti and Ronghui Liu

Abstract. This paper reports on how the abstraction approach of multi-agent systems can be used to represent the complexity inherent in the urban traffic domain, accounting for the importance of modelling travellers' behaviour and their interaction with intelligent transportation technologies. A key premise in the approach proposed is the identification of what we have coined autonomous decision entity, which is defined as an agent shell to structure the way agents can be implemented and inserted into the environment. Such a structure is very flexible in the sense it is only defined in meta-level, comprising sensors, effectors and a reasoning kernel. The conceptual multi-agent model is presented and implemented within the DRACULA simulation suite, which is used for simulation experiments on the analysis of drivers' route and departure time choice.

1. Introduction

The problem of traffic congestion in urban areas has stimulated much interest not limited to traffic and transport engineering. The multidisciplinary nature of such a complex application domain has been increasingly discovered in different fields of knowledge, challenging researchers for new and novel approaches. At a first glance, the obvious problem is that the capacity of urban networks is no longer capable of meeting the more than ever increasing travel demand [16]. The answer to how that can be resolved is not straightforward, not least because it requires understanding of all the factors involved in a transport system and their interactions.

Earlier attempts to address issues on traffic congestion relied on increasing capacity of networks by means of physical modifications or on improving efficiency of existing road capacity through efficient traffic control and management. More recently, researchers and policy makers have oriented their efforts towards more directly influencing the travellers' behaviour patterns. This is an important premise

of the Intelligent Transportation Systems (ITS), which use advanced technology to try to manage and to influence traffic and driving behaviour [2].

The use of autonomous and intelligent technologies allied to the human behaviour is now central in traffic and transportation systems. This has brought about the need for assessing other performance measures, which demands more powerful and expressive modelling and simulation tools. Thus, much work has been carried out either to adapt traditional approaches or to develop new-generation traffic network models to meet ITS requirements (see a review in [20]). More recently, a new paradigm in Computer Science has been widely spread and used by researchers and practitioners: the Multi-agent Systems (MAS). Given their ability to allow for a social side of computer systems, agent technology is a natural metaphor for building a wide range of applications, including naturally the complex traffic and transport systems (see the Guest Editorial by Schleiffer [17]).

This paper reports on how the abstraction approach of MAS can be used to represent the complexity inherent in urban traffic systems, accounting for the importance of modelling travellers' behaviour and their interaction with ITS-based technologies. This research continues our effort to improve the modelling of human behaviour in demand representation (see [13] and [14]). The conceptual idea is implemented within a dynamic network simulation model framework, DRACULA [7], to generate an enhanced multi-agent traffic simulation tool. The paper presents the multi-agent simulation framework and provides example simulation experiments on the analysis of drivers' route and departure time choice.

2. MAS and its potential applications to ITS

MAS is under the umbrella of the Distributed Artificial Intelligence (DAI) and has inspired increasing interest among scientists from different knowledge fields. The rapid evolution in computational resources, both in hardware and in software, has contributed a great deal to its development. The increasing demand for suitable tools to aid the analysis of some complex application domains has motivated much research on the agent-based technologies.

The main premise in multi-agent systems is to model real world in terms of agents that exhibit intelligence, autonomy, and some degree of interaction with other agents and with its environment. Other characteristics of agents include, for example, reactivity, adaptability, pro-activity, and the ability to communicate and to behave socially. The basic structure of an agent features "sensors" through which it can gather information from the environment, and "effectors" through which it can act and behave according to its objectives [15]. This structure can feature both reactive and cognitive abilities, and a mixture of both, to mimic human behaviour in a wide range of applications. Steels [18] suggests that each single agent albeit possibly having a very simple structure can contribute to a more complex and efficient behaviour of the system as a whole. If the behaviour of such

single agent can be backtracked, then this can be used to aid the understanding of the social phenomena.

Owing to all those characteristics and concepts, multi-agent systems have been widely experimented in modelling and simulation and have been applied to a wide range of applications including traffic and transportation engineering [17]. Not surprisingly, most works report on applying the agent-based techniques to control systems and traffic management to make those systems more autonomous and responsive to recurrent traffic demand (e.g. [4]). Nonetheless, an increasing effort has also been dedicated to representing driver behaviour and its underlying decision-making mechanism (e.g. [3], [10], [13], [14]). The analysis of ITS systems through this approximation has been investigated as well (e.g. [12], [19]), and some other works report on applications to freight transport and optimisation of resource use (e.g. [1]).

3. Representing urban traffic as a multi-agent system

3.1. The agent structure

The perspective of allowing the representation of individual elements of a system has inevitably associated multi-agent systems to microscopic simulation. In order to devise a framework where we can conceptualise the traffic domain in terms of a multi-agent system we started with two concepts which are of central importance, namely the day-to-day decision-making and the within-day dynamics. While the former relates to choices individuals make to perform daily journeys (demand formation), the dynamic formulation on the other hand concerns the modelling of how the state of the network changes from day to day and evolves over time (supply variability).

Considering these concepts and accounting for the increasing use of ITS technologies, it is possible to identify which components of the domain we might model as agents. With this purpose, we use a simple rule in our approach: entities that play a decision role in an autonomous fashion are considered to be autonomous decision entities (ADE) and therefore are potential agents. To avoid going too much into detailed specifications, the task of identifying agents within a system is basically reduced to the identification of ADEs. Surveillance systems would be simply considered as the sensor part of an agent structure, for instance.

We have devised an agent shell to structure the way agents can be implemented and inserted into the environment. Such a structure is very flexible in the sense it is only defined in meta-level, comprising sensors through which the agent can perceive the world and effectors through which it can effectively act. It also has a reasoning kernel that drives the decision-making processes. It is important to notice that this meta-level agent shell only specifies the basic structure for the ADEs, allowing the definition of different kinds of agents with different reasoning capabilities, skills, and goals. Communication among agents is simply considered

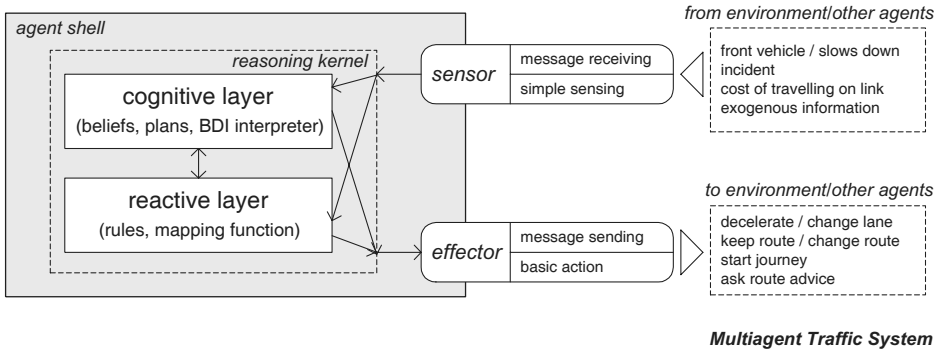


FIGURE 1. A two-layered architecture for the driver agent.

to be an act/sensing behaviour. All messages are issued as actions, through effectors, and received as perceptions, through sensors. The environment is basically formed by the network topology and parameterises all the information shared by the inhabitant agents. Traffic signs and basic rules are considered part of the environment structure and dynamics.

With this conceptualisation it is possible to virtually represent all aspects involved in contemporary traffic scenarios: drivers are agents in the sense they make their decisions concerning route and departure time; travellers are agents as they have to opt among transport modes; each level of decision in an advanced traffic management system could be an agent that interacts directly or indirectly with the others in order to optimise overall traffic performance; in the same way, traveller information systems could be agents interacting with drivers or travellers in order to optimise individual performance levels; and so forth. Albeit all these ADEs are encapsulated into agent shells, they may be internally different, implementing distinct reasoning approaches and having different knowledge representations.

