Czech Technical University in Prague

Faculty of Electrical Engineering

AGENT-BASED COMPUTING FOR INTELLIGENT TRANSPORT SYSTEMS

Habilitation Thesis

Michal Jakob

October 2014

Preface

Transport and mobility play an essential role in the functioning of modern society – being integral part of our daily lives, employing millions of people and contributing significantly to the economy. Due to their characteristics – comprised of massive numbers of self-interested entities acting and interacting in a shared, resource-constrained environment – transport systems are inherently *multiagent systems*. Formalizing and solving transport and mobility problems using the multiagent systems framework is therefore fitting and useful, as confirmed by the numerous successful applications of agent-based techniques in transport over the last decade.

The natural fit between transport and multiagent systems was a strong motivation also for my own research on applying agent-based and, more generally, artificial intelligence techniques to problems in transport and mobility. My research, conducted together with my colleagues at the Czech Technical University in Prague and with external collaborators, concerns three specific topics: (i) agent-based simulation modelling of transport and mobility, (ii) advanced multimodal journey and route planning and (iii) agent-based resource allocation for on-demand transport services. Although many of my results have wider applicability, most of them have been obtained in the domain of maritime transport or, more recently, in the domain of integrated multimodal urban mobility.

My research contributions to the above topics, generated in the course of approximately five years, have been published in 26 peer-reviewed scientific papers, selected nine of which are included in this thesis. The research I conducted with my collaborators has also resulted in several advanced software systems, some of which have been released as open source software.

The thesis starts with a brief introduction to multiagent systems in transport and mobility, followed by a commented summary of my specific research contributions to the field. The summary of my existing results to date is then complemented with an outlook for future research. The rest of the thesis comprises the reprints of the nine selected scientific papers.

Acknowledgments

I would like to thank to my numerous collaborators who contributed to the work presented in this thesis. In particular, I would like to thank to my Ph.D. students Ondřej Vaněk and Jan Hrnčíř for being such great research partners – independent in their thinking while receptive to advice and very effective in elaborating research ideas into tangible research results. I am further grateful to Michal Pěchouček, the head of the Agent Technology Center, for creating an inspiring and productive research environment. Most of all, I would like to thank to my wife Nora and my sons Max and the very little Leo for their admirable patience and support during the challenging period of finishing this work.

Contents

Chapter 1

Introduction and Research Summary

Transport and mobility play an essential role in the functioning of modern society – being integral part of our daily lives, employing millions of people and contributing significantly to the economy. In Europe[1], the transport industry directly employs more than 10 million people, accounting for 4.5% of total employment, and represents 4.6% of GDP. 13.2% of every household's budget is spent on average on transport goods and services. Logistics, such as transport and storage, account for 10-15% of the cost of a finished product.

Not everything is perfect with our current transport systems, though. Transport has a major negative environmental impact, contributing significantly to greenhouse gas, noise and other pollutant emissions. Despite a steady decline, transport-related accidents are still a major cause of premature death in Europe, with 26,000 road fatalities in 2013 alone[2]. Finally, traffic congestion costs Europe an estimated 100 billion Euro, i.e., the whole 1% of GDP.

There is therefore a strong motivation for further improving the efficiency, safety and sustainability of transport systems. The application of information and communication technology to transport – referred to nowadays broadly as *intelligent transport systems (ITS)* [59] – has long been recognized as a promising way to reach these objectives. The potential of advanced computation and information processing techniques for improving transport operations has been further multiplied by recent technological advances, in particular the advent of ubiquitous internet connectivity, cloud computing, pervasive sensor technology and GPS-enabled smartphones. With large volumes of historic and real-time data available, the transport field has become ready for the application of a wide range of *intelligent* data analysis, control and planning techniques capable of turning collected data into insight and effective action [40]. Increasingly, for the reasons explained below, the application of intelligence computational techniques to transport is taking place within the framework of multiagent systems.

[1] source: `http://ec.europa.eu/transport/strategies/facts-and-figures/transport-matters/index_en.htm`

[2] source: `http://ec.europa.eu/transport/road_safety/specialist/statistics/index_en.htm`

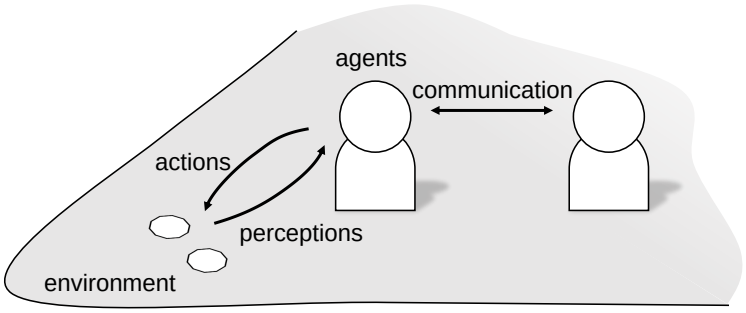


Figure 1.1: High-level conceptual model of the multiagent system.

1.1 Transport and Multiagent Systems

With an acceptable level of simplification, the *multiagent system* [60] can be defined as a system composed of multiple autonomous entities, termed *agents*, situated and interacting within a shared environment. The environment represents the physical space surrounding the agents – agents can modify the state of the environment by performing actions and the agents are informed about the state of the environment and its changes through perception. We assume that the agents are endowed with intelligence that allows them to select and execute such actions that bring them closer to their goals. However, as the environment is one and the agents are many, the actions of individual agents can mutually interact and produce results that, for better or worse, would not be achieved by individual agents alone. Based on the motivations of agents and the nature of interactions, multiagent systems can be classified as *cooperative* (all agents act towards a single shared team goal), *competitive* (agents pursue their individual self-interest) or a combination of both. In addition to implicit interaction through the environment, agents can also interact directly, i.e., bypassing the environment, through message-based communication. See Figure 1.1 for an illustration of a high-level conceptual model of a multiagent system.

1.1.1 Properties of Transport Systems

In transport systems, many autonomous actors, such as passengers, drivers or transport operators, pursue their transport-related objectives within the context of a shared and capacity-constrained transport infrastructure and limited-size vehicle fleets. The individual actors in the transport system interact with each other as well as with the transport infrastructure (e.g., queuing at junctions), and produce emergent, potentially complex global behaviour (e.g., traffic waves).

Transport systems exhibit many of the characteristics typical for multiagent systems:

- **Multiple self-interested actors** – Transport systems involve multiple and often very many actors with different, potentially conflicting goals.
- **Frequent interactions** – Actors in transport systems interact locally by observing their surrounding physical space as well as at a distance using communication technology (the latter is greatly facilitated by now ubiquitous internet connectivity).

- **Limited-capacity resources** – Both the transport infrastructure (e.g., road space, or parking places) and vehicle (e.g., taxis, or train seats) resources have limited capacity. This can give rise to competition between the actors in the transport system.

- **Spatial distribution** – Transport systems are geographically extended, spanning areas from single neighbourhoods and cities up to entire countries and continents. Because of the globalization, even very distant parts of transport systems have become connected and can affect each other.

- **High dynamism** – With many of their actors in motion, transport systems operate under constantly changing conditions.

- **No single central authority** – There is no single central authority which would direct the behaviour of transport systems. Although a common regulatory framework is typically in place, it is not fully enforceable and the self-interested behaviour within the bounds of the regulation is in any case decided freely by the actors in the transport system.

- **Autonomy** – Automation and consequently autonomy have been increasing at all levels of transport systems. Many of traffic and transport control and management decisions are now made with minimum or zero direct human action. Autonomy is posed to continue growing strongly in the future.

Due to their structural and behavioural properties, transport systems therefore essentially *are* multiagent systems. The solutions of transport modelling, control and optimization problems should therefore benefit from the application multiagent system models and techniques.

1.1.2 Research on Agents in Traffic and Transport

The suitability of the multiagent systems framework for conceptualizing and solving mobility and transport-related problems has been reflected in the growing number of publications on agents in traffic and transport. A systematic exploration of agent-based techniques for intelligent transport systems started around the turn of the millennia – the first review [46] of agent-based approaches to transport problems was published in 2002, followed by [5] in 2005. So far the latest review [15], published in 2010, provides a detailed analysis of almost 130 papers, which demonstrates the growing popularity of the agent-based computing in intelligent transport systems.

So far, agent-based approaches in transport have been primarily applied in the following areas:

- **Traffic microsimulation modelling**, including cognitive driver models, lane following and lane changing models, or vehicle-to-vehicle and vehicle-to-infrastructure system models.

- **Vehicle dispatching, routing and scheduling**, including dynamic pickup and delivery problems in on-demand transport and logistics, taxi fleet management, or air traffic take-off/landing slot allocation.

- **Collaborative driving and driving assistance**, including dynamic route guidance, truck platooning, or driverless cars.

- **Intelligent traffic management**, including self-organising traffic light control, anticipatory vehicle routing, proactive and cooperative junction control, or bus priority handling.

Most of the applications of agent-based techniques can be found in road transport; the second most addressed mode of transport is air transport. In contrast, work on agent-based approaches to rail and in particular water transport is still scarce. With key technological enablers now in place, the adoption of multiagent systems solutions is likely to increase significantly for all transport modes in the coming years.

1.2 Overview of My Research

Compared to the majority of the published research on agents in traffic and transport, my work treats transport problems on a higher-level of abstraction. Instead of detailed vehicular traffic modelled and/or controlled over short time frames (seconds to minutes), my research concerns mobility of people and freight over longer time periods (tens of minutes, hours or even days).

Overall, my research addresses three main topics:

1. **Simulation and modelling** where I explore agent-based simulation modelling techniques for gaining insight and foresight regarding the operation of complex transport systems (see Section 1.3 for more details).

2. **Journey and route planning**[3] where I explore agent-based and other artificial intelligence techniques for helping people and vehicles to travel efficiently in large-scale, possibly multimodal transport systems (Section 1.4).

3. **Transport resource allocation** where I explore novel agent-based ways of coordinating the use of potentially scarce vehicle and infrastructure capacity in transport systems (Section 1.5).

From the many transport application subdomains, my research focuses on two areas. First (from around 2009 until 2012), I worked on problems concerning maritime transport, in particular the security aspects of long-distance merchant shipping (see [36] for a summary). More recently (from around 2011 onwards), I have shifted my focus and I have been conducting research on agent-based techniques for integrated multimodal transport and mobility[4], particularly in the urban context.

Most of my research addresses new, previously scientifically little explored yet practically significant problems, as opposed to perfecting solutions to already well-established problems. This was the case of all my work on applying agent-based techniques to fighting maritime piracy and is also the case of most of my work on integrated multimodal urban mobility. Table 1.1 provides a list of all my 26 research papers relevant to the topic of this habilitation thesis, categorized based on the topic and the transport domain. Nine of the papers are included in the thesis; the rest can be obtained on-line.

[3] I use the term *journey planning* when the result of the process is a sequence of logical instructions for executing a trip, such as when planning for public transport trips; I use the term *route planning* or simply *routing* when the result corresponds directly to a path in the physical transport network, such as when planning for cars or bicycles.

[4] I use the term *transport* to stress the technical and physical aspects of moving people and freight around. I use the term *mobility* when considering broader behavioural, sociological and economic aspects of travel.

	MULTIMODAL URBAN TRANSPORT	MARITIME SHIPPING
SIMULATION	• AgentPolis: towards a Platform for Fully Agent-based Modeling of Multi-modal Transportation (2012) [33] • **Modular Framework for Simulation Modelling of Interaction-Rich Transport System (2013) [32]** • Agent-based Simulation Testbed for On-demand Transport Services (Demonstration)(2014) [58] • Agent-based Simulation Testbed for On-demand Mobility Services (2014). [57] • Simulation Testbed for Autonomic Demand-responsive Mobility Systems (2014, to appear) [56]	• AgentC: Agent-based System for Securing Maritime Transit (Demonstration) (2011) [35] • Using Multi-Agent Simulation to Improve the Security of Maritime Transit (2011) [50] • **Agent-based model of maritime traffic in piracy-affected waters (2013) [52]** • Using Data-Driven Simulation for Analysis of Maritime Piracy (2013) [54]
JOURNEY PLANNING AND ROUTING	• **Mixed-Reality Testbeds for Incremental Development of HART Applications (2012) [34]** • SUPERHUB: a user-centric perspective on sustainable urban mobility (2012) [13] • **Generalised Time-Dependent Graphs for Fully Multimodal Journey Planning (2013) [29]** • Bicycle Route Planning with Route Choice Preferences (2014) [30] • **Exploring Pareto Routes in Multi-Criteria Urban Bicycle Routing (2014) [47]** • Personalized Fully Multimodal Journey Planner (2014) [31] • **Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach (2014) [28]** • Advanced Public Transport Network Analyser (2014) [44]	• Transiting Areas Patrolled by a Mobile Adversary (2010) [50] • Iterative Game-theoretic Route Selection for Hostile Area Transit and Patrolling (2011) [53] • Computing Time-Dependent Policies for Patrolling Games with Mobile Targets (2011) [10] • **Extending Security Games to Defenders with Constrained Mobility (2012) [49]**
RESOURCE ALLOCATION	• **A Profit-Aware Negotiation Mechanism for On-Demand Transport Services (2014) [24]** • **Mechanism Design for Profitable On-Demand Services (submitted 2014) [23]**	
CROSS-TOPIC		• Employing agents to improve the security of international maritime transport (2010) [38] • Using agents to improve international maritime transport security (2011) [37] • Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy (2012) [36]

Table 1.1: Overview of my publications related to the topic of the habilitation thesis. Publications included in the thesis are marked in bold. See full references in the Bibliography section at the end of the thesis.

1.3 Topic 1: Agent-based Simulation Modelling of Transport and Mobility

The already high complexity of transport systems continues to rise. New types of transport services are introduced, increasing the number of entities in transport systems and the number of combinations in which they work together. Ubiquitous connectivity and advancing automation dramatically increase the speed, frequency and reach of human-vehicle-infrastructure interactions in transport systems, further contributing to their rising complexity. Such developments bring the dynamics of transport systems closer to that of large-scale *complex adaptive systems* [43] and, consequently, make their operation difficult to understand and foresee.

Simulation modelling is an established approach for studying complex socio-technical systems; it is therefore also applicable for analysing the behaviour of transport systems. Unfortunately, existing popular transport modelling toolkits do not provide sufficient support for modelling some of those aspects of transport systems that contribute strongly to their complexity. This is because the traditional transport modelling approaches rely on equation-based or discrete-event simulation modelling – while powerful, these approaches are not well suited to model systems with autonomous, adaptive agents with goal-oriented behaviour and context-dependent interactions. Specifically, the traditional approaches lack the support for modelling *anytime, ad hoc interactions* between the actors of the transport system and the just-in-time decision making required for participating in such interactions. Capturing both well is essential for accurately modelling the behaviour of present and, in particular, future autonomous transport systems.