To demonstrate our approach we have started by modelling the driver agent whose structure is depicted in Figure 1. We have designed a two-layered reasoning kernel to base the driver model so that it is able to exhibit both reactive and cognitive behaviours to some extent. The reactive layer relies on a simple set of rules that map perceptions to actions. Individual's driving abilities, in terms of car-following and lane-changing behaviours, are performed in this layer. The more complex decisions, such as whether to travel, which itinerary to follow, and what time to start the journey are addressed in the cognitive layer. For the cognitive approach we use the belief, desire, and intention (BDI) architecture described by Rao [11], which deals with reasoning on the basis of those metal states and their relations. According to Rao [11], the reasoning kernel of the driver agent can be represented by the tuple $\langle E, B, P, I, A, S_E, S_O, S_I \rangle$, where E, B, P, I and A are sets of events, base beliefs, plans, intentions and basic actions respectively. S_E, S_O and S_I are the selection functions for events, applicable plans and intentions. The

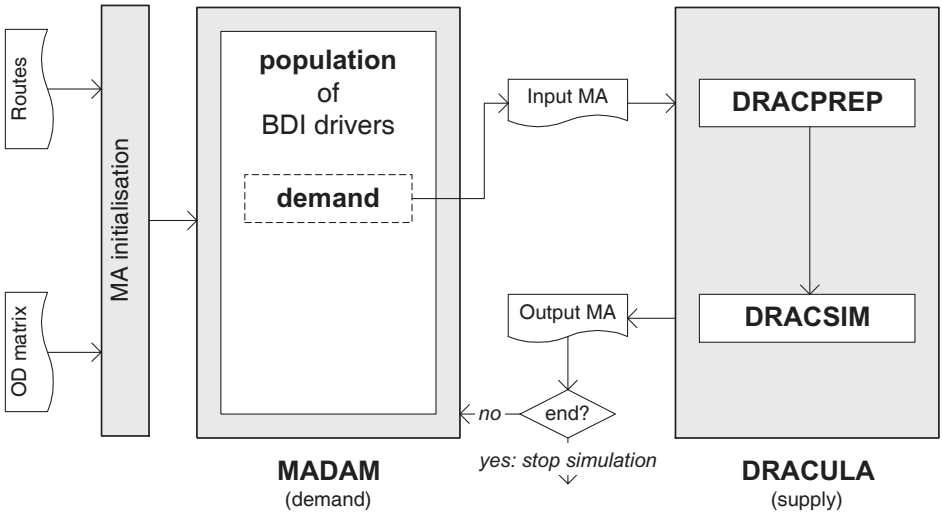


FIGURE 2. The MADAM+DRACULA simulation framework.

task of defining an agent is then reduced to identifying the sets of base beliefs and plans. Intentions are generated dynamically, as triggering events are selected.

3.2. An agent-based model for demand analysis: the experimental framework

The conceptual multi-agent model presented was implemented within the DRACULA simulation suite, as depicted in Figure 2. DRACULA is a microscopic network simulator that has been developed in the Institute for Transport Studies, at the University of Leeds [7]. It comprises basically a demand and a supply model to implement the concepts of day-to-day decision-making and within-day dynamics. Contrary to the approaches based on fixed matrix structures, the demand stage predicts the level of individual demand on a certain day from a full population of potential drivers. In the supply side vehicles individually move throughout the network to their destinations, and drivers are able to gather information about trip conditions to improve future choices. The main choice dimensions available in DRACULA are route and departure time, although its modularity allows for the development of others, such as an en-route diversion capability.

MADAM is the multi-agent demand model that replaces the original *Demand* side of the DRACULA suite. Rather than representing travel choices through variable values in a simple data structure, defined beforehand, demand results from the cognition mechanism of each single driver agent that commits to intentions that are generated dynamically from non-instantiated options. JAM [5] was used to underlie the implementation of the cognitive kernel. On the *Supply* side, the traffic dynamics is represented microscopically based on modelling individual vehicles' car-following and lane-changing behaviour [6]. *Dracprep* sets environment

conditions on each day (road capacity may vary due to weather, parked cars, and accidents, for instance) whereas *Dracsim* conducts the microscopic simulation of each individual vehicle's movement through the network. The *MA Initialisation* module synthesises the population for the experiment from an *OD matrix* and route alternatives are assigned to each driver from a list of possible routes for each origin and destination pair.

The initial set of base beliefs for each driver agent of the population can be either generated after a first run of the *Supply* side, so that the usual desired arrival time can be either estimated or set to default values. The *Input_MA* file gathers drivers' decisions on route and departure time, so that they can be launched onto the network to perform their journeys at selected departure times on each day. On the other hand, the *Output_MA* file returns the travel costs experienced by each driver in terms of realised travel time (these are the perceptions of each driver during the course of the journey simulated in DRACULA's supply model) and the base beliefs sets are updated. On the following day, the driver uses his updated beliefs to make his decision and this process is repeated all over for a specified number of days, which is defined at the beginning of the simulation.

3.3. A simple example and simulation results

A simple example was set up to use the above model framework to analyse demand variability. The choice behaviour selected is based on the preferred arrival time model, as it is implemented in DRACULA [7]. Departure time is chosen in response to a traveller's previous experiences and preferred arrival time. The absolute delay for a driver m travelling from certain origin i to a destination j on day k is given in Equation 1, where $d_{ijm}^{(k)}$ is the departure time, $t_{ijm}^{(k)}$ is the travel time, and $a_{ijm}^{(k)}$ is the desired arrival time. As suggested by Mahmassani et al. [8], drivers are likely to be indifferent to early arrivals. Drivers are further assumed to be indifferent to a delay of $\epsilon_m t_{ijm}^{(k)}$, where ϵ_m is drawn from a uniform $[0, \epsilon]$ distribution. Equation 2 represents the lateness perceived by individuals.

$$\delta_{ijm}^{(k)} = d_{ijm}^{(k)} + t_{ijm}^{(k)} - a_{ijm}^{(k)} \tag{1}$$

$$\Delta_{ijm}^{(k)} = \delta_{ijm}^{(k)} - \epsilon_m t_{ijm}^{(k)} \tag{2}$$

Accounting for that fact, we consider that users only adjust their departure time for a future journey in the case of $\Delta_{ijm}^{(k)} > 0$, otherwise they will keep the same departure time. The adjustment is made in Equation 3.

$$d_{ijm}^{(k+1)} = \begin{cases} d_{ijm}^{(k)}, & \text{if } \Delta_{ijm}^{(k)} \leq 0 \\ d_{ijm}^{(k)} - \Delta_{ijm}^{(k)}, & \text{if } \Delta_{ijm}^{(k)} > 0 \end{cases} \tag{3}$$

An important simplification of this model is that drivers are virtually indifferent to early arrivals, which may not be so related to the reality of commuters. Other types of behaviour were also suggested and implemented according to this approach [13]. The route choice is based on the bounded rational behaviour (e.g.