Over the past 20 years, agent-based simulation modelling has emerged as a powerful new paradigm for system modelling. *Agent-based simulation modelling* (ABSM)[9] is a bottom-up modelling approach that models the behaviour of the target system at the level of its constituent individual actors. In contrast to top-down modelling approaches, the global, system behaviour is not modelled explicitly in agent-based models but emerges from the behaviour and interaction of individual actors. Because of this, agent-based simulation modelling is well suited for modelling systems involving goal-oriented, adaptive, and frequently interacting actors. Hence, given our analysis in Section 1.1.1, agent-based simulation models are a well-suited tool for modelling and studying complex, interaction-rich transport systems. Because of the relative novelty of the agent-based simulation approach, its *genuine*[5] applications in the transport domain are, however, still relatively scarce and limited to specific subproblems, in particular road and air traffic microsimulation.

Below, I summarize my research contributions to agent-based simulation modelling in both the maritime traffic and the multimodal urban transport domain.

1.3.1 Agent-based Modelling of Piracy-affected Maritime Traffic

The use of simulation models to support policy design and operational management has a long tradition in the transport field. The vast majority of the work,

[5] Although some transport models promote themselves as agent-based, they only employ the agent concept on a superficial level. In particular, they leave out of consideration the critical properties of agent autonomy and reactivity, both of which are important for capturing adaptive and emergent system properties.

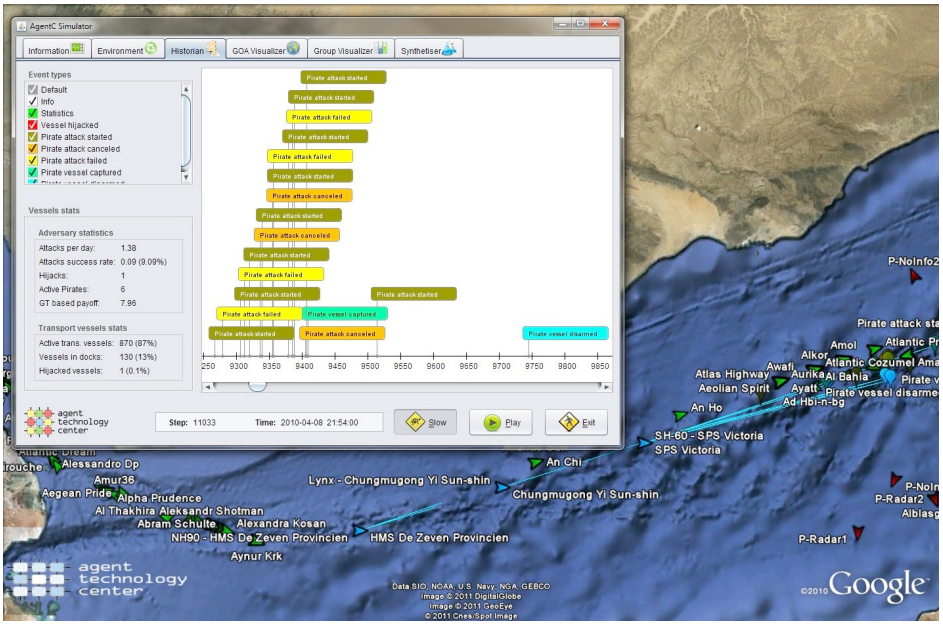


Figure 1.2: AgentC-based simulation of maritime shipping traffic and pirate activity in the Gulf of Aden. See Appendix A for more information.

however, focuses on ground transport [8].

In the maritime domain, applications of simulation models are surprisingly scarce. This was in particular true in the case of decision support for fighting maritime piracy, which posed, and to a great extent still poses, a major security threat for the global shipping industry. At the time when this research was conducted (i.e., years 2010-2012 when the piracy problem peaked), existing simulation modelling work in the maritime domain either focused on traffic in ports and national, coastal waters [26] or used high-level equation-based models [11] unfit for capturing individual-level behaviour and inter-vessel interactions essential for modelling maritime piracy. Furthermore, none of then existing models was concerned with the security of maritime shipping lanes[6].

This is why, together mainly with my Ph.D. student Ondřej Vaněk and in close collaboration with the U.S. Office of Naval Research, I developed agent-based techniques and software tools (known as the AgentC platform, see Figure 1.2 for an illustrative screenshot) for modelling maritime traffic in piracy-affected waters. The AgentC model represents the movement and other activities of merchant, navy and pirate vessels in piracy-affected waters of Indian Ocean. Modelling the behaviour and interactions of thousands of individually simulated vessels, the model is capable of capturing the complex dynamics of the maritime transport system threatened by maritime piracy and, consequently, allows assessing the effect of a range of piracy countermeasures.

The development of the simulation model required collecting, analysing and integrating a variety of data sets, conducting interviews with domain experts, developing new techniques and technologies for simulating the behaviour of individual vessels (including e.g., global voyage route planner) and their interactions.

[6]Recently, new work, citing our research, was published that specifically deals with (agent-based) modelling or piracy (e.g., [55, 20]).

It also required the development of a novel methodology for data-driven calibration and validation of agent-based simulation models. At the time of its release in 2012, AgentC was the first simulation model (of any kind) of maritime traffic in piracy-affected waters and, in fact, the first application of agent-based simulation to global maritime shipping modelling.

The results of our work have been published in a series of papers [35, 51, 54, 52], the last and most comprehensive of which

> Ondřej Vaněk, Michal Jakob, Ondřej Hrstka and Michal Pěchouček: Agent-based Model of Maritime Traffic. In *Piracy-affected Waters. Transportation Research Part C: Emerging Technologies.* 2013, vol. 36, p. 157–176. ISSN 0968-090X. (my contribution: 30%)

is included in the thesis in Appendix A.

Because of its timeliness and originality, our work of maritime piracy simulation attracted strong interest of several leading institutions actively involved in fighting maritime piracy, in particular the United Nations International Maritime Organization (London, UK), U.S. Navy Research Lab (Monterrey, CA, USA), NATO Undersea Research Centre (La Spezia, Italy), Naval Postgraduate School (Monterrey, CA, USA) and Space and Naval Warfare Systems Center Pacific (San Diego, CA, USA). The developed AgentC simulation platform has been transferred to the U.S. Navy Research Lab for further development and potential operational use.

1.3.2 Fully Agent-based Modelling of Transport Systems

As already argued, due to the strongly multiagent nature of transport systems, agent-based simulation modelling is a particularly suitable and effective approach for modelling transport and mobility. The agent-based approach has been particularly widely adopted in *microscopic traffic simulations* [22] which model, with a very high-level of detail, the behaviour of individual vehicles and their interactions with other vehicles and the road infrastructure. At present, a variety of mature commercial as well as open-source simulation software tools exist for traffic microsimulation.

The agent-based approach has also been applied for modelling transport and mobility on a higher-level of abstraction. Demand-responsive transport systems [27], taxis [16], ride-sharing [21] and vehicle-sharing [3] services have all been simulated using agent-based approaches. In contrast to microsimulation models, however, there are no readily available agent-based modelling tools that allow expressing complex interactions of vehicles, drivers, passengers and dispatchers typical for such transport and mobility services. This is also the case for the otherwise relevant MATSim simulation framework [2] – although MATSim uses individual-level modelling, it treats individuals as passive data structures whose state can only be updated synchronously by central modules at infrequent, predefined points in time. Such a centralized approach introduces a significant modelling gap – in reality, agents in transport systems make just-in-time decisions asynchronously at different occasions throughout a day, often in reaction to external observations or communication – and consequently makes it difficult to model transport systems which rely on frequent interactions.

Because of the lack of suitable toolkits, almost all of higher-level agent-based transport simulation models have therefore been developed from scratch using general-purpose programming languages (most often C++ or Java). In the case

of few exceptions where general purpose toolkits were employed (e.g., [18, 41]), model developers faced considerable difficulties expressing and implementing required model behaviour using their chosen toolkit; this resulted in long development times and/or reduced fidelity of implemented models.

This is why, building on the experience from maritime traffic simulation, I set out, together with my colleague Zbyněk Moler, to develop a flexible toolkit for fully agent-based simulation modelling of multimodal transport systems. The result of the work is the AgentPolis simulation platform, designed from its inception to support the modelling of interaction-rich multimodal ground transport systems. The work on the AgentPolis platform was described in two publications [33, 32], the latter of which

> Michal Jakob and Zbyněk Moler: Modular Framework for Simulation Modelling of Interaction-Rich Transport Systems. In *Proceedings of the 16th IEEE Intelligent Transportation Systems Conference (ITSC)*, p. 2152–2159, 2013. (my contribution: 60%)

is included in the thesis in Appendix B.

As one of its main applications, we have utilized the AgentPolis platform for implementing an agent-based model of multimodal mobility. Adopting the activity-centric mobility modelling approach, our model aims to simulate travel in a large-scale multimodal transport system. Initially developed for the South Moravian region, the model is going to be later extended to cover the whole area of the Czech Republic.

Furthermore, recognizing the rising importance of on-demand transport services, I, together with my colleague Michal Čertický and a former Ph.D. student Radek Píbil, used the AgentPolis platform as a basis for the *flexible mobility services simulation testbed*. The testbed allows researchers and practitioners to easily evaluate and compare the performance of different vehicle routing and allocation mechanisms under various scenarios. The testbed was described in a series of publications [57, 58, 56]. Importantly, the flexible mobility services testbed has been released as an open source software and is freely available at `https://github.com/agents4its/mobilitytestbed`.

1.3.3 Mixed-Reality Testbeds for Autonomous Systems

Despite the valuable role computational simulation plays in foreseeing the operation of transport systems, in certain cases simulation alone may not provide enough accuracy and, consequently, confidence for critical decisions to be made. In such cases, evaluation on models that more closely corresponds to the real-world system under consideration is needed.

That is why, in a parallel but related research stream to my transport simulation modelling work, I explored the concept of *mixed-reality testbeds*. In mixed-reality testbeds, computational simulation is complemented with the use of real physical assets in order to construct models that better approximate the operation of the target system. In order to match the scale of the real-world systems, mixed-reality testbeds use a low number of physical assets and complement them with a high number of computationally simulated assets. This way realistic issues, such as those arising from imperfect sensors, physical dynamics or real-world communication channels, can be identified and explored, without costly and potentially risky full-scale tests with the target physical system.

Our main contribution in the area of mixed-reality testbeds was a proposal of a methodology for incremental multi-level mixed-reality development. The methodology allows using mixed-reality testbeds of various sizes and virtualization levels in a way that maximizes the effectiveness of autonomous system development and evaluation, in the transport domain and beyond.

The results of this research have been published in [34]

> Michal Jakob, Michal Pěchouček, Michal Čáp, Peter Novák, and Ondřej Vaněk: Mixed-Reality Testbeds for Incremental Development of HART Applications. In *IEEE Intelligent Systems*. 2012, vol. 27, p. 1541–1672. (my contribution: 35%)

which is included in the thesis in Appendix C.

1.4 Topic 2: Advanced Multimodal Journey and Route Planning

The growing complexity of transport systems and the accent on their optimum utilization, drives the development of advanced tools that help people make right travel decisions.

Although journey and route planning have been traditionally studied by graph algorithm researchers in the computer science community, the artificial intelligence and multiagent systems perspective has been gaining importance. The need to employ agent-oriented approaches arises when planning journeys with the consideration of other actors, both in cooperative settings (such as in ridesharing) and in non-cooperative settings. Journey and route planning problems involving multiple actors are best framed as multiagent problems and solved using techniques from multiagent planning and computational game theory. The strong drive towards *personalized* journey planning, in which journey recommendations are tailored to the specific needs of each individual traveller, is another impulse for more agent-oriented approaches. This is because personalized journey planning requires agent-oriented modelling techniques capable of representing user's constraints and preferences effectively.

My research contributions to the field of journey planning and routing concerns all of the above aspects.

1.4.1 Real-time Fully Multimodal Journey Planning

My first contribution concerns fully advanced multimodal journey planning in the urban context. The advent of new types of mobility services, such as bike, electric scooter or car sharing, real-time carpooling or next-generation taxi, has further expanded the already rich portfolio of means of travel available in modern cities. Providing intelligent tools that would help citizens make the best use of the mobility services on offer is thus needed more than ever. Despite recent algorithmic advances [4], existing journey planners address this need only partially. In particular, they only consider a limited subset of transport modes and their combinations and do not have full support for working with real-time information.

This is why, together with my Ph.D. student Jan Hrnčíř, I have conducted research on *fully* multimodal journey planning that supports the full spectrum of available mobility services and their combinations. In our approach, a journey can consist of any combination of scheduled public transport modes (e.g., bus,

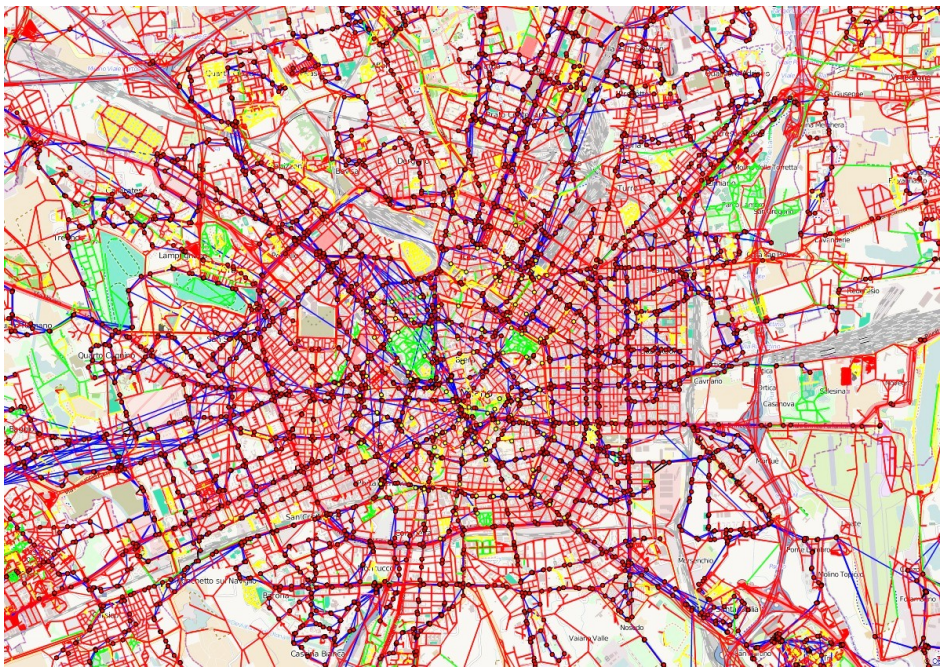


Figure 1.3: A visualization of the generalised time-dependent graph representing the multimodal transport network of Milan, Italy. See Appendix D for more information.

tram and underground), individual modes (e.g., walk, bike, shared bike and car), and on-demand (e.g., taxi) modes. We have adopted a representation-centric approach to solving the fully multimodal journey planning problem. Instead of providing purpose-specific journey planning algorithms, we have introduced *generalised time-dependent (GTD) graphs* that allow representing the fully multimodal journey planning problem as a standard graph search problem and consequently allow using general shortest path algorithms to solve it (see Figure 1.3 for an example instance of the GTD graph). Importantly, this approach allowed us to reuse the GTD representation and associated tools for a wide range of additional applications.