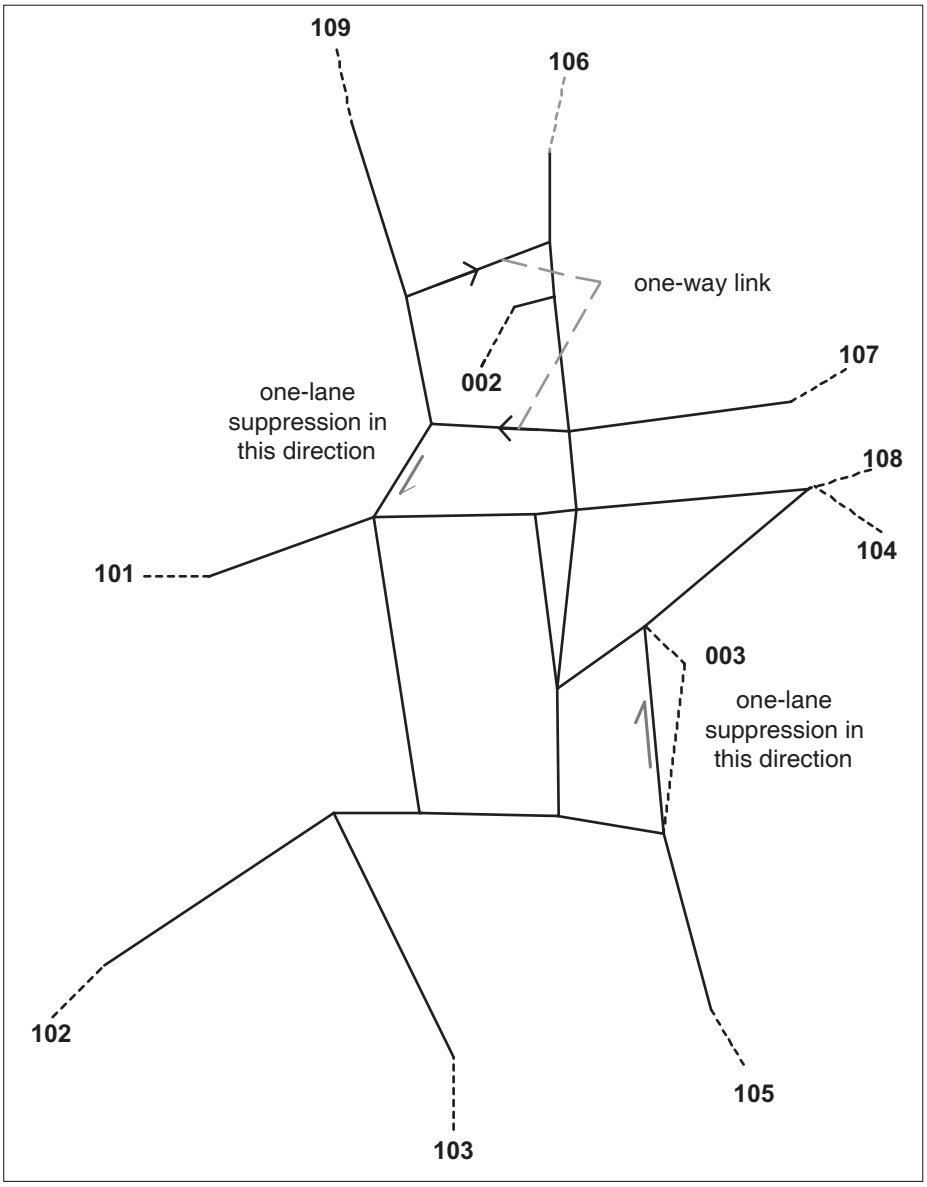


FIGURE 3. Schematic representation of the experimental network (numbers represent zone codes for origins and destinations).

[9]), where drivers are assumed to use their habit routes as on the last day, unless the cost expected for the minimum cost route is significantly better.

A small network with 54 links connecting 14 junctions was elected (see the schematic representation of the network in Figure 3 and a snapshot of the simulation environment in Figure 4). Most road junctions follow a priority regime; two of the intersections are controlled by traffic signals. In this simple scenario, demand is generated from a population of 2,323 agents corresponding to the total number of trips in the OD (origin-destination) matrix for the selected network. Each agent's day-to-day choices on route and departure time are simulated as the agents perform their trips to/from 11 zones (i.e., there are 11 zones generating traffic onto and 11 zones draining traffic from the network). At the beginning of each day, users of pre-trip information systems are supplied with updated information on the prevailing conditions of the network, so that drivers can revaluate their choices. A hypothetical morning peak period starting at hour 8 is considered and the simulation is carried out from day 0 to day 100. Two incidents were introduced (in terms of one-lane suppression in the links indicated in Figure 3) that were programmed to start on day 50 and to last for the remaining peak period until day 100. Different fractions of informed drivers were considered in different runs of the experiment, which represent the percentage of drivers that effectively use the information provided. After being informed that a link within its itinerary

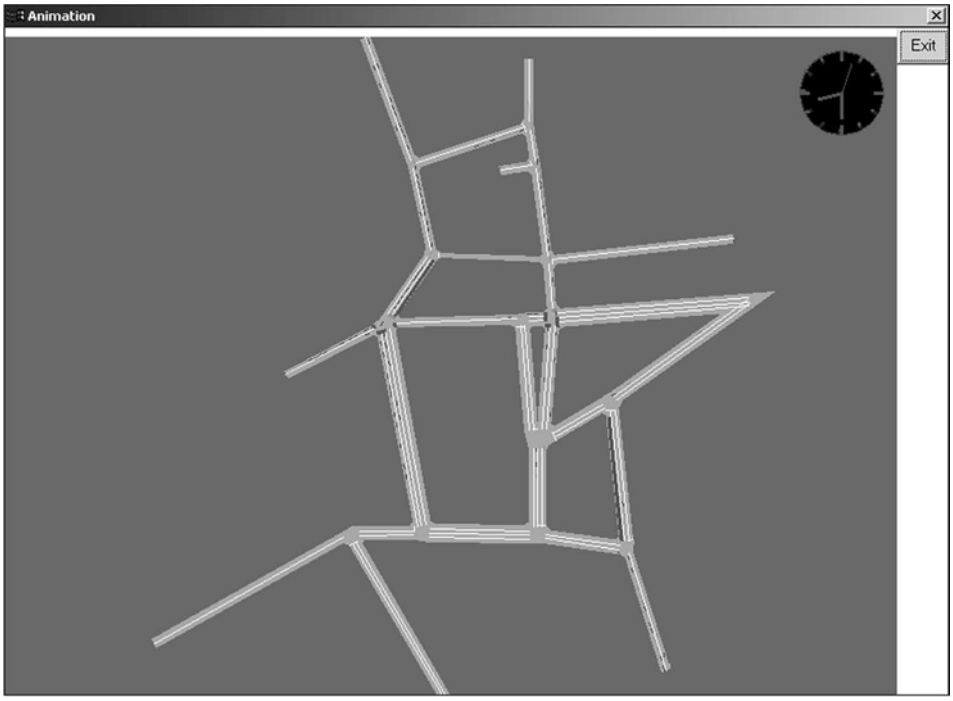


FIGURE 4. A snapshot of the network in the simulation environment.

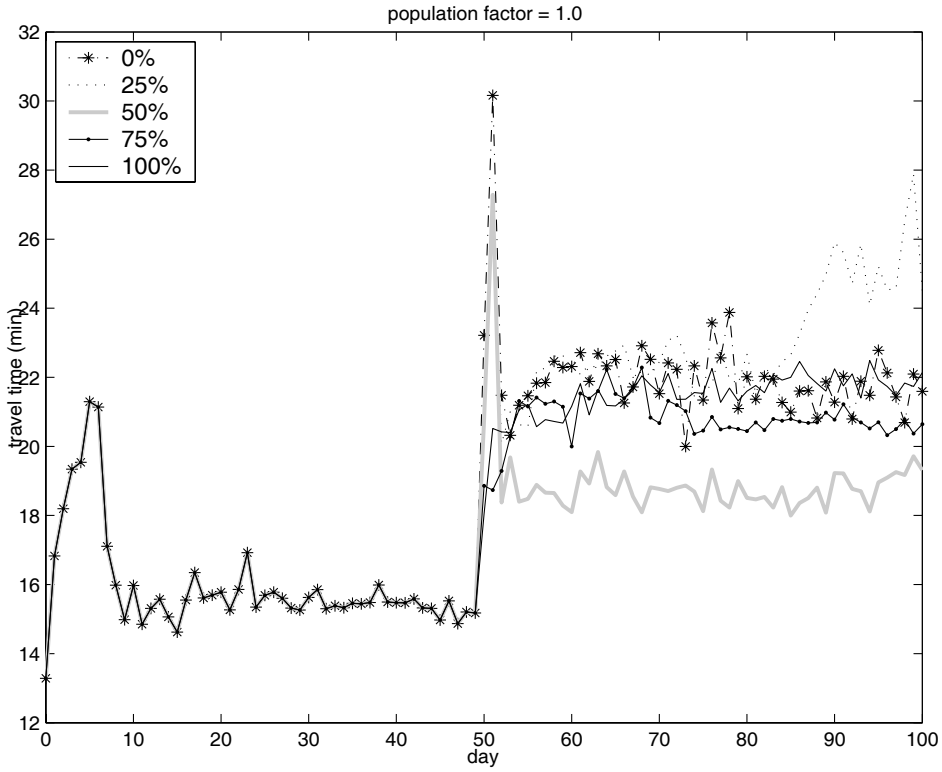


FIGURE 5. Average travel times for drivers travelling through the observed OD pair.

is probably congested, the driver agent tries to select the best alternative path among those that do not contain the link affected. If it is not possible to avoid the congested link, the driver keeps its original route choice.

TABLE 1. (%) fraction of informed users in population, (μ) average travel time, (σ) standard deviation over the 100 days simulated.

%	μ	σ
0	19.02	$\pm$3.379
25	19.54	$\pm$3.920
50	17.46	$\pm$2.055
75	18.40	$\pm$2.697
100	18.73	$\pm$3.007

A major OD movement was selected for analysis (for trips made from zone 109 to zone 105, Figure 3). For this OD pair, three possible routes were available one of which containing a link affected by an incident. The simulation results are presented in Figure 5, which shows the values of average travel time observed over the total number of days simulated (see Table 1 for μ and σ values, as well). The average travel times for trips in the specific OD pair experience increases after day 50, for all fractions of informed drivers. There is a tendency for the average travel time to settle down in different levels after the introduction of the incidents onto the network. The best level is reached when 50% of the drivers are informed about the incident prior to starting their journeys. The results also show that the worst levels are yielded when none or only few (25%) of the drivers have access to the information about the incidents. We have also noticed by observing other OD pairs of the network that their travel times have all been affected, to a greater or lesser extent, even though some routes are not directly affected by the incidents. This is possible due to traffic interaction at intersections.