In a subsequent development, we have further extended our GTD representation to support time-varying historic and real-time information about conditions in the transport system (i.e., actual traffic flow speeds, delays and disruptions in the public transport network, or the availability of bicycles in bike-sharing stations).

The results of our research have been integrated in the core journey planning component of the SUPERHUB platform for sustainable multimodal urban travel, developed under the *SUstainable and PERsuasive Human Users moBility in future cities (*SUPERHUB*)* project [13]. Through the SUPERHUB project field trials, our journey planning algorithms were successfully tested by several thousand users in four big European cities (Barcelona, Milan, Helsinki and Brno).

The results of our work on fully multimodal journey planning have been published in [31, 29], the latter of which

for Fully Multimodal Journey Planning. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*. 2013, p. 2138–2145. (my contribution: 30%)

is included in this thesis in Appendix D. An article describing the extension of the GTD representation towards efficiently handling real-time information is currently in preparation.

The GTD representation and selected journey planning algorithms have also been leveraged in our research on analysing accessibility in multimodal transport systems, early results of which were briefly described in [44]. Importantly, our research on transport accessibility analysis has been integrated into a working prototype of an online transport network analyser, which is available, at the time of writing, at `http://transportanalyser.com`. Since its launch in September of 2013, the online transport analyser has been used by several thousand users.

1.4.2 Bicycle Routing with Realistic Route Choice Preferences

In contrast to car and public transport journey planning, for which advanced algorithms and mature software implementations exist [4], bicycle route planning is a surprisingly underexplored topic. Although numerous bicycle route planning applications have recently emerged (e.g., Cyclestreets[7] or BBBike[8]), these applications follow ad-hoc approaches and provide very little information about their internal models and search algorithms.

Interestingly and importantly, compared to car drivers, cyclists consider a significantly broader range of factors while deciding their routes. By employing questionnaires and GPS tracking, researchers have found that besides travel time and distance, cyclists are sensitive to slope, turn frequency, junction control, noise, pollution, scenery, and traffic volumes [12]. Moreover, the relative importance of these factors varies among cyclists and can also be affected by weather conditions and the purpose of the trip [12]. Such a user- and context-dependent multi-criteriality makes bicycle routing a particularly difficult category of routing problems.

This is why together primarily with my postdoc Qing Song and master student Pavol Žilecký, I have conducted research on bicycle routing that takes realistic route choice preferences into account.

Our first contribution to bicycle routing has been a proper formalization of the multi-criteria bicycle routing problem. Although relatively straightforward, such a formalization had not been previously available. The flexible, hierarchical model we have developed relies on sets of features, criteria and costs to capture the rich semantic information contained in the underlying bicycle network and map data in a form amenable to multi-criteria shortest path search.

Our second contribution to bicycle routing focused more on the algorithmic part of the problem. We applied the multiple label correcting algorithm [42] for finding a full set of Pareto routes in a multi-criteria bicycle routing problem. To reduce the potentially very large number of Pareto solutions, we have introduced a route selection algorithm, based on hierarchical clustering, for extracting a small representative subset of Pareto routes.

[7] `http://www.cyclestreets.net/`
[8] `http://www.bbbike.org/`

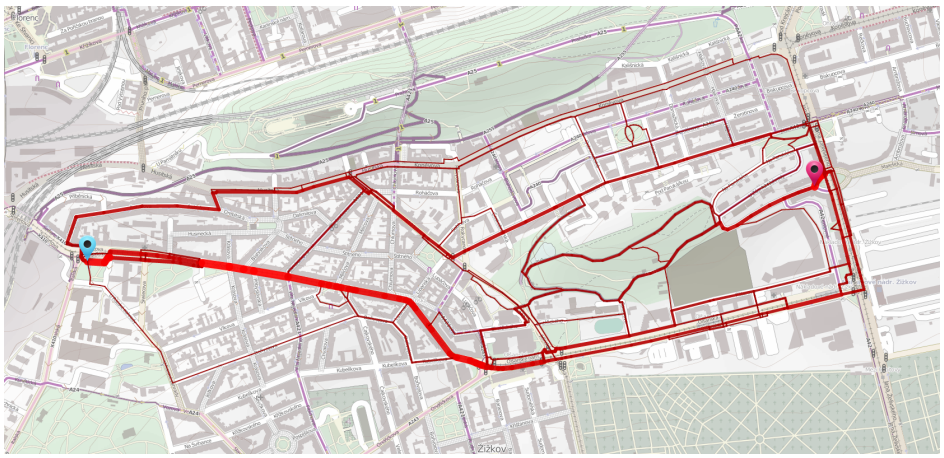


Figure 1.4: The results of multi-criteria bicycle route search in the urban environment of Prague (503 Pareto routes shown - the thicker the edge, the more Pareto routes follows the edge). See Appendix E for more information.

The results of our research have been published in two papers [30, 47], the latter of which

> Qing Song, Pavol Žilecký, Michal Jakob and Jan Hrnčíř. Exploring Pareto Routes in Multi-Criteria Urban Bicycle Routing. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*. 2014. (my contribution: 25%)

is included in the thesis in Appendix E. A follow-up journal paper studying the effect of speed-up heuristics for accelerating the generally very time-consuming optimal multi-criteria search is under submission.

The results of our research on bicycle routing have also been integrated in a working bicycle route planner that is available, at the time of writing, to the public at `http://cykloplanovac.cz`.

1.4.3 Planning Shared Journeys on Timetabled Transport Services

One way to tackle traffic congestion is through ridesharing, i.e., purposeful and explicit planning to create groups of people travelling together in a single vehicle for parts of the journey. Participants in such schemes can benefit from ridesharing in several ways: sharing parts of a journey may reduce cost (e.g., through group tickets), carbon footprint (e.g., when sharing a private car), and travellers can enjoy the company of others on a long journey.

Ridesharing is a known and studied problem – existing work (e.g., [7]), however, focuses exclusively on ridesharing using vehicles that can move freely on a road transport network, without schedule or route restrictions. Planning shared rides on timetabled public transport network has not been previously addressed.

To address this gap, I contributed, together with my Ph.D. student Jan Hrnčíř and his former M.Sc. supervisor Michael Rovatsos, to the development of a novel agent-based approach. The approach employs the recently introduced domain-independent *best-response multiagent planning* [39] and specializes it for the spe-

cific purpose of planning shared journeys on timetabled transport services. The key benefit of the approach is its scalability to real-world public transport networks.

The results of the research have been published in [28]

> Jan Hrnčíř, Michael Rovatsos and Michal Jakob: Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach. Special issue of Journal of Intelligent Transportation Systems: Technology, Planning, and Operations. 2014. (my contribution: 20%)

which is included in the thesis as Appendix F.

1.4.4 Route Planning in Adversarial Scenarios

Route planning in adversarial settings presents a very challenging variant of route planning problems. Assuming the ability of agents to reason about the actions and strategies of the opponent, such route planning problems are best studied within the context of non-cooperative game theory and in particular security games [48].

There are two main variants of the problem. In the first variant, the agent needs to find a route crossing a controlled area such that the agent avoids (or maximizes the probability of avoiding) being detected and/or intercepted by the adversary. In the other variant of the problem, the agent needs to find a patrolling route around a sensitive asset such that the chance of an adversary successfully attacking the asset is eliminated or at least minimized.

Traditional models, such as those put forward by the ambush games and search games frameworks [45, 25], only consider one side of the problem setting to be mobile. In the research conducted together primarily with Ph.D. students Ondřej Vaněk and Branislav Bošanský, we have extended existing models and algorithms towards situations where both agents are mobile.

The first stream of work concerned the development of a routing strategy for a patrolling agent protecting a group of mobile vulnerable assets. Motivated by the work on navy escorts protecting groups of vessels transiting pirate waters, the solution required an extension of models and algorithms for patrolling games [1] towards situations where the protected assets are mobile. The results of this line of research have been published in [10].

The second stream of work addressed the problem of optimum route finding for an agent crossing an area roamed by a mobile adversary with a fixed base and limited endurance. Motivated by the problem of merchant ship routing through piracy waters, the solution of the problem required the development of novel single- and double-oracle methods for the efficient computation of optimal randomized routing strategies. The results of this work have been published in a series of papers [50, 53, 49], the last of which

> Ondřej Vaněk, Branislav Bošanský, Michal Jakob, Viliam Lisý and Michal Pěchouček: Extending Security Games to Defenders with Constrained Mobility. In *Proceedings of AAAI Spring Symposium on Game Theory for Security, Sustainability, and Health.* 2012. (my contribution: 10%)

is included in the thesis in Appendix G.

Overall, the work on route planning in adversarial domains has revealed the combinatorial challenges of route planning with strategic, game-theoretic models.

While in the standard journey planning setting, transport networks comprising millions nodes and edges can be efficiently searched, even significantly smaller problems (thousands of nodes) can become hard to solve when considered in the adversarial, game-theoretic setting.

1.5 Topic 3: Agent-based Resource Allocation for On-Demand Transport Services

The availability of near-ubiquitous internet connectivity and the widespread adoption of GPS-equipped smartphones have enabled new types of mobility services – such as real-time ridesharing, free-floating bike, scooter and car sharing or peer-to-peer parking. The common property of such services is that they support and often require pre-arrangement and/or booking by the user. Given the limited capacity of such services, a key problem, particularly in the periods of high demand, is how to allocate the available service capacity to users.

In general, the problem of resource allocation has been widely studied by the multiagent research community [17]. Various models for different types of allocated goods, languages for representing agent preferences, measures of social welfare as well as protocols and strategies for allocation procedures have been explored.

In the road transport domain, however, the problem of resource allocation has mostly been studied under the assumption of full cooperation. This is certainly the case with the vast majority of work on vehicle allocation and routing in pickup and delivery problems [6], which consider the vehicles as fully controlled by a single dispatcher. The cooperative setting is an adequate model for transport operations run by a single company. It is, however, ill-suited for the emerging open transport service marketplaces which continuously match large numbers of self-interested transport providers with customers seeking transport services.

That is why, together with my postdoc Malcolm Egan, I started investigating the problem of vehicle allocation in competitive formulations of vehicle allocation and routing problems. Our overall approach is to leverage general marketplace-based allocation techniques (in particular auctions [60, Chap. 7]) and to adapt and specialize them for specific problems in transport resource allocation.

In our first contribution, we explicitly considered profit maximization (as opposed to traditionally considered cost minimization) in the dial-a-ride [19] category of pickup and delivery problems [6]. We have proposed a novel profit-aware negotiation mechanism that accounts for both passenger and service provider preferences. The negotiation mechanism prices each passengers journey, in addition to providing vehicle routing and scheduling. We have proved a stability property of our negotiation mechanism using a connection to hedonic games. We have also showed via simulations the dependence of the service provider profit and passenger prices on the number of passengers as well as passenger demographics. Our key observation was that increasing the number of passengers has the effect of increasing passenger diversity, which in turn increases the service providers profit.

The results of this research have been published in [24]

> Malcolm Egan and Michal Jakob. A Profit-Aware Negotiation Mechanism for On-Demand Transport Services. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, p. 273–278. 2014.

(my contribution: 40%)

which is included in the thesis in Appendix H.

Because of the recency of the topic, the above is our only publication on competitive transport resource allocation published at the time of submitting the thesis. Because of the importance of the topic for my future research and in order to provide a more comprehensive exposition of the topic, I also include a journal manuscript [23]

> Malcolm Egan and Michal Jakob. Market Mechanism Design for Profitable On-Demand Transport Services. Submitted to *Transportation Research Part B: Methodological*. 2014. (my contribution: 25%)

which has been submitted and is currently under review. The manuscript generalizes some of the assumptions of the earlier paper [24] and reframes the problem in a way better aligned with the standard research on multiagent resource allocation. The manuscript is included in the thesis in Appendix I.

Chapter 2

Outlook and Future Work

There has never been a better time for research on multiagent systems for transport and mobility. With the continuation of current trends, the multiagent properties of transport systems identified in Section 1.1.1 are going to become even more pronounced in the future. In particular, as automation progresses, transport systems will gradually morph into massive multiagent systems in which millions (and later billions) of people, (semi-)autonomous vehicles and transport infrastructure elements continuously interact to deliver smooth mobility services. For engineering and managing such future transport systems, multiagent systems models and techniques will thus be essential.

In this section, I outline several directions in which I personally would like to contribute to the future development of agent-based techniques in transport and mobility.

2.1 Common Future Themes

There are several themes which are common to my future research in all of the three main research directions I pursue (i.e., simulation, journey and route planning, and agent-based resource allocation).

Perhaps the most important theme is the significantly increased utilization of large real-world data sets and data analysis techniques in transport systems modelling, management and optimization. As transport and mobility embrace the big data era, there is a strong growth in the size and variety of available transport data. Fully utilizing the potential of such data will be key to further improve the quality of models and efficiency of algorithms for intelligent transport systems. I will give more specific examples below when talking about the individual research directions.

The other important theme concerns research methodology. Given their size, complexity and variety, statistically relevant insights into the operation of transport systems require conducting very high numbers of computational experiments. The research process in intelligent transport systems thus, in addition to becoming more data-driven, needs to be scaled up to take the full advantage of high-performance computing resources available. This will require the development of design of experiments techniques that optimize the configuration of experiments so as to maximize information gain from each experiment and, consequently, to minimize the number of experiments needed to reliably answer research questions of interest. I aim to further develop our initial work in this strategic direction as

it is instrumental to increasing our empirical research capacity across a range of topics.

2.2 Future Research in Transport and Mobility Simulation

There are several specific directions in which I, together with my colleagues, aim to advance research on transport and mobility simulation.

In line with the big data trends mentioned, we aim to incorporate large data sets on human movement extracted from cellular telecommunication networks in our multimodal mobility simulation models. To this end, we aim to develop new machine-learning-based techniques for automatically extracting, calibrating and validating activity-centric mobility models from real-world trip and activity data.

We also aim to scale up the number and size of transport simulation experiments performed and to take advantage of newly available supercomputing facilities. To this end, we will upgrade our simulation management and design of experiment tools to support parallelized execution and automated processing of thousands of simulations involving up to millions of agents on distributed computing infrastructures.

Such an upgrade should allow us to successfully complete the development the agent-based multimodal mobility model of the whole Czech Republic, which requires simulating over 10 million agents. It will also allow us to scale up simulation-based empirical study of agent-based vehicle allocation and routing techniques for on-demand mobility. By conducting large-scale computational experiments, we aim to gain deeper insights into how the topology of the transport network, the spatio-temporal distribution of travel demand and the size and structure of the vehicle fleet affect the performance of on-demand mobility services.