4. Closing Remarks and Future Work

To improve microscopic simulation of urban networks with multi-agent systems is a goal we have being pursuing in our research. We have proposed a methodological approach to represent contemporary traffic systems by means of a society of agents that cohabitate in a common environment. In such an approach, autonomous decision entities identified in real world can be implemented within an agent shell. The flexible structure conceptualised for the agent shell allows for the use of different reasoning mechanisms, knowledge representation and learning abilities so that different agents can be implemented and inserted into the environment. We profit from the social ability of multi-agent systems to observe the overall system behaviour that emerges from interaction among multiple entities. To test this approach, we have implemented the driver agents with both reactive and cognitive representations. The basic framework of the DRACULA suite was extended to support such an approach and some experiments were carried out. The ability to generate options dynamically, while perceiving the world and executing the reasoning process can be considered a cutting edge between multi-agent traffic simulation and traditional microscopic models, whose options are defined beforehand.

The framework is undergoing further research and development, including the reengineering of the DRACULA suite to fully support agent-based modelling and simulation and the design of an interactive and friendly API to develop different agents, based on the meta-level structure of the agent shell. Our attention is also focused on devising a methodological approach for validating and calibrating multi-agent traffic models.

References

[1] J. L. Adler and V. J. Blue. A cooperative multi-agent management and route-guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5–6):433–454, 2002.

[2] K. Chatterjee and M. McDonald. Modelling the impacts of transport telematics: current limitations and future developments. *Transport Reviews*, 19(1):57–80, 1999.

[3] H. Dia. An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research Part C: Emerging Technologies*, 10(5–6):331–349, 2002.

[4] J. Hernández, S. Ossowski, and A. Serrano. Multiagent architectures for intelligent traffic management systems. *Transportation Research Part C: Emerging Technologies*, 10(5–6):473–503, 2002.

[5] M. J. Huber. JAM: a BDI-theoretic mobile agent architecture. In *Proceedings of the 3rd International Conference on Autonomous Agents*, pages 236–243, Seattle, Washington, May 1999. New York: ACM Press.

[6] R. Liu. The DRACULA microscopic traffic simulation model. In R. Kitamura, M. Kuwahara, and M. Schreckenberg, editors, *Transport Simulation*. Springer, 2003. [To appear].

[7] R. Liu, D. Van Vliet, and D. P. Watling. DRACULA: dynamic route assignment combining user learning and microsimulation. In *Proceedings of the 23rd PTRC European Transport Forum, PTRC*, volume E, pages 143–152, Coventry, UK, 1995. London: PTRC.

[8] H. Mahmassani, S. Hatcher, and C. Caplice. Daily variation of trip chaining, scheduling, and path selection behaviour of work commuters. In *Proceedings of the 6th International Conference on Travel Behaviour*, Quebec City, 1991.

[9] H. S. Mahmassani and R. Jayakrishnan. System performance and user response under real-time information in a congested traffi corridor. *Transportation Research Part A: Policy and Practice*, 25(5):293–307, 1991.

[10] K. Nagel and F. Marchal. Computational methods for multi-agent simulations of travel behaviour. In *10th IATBR Conference*, Lucerne, Switzerland, 2002.

[11] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW*, volume 1038 of *Lecture Notes in Computer Science*, pages 42–55, Eindhoven, The Netherlands, 1996. Berlin: Springer.

[12] M. Rickert and K. Nagel. Experiences with a simplified microsimulation for the Dallas/Fort Worth area. *Internation Journal of Modern Physics C*, 8(310):483–503, 1997.

[13] R. J. F. Rossetti, R. H. Bordini, A. L. C. Bazzan, S. Bampi, R. Liu, and D. Van Vliet. Using BDI agents to improve driver modelling in a commuter scenario. *Transportation Research Part C: Emerging Technologies*, 10(5–6):373–398, 2002.

[14] R. J. F. Rossetti, R. Liu, H. B. B. Cybis, and S. Bampi. A multi-agent demand model. In *Proceedings of the 13th Mini-Euro Conference and The 9th Meeting of the Euro Working Group Transportation*, pages 193–198, Bari, Italy, June 10–13 2002. Bari: Polytechnic University of Bari.

[15] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[16] SACTRA. *Standing Advisory Committee on Trunk Road Assessment (SACTRA), Trunk Roads and the Generation of Traffic*. HMSO, London, 1994.

[17] R. Schleiffer. Intelligent agents in traffic and transportation. *Transportation Research Part C: Emerging Technologies*, 10(5–6):325–329, 2002.

[18] L. Steels. Cooperating between distributed agents through self-organisation. In Y. Demazeau and J. P. Muller, editors, *Decentralized A.I.*, pages 175–196. North-Holland, Amsterdam, 1990.

[19] J. Wahle, A. Bazzan, F. Klügl, and M. Schreckenberg. The impact of real-time information in a two-route scenario using agent-based simulation. *Transportation Research Part C: Emerging Technologies*, 10(5–6):399–417, 2002.

[20] D. P. Watling. Urban traffic network models and dynamic driver information systems. *Transport Reviews*, 14(3):219–246, 1994.

Rosaldo J. F. Rossetti
Gestão de Sistemas e Tecnologias de Informação,
Universidade Atlântica,
Rua dos Paióis, S/N,
2745-615 Barcarena, Portugal
e-mail: `rrossetti@uatla.pt`

Ronghui Liu
Institute for Transport Studies,
The University of Leeds,
36–38 University Road,
LS2 9JT Leeds, United Kingdom
e-mail: `rliu@its.leeds.ac.uk`

A Message-Based Framework for Real-World Mobility Simulations

Christian Gloor, Duncan Cavens and Kai Nagel

Abstract. There is considerable interest in the simulation of systems where humans move around, for example for traffic or pedestrian simulations. Any such simulation consists of two layers: the simulation of the "physical" system, which includes effects such as interaction with other agents or the environment; and the simulation of the "mental" layer, which generates strategies of the agents. The traditional way to couple the modules is to use files. The disadvantage of that approach is twofold: The computational performance is limited by I/O; and the modules can only be run sequentially.

In order to overcome these problems without sacrificing modularity, a message-based approach is presented. Agent strategies are sent via messages to the simulation of the physical system, which executes them and sends back performance information in the form of "events". The strategic modules listen to these events, memorize them in some appropriate way, and possibly generate revised strategies. These strategies are sent to the simulation of the physical system immediately, so that the representation of the agent in the physical system will switch to the new strategy right away.

In addition, the same messages can also be used to plug helper modules, such as viewers or recorders, into the system. An implementation of the framework is tested within our project, which explores the feasibility of using autonomous agent modeling to evaluate future scenarios in a tourist landscape in the Swiss Alps.

1. Introduction

As many planning problems focus on processes that evolve over time in a complex environment, it is often difficult to evaluate the long term implications of a planning decision. Computer simulations can be used as a method for evaluating proposed future scenarios in planning [18]. However, most simulation efforts in spatial planning have focused on large spatial scales (such as at the city and regional levels) and on relatively abstract concepts (such as land use patterns, traffic

and economic development), while one can argue that the planning decisions that have the most impact on individual citizens tend to be either at a relatively small scale or have very local impacts.

Multi-agent simulations, where each agent is modeled individually, allow to look at this problem from a point of view of a person walking in an area. We developed a large scale pedestrian simulation which is intended for the simulation of hikers in the Alps [1, 8, 9], but which should also be applicable to related problems such as urban park design, building design, evacuation simulation [6], department store design, etc. The same computational techniques will be applicable for any kind of distributed multi-agent mobility simulation, in particular for multi-agent traffic simulations.

Any such simulation system does not just consist of the mobility simulation itself, but also of modules that compute higher level strategies of the agents. For traffic and for hiking simulations, the most typical of these modules are: (i) demand generation; (ii) route generation. The demand generation module generates demand for movement between different locations (trips); the routing module computes the actual paths that these trips will follow.

A major and very important difference to the traditional approach is that it is now possible to make all the modules completely microscopic on the level of the hikers. Microscopic means that in all modules each individual agent retains its identity, including, for example, gender, age, income, physical fitness, or remaining energy level. It is, we hope, easy to see how such information can be used for better modeling.

Traditional implementations of transportation planning software, even when microscopic, are monolithic software packages [15, 3, 13, 16]. By this we do not dispute that these packages may use advanced modular software engineering techniques; we are rather referring to the user view, which is that one has to start one executable on one CPU and then all functionality is available from there. The disadvantage of that approach is twofold: All the different modules add up in terms of memory and CPU consumption, limiting the size of the problem. And second, although the approach is helpful when starting as one software project, it is not amenable to the coupling of different software modules, developed by different teams on possible/different operating systems.

A first step to overcome these problems is to make all modules completely standalone, and to couple them via files. Such an approach is for example used by TRANSIMS [19]. The two disadvantages of that approach are: (1) The computational performance is severely limited by the file I/O performance. (2) Modules typically need to be run sequentially. Each module needs to be run until completion before starting the next module. For example, the routing module can only be run before or after the mobility simulation. This implies that agents cannot change their routes while the mobility simulation is running.

Another possibility is to use a database system for information exchange between modules. This is the easiest to imagine if modules still run sequentially. Then each module changes the state of agents in the database, and some central

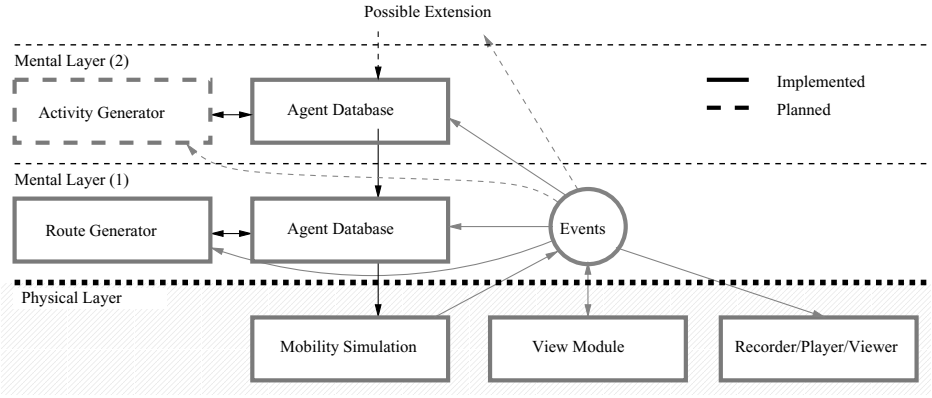


FIGURE 1. Any simulation system does not just consist of the pedestrian simulation itself (physical layer), but also of modules that compute higher level strategies of the agents (mental or strategic layers).

schedule decides which module to run at which time. This approach is, for example, used by URBANSIM [20].

The approach presented in this paper is to couple the modules by messages (Fig. 1). In this way, each module can run on a different computer using different CPU and memory resources, which overcomes the memory bottleneck of the monolithic approach. The approach also avoids the bottleneck of file I/O, since data is not written to file at all while the simulation is running. Finally, the message-based approach allows real-time interaction between the modules: for example, if an agent is blocked in congestion, the strategy generation modules can react to this new situation and submit, say, new routes or activities while the agent is still en-route.

On simulations with tens of millions of agents, issues such as bandwidth usage, packet loss, and latency become increasingly important. As a result, we use different network protocols and implementations tailored to specific requirements of inter-module communication. This paper will also discuss some of these protocols, and the diverse purposes they serve in a distributed multi-agent simulation.

2. The Framework

As said above, our approach is to model each tourist individually as an "agent". A synthetic population of tourists is created that reflects current (and/or projected) visitor demographics. These tourists are given goals and expectations that reflect existing literature, on-site studies, and, in some cases where sufficient data is not available, are based on experts' estimates. These expectations are individual,

meaning that each agent could potentially be given different goals and expectations.

These agents are introduced into the simulation with initial plans (see later), but with no "knowledge" of the environment. The agents execute these plans, receiving feedback from the environment as they move throughout the landscape. At the end of each run, the agents' actions are compared to their expectations. If the results of a particular plan do not meet their expectations, on subsequent runs the agents try different alternatives, learning both from their own direct experience, and, depending on the learning model used, from the experiences of other agents in the system.

After numerous runs, the goal is to have a system that, in the case of a status quo scenario, reflects observed patterns in the real world. In this case, this could, for example, be the observed distribution of hikers across the study site over time.

A "plan" can refer to an arbitrary period, such as a day or a complete vacation period. As a first approximation, a plan is a completely specified "control program" for the agent. It is, however, also possible to change parts of the plan during the run, or to have incomplete plans, which are completed as the system goes.

2.1. The physical layer (mobility simulation)

In our architecture, agents' plans are submitted to the mobility simulation. The mobility simulation executes all plans simultaneously, computing interactions of the agents with the environment and with each other. For example, if two agents want, according to their plans, to be at the same place at the same time, the physical interaction of the simulation will prevent that and compute physically plausible solutions instead.

Information about each agent's performance is sent back to other modules in the form of events. Examples of events are "agent left hotel", "agent entered link", "agent had nice view", etc. Events come together with a time stamp and the agent id number.

Many modeling techniques exist for the simulation of pedestrian movements. For our simulations, we need to maintain individual particles, since they need to be able to make individual decisions, such as route choices, throughout the simulation. This immediately rules out field-based methods, where particles are aggregated into fields. We also need a realistic representation of inter-pedestrian interactions, which rules out mesoscopic models, such as queue models or smooth particle hydrodynamics models.

For microscopic simulations, there are essentially two techniques: methods based on coupled differential equations, and cellular automata (CA) models. In our situation, it is important that agents can move in arbitrary directions without artifacts caused by the modeling technique, which essentially rules out CA techniques. We therefore use a coupled differential equation model for pedestrian movement (social force model [10]).

An important functionality of our simulation is to evaluate the visual quality of the landscape. It turns out that this can be done by using the 3d visualizer

that is also used for humans to interact with the computer system (see below). However, instead of displaying a view on the screen after it is computed, the video memory is read out in order to determine what individual agents "see" as they move through the landscape. The agents' field-of-view is analyzed, and events now contain information describing what the agent sees. Depending on the needs of the brain modules, these events either list all of the individual objects (houses, restaurants, forest stands, or individual trees) or return synthesized information about particular visual metrics (such as enclosure, percentage of view that is non-vegetative, etc.)

2.2. Learning

Initially, every agent starts with a plan that, in its opinion, fulfills its expectations. For example, if the period of interest is a day, then such an initial plan might refer to a specific hike. To do this, the agent chooses *activity locations* it wants to visit, like hotel, peak of mountain, restaurant etc. The chain of activity locations is then handed over to the *routing* module, which calculates the routes between activities according to the information available. This information can be static and global, like shortest path information based on the street network graph.

The mobility simulation then executes the routes. The agent experiences the environment and sends its perception as events to the other modules.

From here on, the system enters the *replanning* or *learning loop*. The aforementioned idea is that the agents go through the same period (e.g. day) over and over again. During these iterations, they accumulate additional information, and try to improve their plan.

The two critical questions are (1) how to accumulate, store, and classify that information, and (2) how to come up with new plans. Both questions are related to (artificial) intelligence, and we are certainly far away from answering them in their entirety. Nevertheless, our system contains the following elements which makes it able to learn.

As one can see in Fig. 1, there are **agent databases** associated with each level of the planning hierarchy (e.g. activities, routes). The task of these agent databases is to store plans and to accord scores to them. That is, every time an agent comes up with a new plan, that plan is added to the repertoire of plans. In addition, the agent databases listen to the events emitted by the mobility simulation, and use these events to calculate a score for each plan once it has completed. If an agent database module assigns a bad score to a simulated hike, it tries to avoid its elements in future hikes. If it gets good feedback, however, it will try to reuse the elements of a hike, and combine these into a new hike, which will be simulated and scored again.

The agent databases, at each level of the hierarchy, need to obtain new plans from time to time. Such plans can be constructed by the following methods:

- New plans can be constructed from global information.
- New plans can be constructed from information that an individual agent has accumulated.

- New plans can be constructed by randomly modifying an existing plan. In the language of genetic algorithms, this is called (generalized) **mutation**.
- New plans can be constructed by taking two existing plans and combining elements of them. In the language of genetic algorithms, this is called (generalized) **crossover**.