2.3 Future Research in Journey and Route Planning

As far as journey planning and routing is concerned, I aim to continue our research both on fully multimodal journey planning and multi-criteria bicycle routing.

One of the recent practically motivated yet scientifically challenging problems in multimodal journey planning is *planner integration*. In planner integration, multiple existing journey planners, each with limited geographical and/or transport mode coverage, are combined in order to provide a fully multimodal journey planning service across the whole region of interest. To address this problem, we have already successfully tested a novel *meta-planning* approach to planner integration, which utilizes an abstracted meta-level representation of the planning problem to guide the journey plan search. To make the approach practically applicable, we aim to tackle several challenging issues, in particular the choice of a suitable meta-level representation of the multimodal transport network, the construction of the meta-level transport network representation from minimum map and service data, and efficient meta-level search algorithms and control strategies.

We will also continue our promising research on bicycle routing in challenging urban conditions. First, we aim to improve speed-up techniques in order to bring multi-criteria search times for realistic trip distances to under one second. Next, we aim to extend the underlying cycle network graph representation to more

accurately capture real-world features considered by cyclists in their route choice. Related to this, we aim to develop methods that would improve bicycle routing by learning more accurate models of cyclists' behaviour from real-world GPS tracks data. Finally, we plan to continue collaborating with human-machine interface researchers on user-friendly ways for exposing bicycle routing functionality to users on mobile and wearable devices.

2.4 Future Research in Agent-based Transport Resource Allocation

Market-based resource allocation is currently perhaps the most promising area for the application of multiagent systems in transport and mobility. In this direction, we aim to extend our initial work on profit-maximizing negotiation protocols for on-demand transport services. Our immediate objective will be to better align our model with the general multiagent resource allocation framework and, in particular mechanism design and auction theory, in order to be able to exploit relevant formal models and theoretical results. This should bring us closer to our long-term research objective, which is to understand the fundamental trade-offs between the efficiency, fairness and profitability in on-demand transport services markets.

In a parallel quest for model fidelity and real-world relevance, we aim to utilize newly acquired data sets about real-world operation of on-demand transport services. By employing intelligent data analysis, we aim to build more accurate models of passenger and driver behaviour, and to use such models to optimize the passenger-vehicle matchmaking process, which is central to transport services marketplaces.

Once we have better understood the behaviour of on-demand transport services markets in the single-passenger setting, we aim to extend our approach towards shared rides. In contrast to existing work, we aim to ground our approach to ridesharing in coalitional game theory – this should allow deriving principled solutions that are also individually rational from the passenger's perspective and which should thus improve the stability and long-term viability of ridesharing systems.

2.5 Further Research Opportunities

The future opportunities are not limited to the three main research directions I have been already pursuing.

Valuable results can be in fact obtained by leveraging the synergies between the individual research lines. For example, we can use our agent-based simulation framework to explore the behaviour of transport services marketplaces in a wide range of conditions. Because of the complexity of the problem, only very limited results can be obtained formally and computational studies are essential for understanding how different design choices affect the performance of market-based resource allocation.

Another opportunity lies in the combination of multimodal journey planning with transport resources allocation. In fact, the ability to have tickets and reservations required for a trip automatically arranged is crucial for the concept of seamless door-to-door mobility. We have already made initial steps in this direc-

tions by introducing the concept of journey plan *resourcing*. A principled solution of the problem, however, will require an in-depth exploration of how planning and resource allocation should be mutually combined to provide reliable journey plans in capacity-limited environment.

There are also exciting opportunities immediately outside of the area of my research to date. Many of our results on simulation, journey planning and resource allocation for passenger mobility transfer well to freight logistics. Of particular interest is the problem of same-day (or even same-hour) delivery in urban areas with its substantial challenges of allocating and routing vehicles in a highly-dynamic environment.

The fast maturing concept of *connected vehicles* [14] (and the broader development of the *internet of things*) opens up a whole new area for the application and further development of our results. By providing standard interfaces to vehicle-to-vehicle and vehicle-to-infrastructure interaction, connected vehicle platforms will enable integrating our vehicle routing and allocation techniques with the car localization and navigation subsystems, and consequently pave the way towards fully autonomous multiagent transport systems.

Ultimately, many of the future research topics outlined above are likely to become the part of the newly emerging discipline of *computational transportation science* [61]. Promoted as the science behind intelligent transport systems, computational transportation science aims to combine computer science and engineering with the modelling, planning, and economic aspects of transport planning and engineering to create more efficient, equitable, liveable and sustainable transport systems and communities.

2.6 Final Remark

Over the past years, I and my collaborators have acquired a firm grounding in the problems and solution techniques at the cross-section of multiagent systems, artificial intelligence and transport research. We have acquired a solid understanding of relevant formal models and algorithms, developed a modular stack of reusable software components, assembled a broad range of crucial real-world transport data sets and built strong links with a number of key academic and industrial players in the field. In the future, we aim to capitalize on these achievements and continue contributing strongly to the theory and practise of intelligent transport systems and computational transportation science.

Appendix A

Agent-based Model of Maritime Traffic in Piracy-affected Waters

O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček. Agent-based model of maritime traffic in piracy-affected waters. *Transportation Research Part C: Emerging Technologies*, 36:157–176, 2013.

Transportation Research Part C 36 (2013) 157–176

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Agent-based model of maritime traffic in piracy-affected waters

Ondřej Vaněk *, Michal Jakob, Ondřej Hrstka, Michal Pěchouček

Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo Náměstí 13, Praha 127 00, Czech Republic

ARTICLE INFO

Article history:
Received 16 January 2013
Received in revised form 9 August 2013
Accepted 9 August 2013

Keywords:
Agent-based simulation
Computational modeling
Maritime transportation
Maritime piracy
Policy assessment

ABSTRACT

Contemporary maritime piracy presents a significant threat to global shipping industry, with annual costs estimated at up to US$7bn. To counter the threat, policymakers, shipping operators and navy commanders need new data-driven decision-support tools that will allow them to plan and execute counter-piracy operations most effectively. So far, the provision of such tools has been limited. In cooperation with maritime domain stakeholders, we have therefore developed AgentC, a data-driven agent-based simulation model of maritime traffic that explicitly models pirate activity and piracy countermeasures. Modeling the behavior and interactions of thousands of individually simulated vessels, the model is capable of capturing the complex dynamics of the maritime transportation system threatened by maritime piracy and allows assessing the potential of a range of piracy countermeasures. We demonstrate the what-if analysis capabilities of the model on a real-world case study of designing a new transit corridor system in the Indian Ocean. The simulation results reveal that the positive past experience with the transit corridor in the narrow Gulf of Aden does not directly translate to the vast and open waters of the Indian Ocean and that additional factors have to be considered when designing corridor systems. The agent-based simulation development and calibration process used for building the presented model is general and can be used for developing simulation models of other maritime transportation phenomena.

1. Introduction

Global maritime shipping lanes are a critical part of the world's transportation infrastructure. 90% of internationally traded goods are transported by sea at least one point in their journey (Earnest and Yetiv, 2009). In the past years, the global maritime transportation system has come under a serious threat from maritime piracy. As a major security and economic threat costing the global economy up to estimated US$7bn (Bowden et al., 2011), contemporary maritime piracy has solicited a concerted international response which has, finally, led to the reduction of the success rate of pirate attacks. The number of pirate attacks, average amount of ransom paid and the number of seafarers held in captivity, however, remain high. In 2011 and for Somali-based piracy alone, there were 181 attacks reported for Somalia, 28 vessels hijacked and 1118 seafarers were taken hostage.[1] Containing piracy also required and continues to require extensive deployment of naval forces which is unsustainable in a long term.