An example of a method that generates elements of plans from scratch based on global information is a **shortest (fastest, best) path algorithm**. This algorithm takes starting and ending location as input, and computes the best path connecting those two as output. An example of such a method is a Dijkstra algorithm [4]. It is possible to use generalized cost functions in such a router, which makes it possible to find different solutions for different types of agents. For example, a physically fit person might prefer a route that is steeper, while a physically less fit person might get a route that includes more possibilities to rest. It is also possible for the router to listen to the streams of events, for example finding out how many agents are on which links as a function of the time-of-day. This makes it possible to include congestion effects.

An example of a method that generates elements of plans from individual agent information is a module based on a mental map (e.g. [2]). This module listens to the stream of events. However, rather than scoring complete plans as the agent database does, it constructs a mental map of the spatial environment. From the times when agents enter and leave links, the mental map learns how long it takes for an agent to walk along each link in the network. Also, the mental map accumulates all the other events that occur on every link. Using these values for each link in the network, a router based on this mental map is able to return a route for each agent, based on its expectation and demographic data. A prototype implementation of such an approach can be found in Ref. [11].

It is possible to make the mental layer module dynamic. In this case, it observes the agent on its path through the virtual environment. As soon as it detects an option that might yield a better score than the current plan, e.g. using a better path, or entering a restaurant, it notifies the agent in the simulation. Using this mechanism, agents are able to react to unpredicted changes in the environment, like weather changes or congestion.

It should be noted that the distinctions between these modules are not sharp. For example, an agent database may run out of memory if it memorizes as separate entities plans that differ only in small details; in that case, the agent database might have to start building a mental map of the world. In this case, it becomes similar to the mental map module as described above.

Fig. 2 shows an example of agent learning. Both pictures show a snapshot of the same region at the same time of day. All agents leave the hotel (A) in the morning, and hike to the mountain peak (B). On the first run (top), when the hikers do not know about the other hikers' intentions, all hikers take the same path, which they consider the best. After 50 iterations, agents have learned to spread out in order to somewhat avoid each other. – Clearly, this depends on

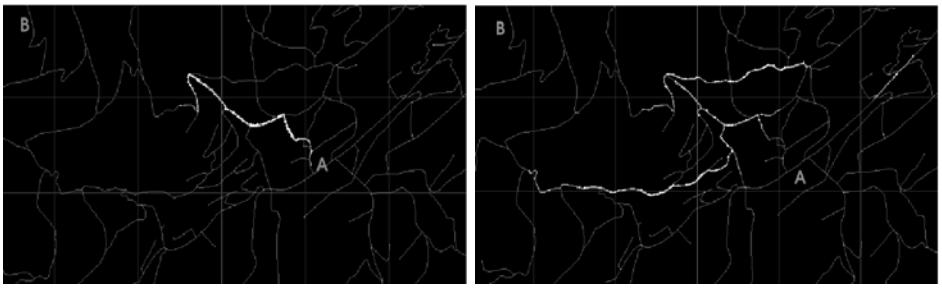


FIGURE 2. Agents leave the hotel (A) in the morning and hike to the mountain peak (B). On the first run, where no hiker knew about the others' intentions (top), and the situation after 50 iterations.

how much hikers really want to be alone, which is a parameter that needs to be estimated from real-world surveys. The simulation can then show the results of changes in that parameter, or it can show how much the satisfaction of hikers with respect to that parameter can change when the environment is changed.

2.3. Initial conditions

The learning process needs to be started somehow. For this, an initial population needs to be generated, and they need to compute initial activities and routes. For the generation of the initial population, a **synthetic population generation** module is used. This could use aggregated information, such as known age or gender distribution of tourists, and generate individual agents from this, which have individual age, gender, income, etc. At this point, the implemented module generates agents with random attributes (varying walking speed, different weights for the generalized cost function in the router).

Initial activity plans are constructed from some global knowledge of the area (i.e. agents are told where attractive view points or where restaurants are); initial routes are constructed based on geometrical distance.

2.4. Helper Modules

For computing a solution, the following modules are not needed. However, a simulation system without would be useless:

Recorder/Player. This module receives and processes agents positions and other data from the simulation. It can store them in a file and/or forward them to any viewers which are currently connected to it.

There are several reasons that the simulation does not directly write to log files itself:

- The simulation can be parallelized, and we need the log output of all the instances in a single file.
- Writing to a remote file system (i.e. NFS) can be very slow.

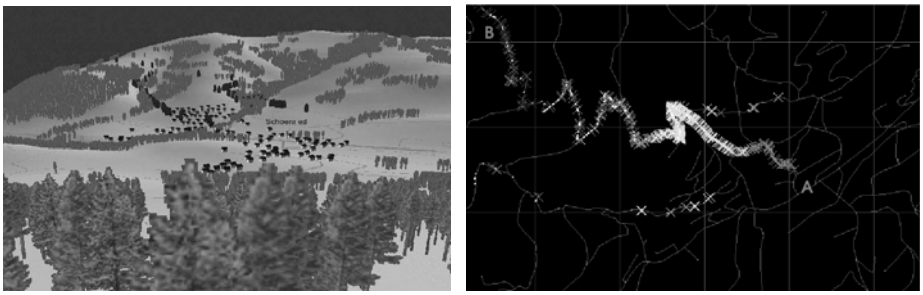


FIGURE 3. A 3-dimensional (left) and a 2-dimensional (right) viewer have been implemented. Both pictures show exaclty the same scenario of agents hiking from a hotel (A) to the peak of a mountain (B). While the 3-dimensional viewer provides a general overview, the 2-dimensional viewer displays specific events, which contain the agents' perception and performance information, in different colors.

- There is a single interface for viewers.
- By splitting the logging functionality from the simulation, we are able to change implementation details for all of the simulation modules without needing to modify the logging code. For example, one could use a database to log the events instead of a file.

In order to visualize simulation runs, we have developed **Viewer Modules**, which are stand-alone applications that connect to the simulation system via the network. Viewers are built so that they directly plug into the live system. The simulation sends agents' positions to the viewer, which allows to look at the scenario from a bird's eye view and observe how the agents move. It is also possible to send that same data stream to a recorder which records it to file, while a player can read the file and send the data stream to the viewer exactly the same way it would come from the simulation directly. There can be any number of viewers connected to a player module, each showing the same scenario from a different perspective, or displaying different accumulations of events. Finally, in order to deal with data conversion issues, it is also possible to pipe the data stream from the simulation through the recorder directly to the player and from there to the viewer.

We have implemented two different viewers. One displays a 2D view, and is suited for situations where a lot of detailed information is needed, for example while debugging (Fig. 2 and Fig. 3, right side). Also, a 3D viewer has been implemented (Fig. 3, left), as one of our overall project goals is to integrate decisions based on visual stimuli. The 3D viewer connects to the simulation using the same protocol as the 2D viewer. The user can move independently of the agents or can attach the camera viewpoint to a specific agent and see the landscape through the eyes of the agent.

3. Technical Implementation

As described above, the modules need to communicate with each other. The mobility simulation receives plans and outputs events. The mental modules receive events, and output and receive plans. The helper modules normally deal with events only.

3.1. MPI/PVM

When a *single module is distributed* across multiple computational nodes, one often uses MPI (Message Passing Interface [12]) or PVM (Parallel Virtual Machine [14]). It is also possible to use MPI or PVM for the communication between *different modules* as described in this paper. That approach has, however, the disadvantage that one is bound to the relatively inflexible options that MPI offers. For example, options to add or remove modules during runtime have only recently been added to the MPI standard, and multicast (see below) is not possible at all.

3.2. TCP

On clusters of workstations, MPI is often implemented on top of TCP (Transmission Control Protocol). TCP is the connection based, reliable protocol of the TCP/IP suite. Initially, a connection from the sender to the receiver must be opened. With this connection, both sides can send their messages as the connection is symmetric. TCP guarantees that the messages arrive, in correct order, and without errors.

3.3. UDP

UDP offers, in comparison to TCP, no control for packet loss. This means that there is no guarantee that the sent packets will arrive. The advantage is that there is considerably less overhead. Also, as will be described later, this offers (via multicast) the option to send events only once, even when many modules listen to them. In a TCP-based implementation, events need to be sent to each listener separately. A message that arrives is guaranteed to be error free, since the UDP protocol includes a checksum.

We use UDP to transmit the agent positions to the visualizers. If the network is down for a few seconds, the simulation does not need to slow down because of lost packets. Once the viewer is back on line, it will receive the latest positions.

There are other situations in which one may accept the loss of messages. For example, if an agent reports that it is blocked in unexpected congestion (e.g. waiting for a cablecar, traffic jam), it needs a new route instantly. If its request is lost or delayed, it makes no sense for the system to buffer its request, since the agent has moved on, and the location in the original request might now be invalid. A new route computed based on the old information will be invalid as well. It is the agent's responsibility to restate its position again if it does not receive a new route after a certain time has elapsed [7].

The amount of packet loss is strongly dependent on the overall number of packets in the network. In state-of-the-art networks, which today are often 1 Gbit

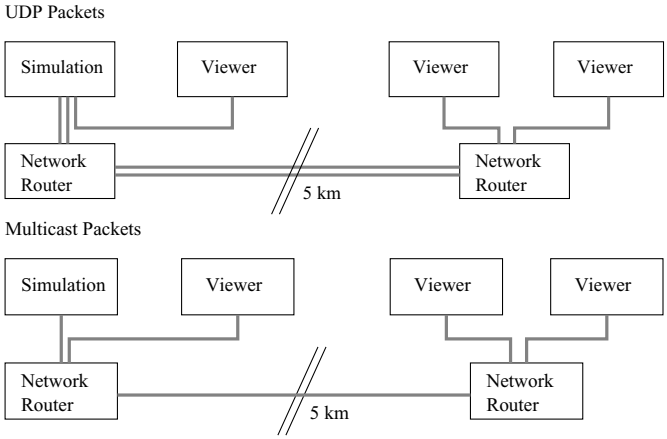


FIGURE 4. Multicast uses one data connection, even if there are multiple receivers subscribed to the multicast channel.

Ethernet, there is hardly any packet loss in the network itself. Losses occur mainly in the sending and receiving network interface cards (NICs), due to overflowing buffers. This is the case, for example, if the CPU is busy so that it cannot read the packets from the buffer quickly enough. The more packets that are sent, the higher the chance that one is lost. With Gbit Ethernet communication, up to 180'000 raw data packets can be sent per second without any losses. Using a naive approach, which is packing one event into one network packet, one obtains 180'000 events per wall-clock second. Since the mobility simulation runs more than 100 times faster than wall-clock time, this results in 1'800 events per simulated second. However, this is not much for a simulation of 1000 or more agents.

Recently, new networking infrastructures have been developed which allow to send packets reliably with UDP. An example is the TNet Hardware [17], used in computer clusters. The TNet hardware assigns a 16 bit CRC (cyclic redundancy check) to each packet. After every transmission over a network link the packet is checked for correctness, and retransmitted if an error is detected.

3.4. Multicasting

Often there is a need for sending the same packet to more than one receiver. This can be achieved by opening multiple TCP connections or by sending multiple UDP packets to the receivers. However, on large simulations, the network interface card (NIC) of the sending host quickly becomes the bottleneck, as it is unable to send out enough packets to keep the receivers fully occupied.

On Internetworks, it is possible to use *multicasting* to send a single message from one computer to several other computers.Because multiple machines can receive the same packet, bandwidth is conserved. Multicasting is particularly useful

for any kind of streaming data such as radio or television broadcasts over the network. Its advantage is that the multiplication of the packets for multiple receivers is not done by the NIC, but by the network itself. This allows to avoid the NIC bottleneck.

Multicasting provides groups of hosts, that are referenced using special IP addresses (224.0.0.0 – 239.255.255.255). The sender chooses one of these groups and sends a single packet to this IP address. A receiver must explicitly join a group first, telling its NIC and the operating system to listen for packets sent to this group. An advantage of this addressing scheme is that the sender does not need to know the IP address of the receiver. This simplifies the configuration of the system substantially. The Internet routers ensure that the packets find their way from the sender to the receivers, once they are registered to the multicast group.

A drawback with multicasting is that it has, similar to UDP, no arrival control. There is no feedback to the sender if all packets arrived at all destinations.However, as described before, this is not a problem for data sent to our viewers.

Our project is a collaboration between two institutes at ETH Zürich. One of them is located more than 5 km away from where our computational cluster is. For every viewer that is connected to the simulation, extra bandwidth is needed. Using multicast, we cannot reduce the bandwidth used for one viewer. But as soon as there are multiple viewers looking at the same general area, the bandwidth remains constant (Fig. 4). Sending agent data to multiple viewers is an instance where multicasting is extremely effective. Using multicasting, we were able to allow multiple viewers at the remote institute without saturating the NIC.

3.5. XML: Extensible ASCII Messages

Viewers are built so that they directly plug into the live system. The simulation sends agents' positions to the viewers, which allows to look at the scenario from a bird's eye view and observe how the agents move. As mentioned before, at this point there are two different viewers, one in 2D and mostly intended for debugging, and one in 3D.

For the 3D viewer, more data has to be sent, since it needs also the *altitude* of an agent. One needs the ability to *add* this value to the data stream in a way that there is no need to change existing viewers.

The Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

The main advantage of XML is that there is no need to enforce a mandatory file format between the sender and the receiver.

Tags (e.g. `event`) and attributes (e.g. `agent="42"`) are not defined in any XML standard *per se*. These tags are introduced by the author of the XML document. This makes XML a perfect choice as a format for data exchange among

different modules of a simulation. Whenever a module introduces a new kind of message, there is no need for a change in any of the modules that listens to the message stream—unless of course, a module wants to handle this new information specifically. This is also the case for a attribute introduced additionally. It is, however, not possible to *change* the name of existing tags or attributes.

A typical *position update message* used in our simulation system encoded in a XML fragment looks like this:

```
<event type="position" agent="42"
x="588440.1" y="150281.4" />
```

Frequent changes in messages, something that happens often in research and development, are possible. Since the receiving modules search for "keyword=value" pairs, an additional attribute is simply ignored. As a result, there is no need to adapt existing modules. Further, due to the fact that messages are transmitted in plain text, debugging of communication is possible without further tools. The example message is perfectly understandable by all modules if another attribute is added:

```
<event type="position" time="23"
agent="42" x="588440.1" y="150281.4" />
```

Since it is possible to attach existing XML parsers to any I/O stream or buffer, there is no difference between reading messages from a file or receive messages over the network. We tested XML over UDP and TCP, both versions are very flexible. Using UDP, however, if a packet containing a piece of a message is lost during transmission, the resulting stream is not necessarily still a valid XML document. This problem is circumvented by always sending complete messages in a packet. Recall that packets are guaranteed to be error free. UDP packets arriving with an error are discarded; TCP packets arriving with an error are re-transmitted.

Problems with XML are that (i) searching for "keyword=value" pairs is slow, (ii) if part of a XML stream is lost, the whole XML context may become invalid, and (iii) XML is plain text, so we have to convert binary numbers into ASCII characters and back. To overcome problems (i) and (ii), we have developed a parser that is specialized in parsing a certain subset of the XML standard. This subset consists of simple tags with no nested tags inside:

```
<tag {attr="value"} />
```

Note that all information is inside the attributes, and therefore inside the tag as well. Therefore no nesting of tags is possible. This subset parser is written as a substitute for existing XML parsers, such as Expat [5]. Therefore, the programmer's interface is exactly identical.

We measured the performance of our subset parser by parsing messages of different size for one second. These measurements were done on a 700MHz Pentium III. The messages were taken directly out of the machines main memory, so no network or disk access was involved. The results are that the off-the-shelf Expat parser parses 110'000 messages per second, while our specialized XML subset

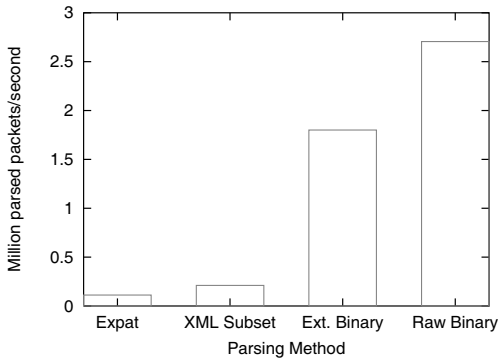


FIGURE 5. Comparison of parsing speed (left): Expat, XML Subset Parser, Extensible Binary Protocol vs. raw binary messages.

parser parses 210'000 messages per second (see Fig. 5). Thus, our XML subset parser is faster by a factor of 2.

3.6. Extensible Binary Protocol

The flexibility of XML is achieved at the cost of parsing ASCII files for tags and well-formatted entries. In multi-agent simulations, millions of individual agents are simulated and have to report to external applications such as graphical viewers or storage managers.

Our experiments have shown that the exchange of multi-agent data in the form of XML streams is very flexible but can reduce the bandwidth substantially due to the overhead of parsing the streams and due to converting numbers represented binary internally into ASCII and back.

In terms of CPU time, parsing the same type of streams data all the time is a waste. Often the structure of a message is known once the simulation is running.

A potential strategy is: initially, the consumer module connects to the producer module and receives a description of the available data. Then the consumer specifies which part of the data and which formatting it needs. Based on that specification, the producer begins to output binary streams formatted according to the consumer specification.

For instance, if agents are agents in a cablecar, a 2D viewer will only require x/y coordinates, while a 3D viewer will require x/y/z positions. The whole process can be seen as a user-defined on-the-fly serialization of the producer's objects (i.e. the agents) and their transfer over a high bandwidth binary channel.

We have implemented a protocol that follows these steps:

- A receiving module asks for certain data values to be transmitted in data messages. This is done by specifying the XML tag names, e.g. "agent" or "time". A example request, as sent over the network, looks like `<request type= "event" time="" agent="" x="" y="" />`.

- The sending module transmits a description of future data messages, e.g. `<description type="event" version="1" time="i" agent="i" x="d" y="d" >`. The field `version` is the sequence number of the description. It is unique for each new description sent.
- The sending module from then on sends plain data, packed according to the description into a binary buffer.

As soon as another receiving module asks for additional data values, they are included into the message as well, after a new description is sent to all the receiving modules. A description is sent every second as well, in case a receiving module lost a description or joined the system without requesting a new message format.

Our measurements have shown that this strategy yields in a factor of 17 faster than by using Expat, and in a factor of 8 faster than by using our XML subset parser (Figure 5, 3rd bar).

3.7. Sending raw binary data

For comparison, also the exchange of raw, binary messages was measured. It communicates the same amount of information as the above, but does nothing with it. This results in the 4th bar in Fig. 5. As one can see, the maximum possible speed as measured by this method is only about a third faster than our extensible binary protocol.

3.8. Conclusions

Using the Extensible Binary Protocol, it was possible to visualize scenarios containing more than 1000 agents running more than 100 times faster than real time. An agent number and the x and y co-ordinates of a position update message consume 20 bytes, when packed binary. It is therefore possible to transmit up to 75 agent positions per packet. If the viewer is able to receive 200'000 packets per second, we are able to display 15 million agents per second. Since the viewer uses its host's CPU cycles for drawing the graphical output as well, the actual number drops again substantially.

4. Discussion and Outlook

It is important to note that the task of the mobility simulation is simply to send out events about what happens; all interpretation is left to the mental modules. In contrast to most other simulations in the area of mobility research, the simulation itself does not perform any kind of data aggregation. For example, link travel times are not aggregated into time bins, but instead link entry and link exit events are communicated every time they happen. If some external module, e.g. the router, wants to construct aggregated link travel times from this information, it is up to that module to perform the necessary aggregation. Other modules, however, may need different information, for example specific progress reports for individual agents, which they can extract from the same stream of events. This would no

longer be possible if the simulation had aggregated the link entry/exit information into link travel times.

Despite this clean separation – the mobility simulation and the modules in the physical layer compute "events", all interpretation is left to mental modules – there are conceptual and computational limits to this approach. For example, reporting everything that an agent sees in every given time step would be computationally too slow to be useful. In consequence, some filtering has to take place "at the source" (i.e. in the simulation), which corresponds to some kind of preprocessing similar to what real people's brains do. This is once more related to human intelligence, which is not well understood. However, also once more it is possible to pragmatically make progress. For example, it is possible to report only a random fraction of the objects that the agent "sees". Calibration and validation of these approaches will be interesting future projects.

5. Summary

This paper reports basic elements of a distributed mobility simulation system. The simulation system consists of two layers, the physical layer (mobility simulation), and the mental layer (strategy/plans generation). The mental layer generates plans, which are submitted to the mobility simulation for execution. The mobility simulation returns events to the mental layer. The modules of the mental layer use these events in different ways, for example to score plans, to compute best paths, or to construct mental maps.

Since the communication needs between these modules is substantial, several methods for message passing are evaluated. Traditional approaches, such as MPI or PVM, are too inflexible for what was intended for this project. For that reason, specialized protocols based directly on the operating system are evaluated. These protocols have trade-offs in terms of ease-off-use, bandwidth consumption, and potential message loss. The overall result is that, albeit at the expense of having to use a variety of protocols for different purposes, even with existing technology simulations with thousands of agents running hundreds of times faster than real time are possible with our approach.

6. Acknowledgments

We thank Ingo Opperman for his work on the binary XML-like protocol, and Bryan Raney for the learning mechanism which computed the transition in Figure 2.

References

[1] ALPSIM www page. www.sim.inf.ethz.ch/projects/alpsim/, accessed 2004. Planning with Virtual Alpine Landscapes and Autonomous Agents.

[2] T. Arentze and H. Timmermans. Representing mental maps and cognitive learning in micro-simulation models of activity-travel choice dynamics. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, 2003. See www.ivt.baum.ethz.ch.

[3] A. Babin, M. Florian, L. James-Lefebvre, and H. Spiess. EMME/2: Interactive graphic method for road and transit planning. *Transportation Research Record*, 866:1–9, 1982.

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

[5] Expat www page. James Clark's Expat XML parser library. expat.sourceforge.net, accessed 2004.

[6] E. R. Galea, editor. *Pedestrian and Evacuation Dynamics 2003*. Proceedings of the 2nd international conference. CMS Press, University of Greenwich, 2003.

[7] C. Gloor. Modelling of autonomous agents in a realistic road network (in German). Diplomarbeit, Swiss Federal Institute of Technology ETH, Zürich, Switzerland, 2001.

[8] C. Gloor, D. Cavens, E. Lange, K. Nagel, and W. Schmid. A pedestrian simulation for very large scale applications. In A. Koch and P. Mandl, editors, *Multi-Agenten-Systeme in der Geographie*, number 23 in Klagenfurter Geographische Schriften, pages 167–188. Institut für Geographie und Regionalforschung der Universität Klagenfurt, 2003.

[9] C. Gloor, L. Mauron, and K. Nagel. A pedestrian simulation for hiking in the Alps. In *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, CH, 2003. See www.strc.ch.

[10] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000.

[11] D. Kistler. Mental maps for mobility simulations of agents. Master's thesis, ETH Zurich, 2004.

[12] MPI www page. www-unix.mcs.anl.gov/mpi/, accessed 2005. MPI: Message Passing Interface.

[13] PTV www page. Planung Transport Verkehr. See www.ptv.de, accessed 2004.

[14] PVM www page. www.epm.ornl.gov/pvm/, accessed 2004. PVM: Parallel Virtual Machine.

[15] M. Rickert. *Traffic simulation on distributed memory computers*. PhD thesis, University of Cologne, Cologne, Germany, 1998. See www.zaik.uni-koeln.de/~paper.

[16] P. Salvini and E. Miller. ILUTE: An operational prototype of a comprehensive microsimulation model of urban systems. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, 2003. See www.ivt.baum.ethz.ch.

[17] SCS www page. http://www.scs.ch/references/tnet.html, accessed 2004. The T-Net Hardware.

[18] H. Timmermans. The saga of integrated land use-transport modeling: How many more dreams before we wake up? In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, 2003. See www.ivt.baum.ethz.ch.

[19] TRANSIMS www page. TRansportation ANalysis and SIMulation System. transims.tsasa.lanl.gov, accessed 2005. Los Alamos National Laboratory, Los Alamos, NM.

[20] P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson. Microsimulation of urban development and location choices: Design and implementation of UrbanSim. *Networks and Spatial Economics*, 3(1):43–67, 2003.

Christian Gloor
Swiss Federal Institute of Technology
Institute of Computational Science,
CH-8092 Zürich, Switzerland
chgloor@inf.ethz.ch

Duncan Cavens
Swiss Federal Institute of Technology
Institute for Spatial and Landscape Planning
CH-8093 Zürich, Switzerland
cavens@nsl.ethz.ch

Kai Nagel
Technical University Berlin
Transport Systems Planning and Transport Telematics
D-10587 Berlin, Germany
nagel@vsp.tu-berlin.de