From the many levels on which solutions of the problem are sought, we focus on the operational management of the situation at sea, as this is the arena where progress can be made in the short term, before long-term sustainable solutions can

* Corresponding author. Tel.: +420 22 43 57 581.
E-mail addresses: ondrej.vanek@agents.fel.cvut.cz (O. Vaněk), michal.jakob@agents.fel.cvut.cz (M. Jakob), ondrej.hrstka@agents.fel.cvut.cz (O. Hrstka), michal.pechoucek@agents.fel.cvut.cz (M. Pěchouček).

[1] Source: International Maritime Bureau (IMB) Piracy Reporting Centre (website: http://www.icc-ccs.org/piracy-reporting-centre).

0968-090X/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.trc.2013.08.009

be developed onshore. To date, military, governmental and industry stakeholders have proposed several types of piracy countermeasures to increase the security of maritime transit, including recommended transit corridors, group transit, escorted convoy schemes, coordinated patrol deployments and on-board security teams. When properly designed and implemented, such measures can significantly improve maritime transportation security with reasonable additional cost. However, due to complex spatial and temporal dependencies between individual countermeasures and external factors, discovering effective, synergistic combinations of piracy countermeasures presents a major challenge.

To address this challenge, we have built AgentC, a data-driven agent-based simulation model of maritime activity in piracy-affected waters. The model aims at helping decision makers reduce uncertainty about the effects of their operational control and regulatory interventions. The model incorporates a wide range of real-world data and, to our best knowledge, is the first computational model that simulates deep sea shipping down to the level of individual vessels. This is crucial for accurately capturing emergent, collective effects arising from the context-dependent interactions of merchant, pirate and navy vessels.

Due to the lack of prior work on the topic, the development of the model prompted the development of a novel methodology for agent-based maritime transportation modeling. Some parts of the methodology could be borrowed from more mature transportation modeling fields; other parts had to be developed from scratch. In addition to the AgentC model, the developed methodology, presented alongside the model itself, is therefore the second major contribution of the paper.

The rest of the paper is organized as follows. After reviewing related work, we describe the AgentC simulation model, detailing how the maritime environment, vessel behaviors and vessel interactions are modeled. We then briefly comment on the implementation aspects of the simulation model and devote significant space to discussing calibration of the model. Finally, we show how the developed model was employed to help answering specific operations research questions concerning the design of maritime transit corridor systems.

2. Related work

The use of agent-based or simulation-based models to support policy design and operational management has a very long-standing tradition in the transportation field. The vast majority of the work, however, focuses on ground transportation (e.g. Hidas, 2002; Waraich et al., 2013) and, to a lesser extent, on air transportation (e.g. Tang et al., 2012).

In the maritime domain, applications of simulation models are surprisingly scarce, as analyzed, e.g., by Davidsson et al. (2005). Existing work either focuses on traffic in ports and national, coastal waters (Hasegawa et al., 2004) or uses high-level equation-based models (Bourdon et al., 2007) unfit for capturing individual-level behavior and inter-vessel interactions essential for modeling maritime piracy. Furthermore, none of the above models is concerned with the security of maritime shipping lanes. Advanced computational methods have been applied in the maritime domain to optimize ship routing (e.g. Norstad et al., 2011; Øvstebø et al., 2011), albeit not taking the security aspect into account. In both cases, the authors use mathematical programming rather than simulation to solve the problem.

As far as the security angle on transportation systems is concerned, existing simulations focus on modeling activities in and around terminals rather than within transportation networks themselves. This is true both for airport security (Chawdhry, 2009) and port security (Koch, 2007). Port security has also been listed as an important area for the application of operations research methods (Crainic et al., 2009), of which simulations are an important representative. The spatial, network aspect of transportation security has been touched upon in the work on modeling critical infrastructures (Barton and Stamber, 2000), however, the emphasis there is mostly on other than transportation types of infrastructures. The problem of securing transportation infrastructures and logistical networks has only been studied in the military context (Ghanmi et al., 2011).

Focusing on the very phenomenon of maritime piracy, existing work is concentrated primarily in the fields of security studies, international relations and global policy (Onuoha, 2010). Only recently, initial attempts at applying computational modeling and optimization to maritime piracy have emerged but focus exclusively on military aspects of the problem: Bruzzone et al. (2011) model piracy around the Gulf of Aden using the discrete-event simulator PANOPEA. The authors focus on evaluating the efficiency and effectiveness of different Command and Control models; only main actors in the Gulf of Aden are considered and the simulation is not scaled to the Indian Ocean where the merchant traffic model is significantly more complicated.

Tsilis (2011) employs the MANA agent-based modeling framework (Lauren and Stephen, 2002) to identify key factors affecting the escort of vulnerable merchant vessels through the Gulf of Aden. The escorting scenario is modeled on a tactical level, focusing on positioning of individual ships and protection of one group of merchant vessels; this is different from our model which adopts a whole-system perspective and considers the security of maritime transportation system as a whole. The MANA framework is also used by Decraene et al. (2010) to analyze requirements on non-lethal deterrents for defending large merchant vessels against pirate attacks; again, the focus is on the tactical level of modeling a single encounter in detail, rather than the system as a whole.

Slootmaker (2011) describes *Next-generation Piracy Performance Surface (PPSN)* model which employs meteorological forecasts, intelligence reports and historical pirate incidents to predict areas conductive to pirate activity around the Horn of Africa. Hansen et al. (2011) further improve the PPSN model by refining the environment model and adding a probabilistic behavioral pirate model, resulting into the *Pirate Attack Risk Surface (PARS)* model. Both PPSN and PARS models are numerical

with only a minor simulation component and are limited to short-term forecasts (several days). They do not directly model real-world behavior and interactions of individual vessels; consequently, their applicability for what-if type of analysis is limited.

Finally, piracy patterns and the effect of countermeasures were also studied using statistical data analysis and data mining (Bowden et al., 2011). The usability of such results for policy design and optimization is limited because the insights gained concern the behavior of the maritime system under current circumstances and are difficult to extrapolate to hypothetical future scenarios.

3. Model description

The AgentC model represents the movement and other activities of selected categories of vessels in piracy-affected waters of the Indian Ocean (we focus on piracy with origins in Somalia, affecting Gulf of Aden, Arabian Sea and West Indian Ocean). A visual overview of the model and its constituent entities is given in Fig. 1.

The description of the model proceeds as follows: after explaining the agent-based modeling methodology, we describe the model of the maritime environment and models of each vessel class in detail. Given the importance of vessel interactions, we separately describe the model of the pirate attack and of selected piracy countermeasures.

3.1. Agent-based modeling methodology

As already stated, we employ *individual-centric* modeling approach, in which the behavior of the modeled system is represented at the micro-level of individual vessels. Vessels are modeled as autonomous agents (Russell et al., 2010) capable of moving freely within the navigation boundaries of ocean waters while interacting with the maritime environment, other vessel agents and other actors (such as shipping operators or traffic coordinators).

Based on the literature and discussions with domain experts, we identified *merchant vessels*, *pirate vessels* and *navy vessels* as main vessel classes which are therefore explicitly represented in our model as vessel agents. For most of the time, each vessel agent pursues its individual goals, however, there are situations where multiple vessel agents interact—such interactions are either non-cooperative (such as pirate attacks or navy warship counter-pirate interventions) or cooperative (such as merchant vessel agents' requests for help to navy vessel agents). Vessel agent interactions play a critical role in the dynamics of maritime piracy and make the agent-based, micro-simulation approach vital for accurately modeling the effect of piracy on maritime transportation, primarily because it allows capturing the phenomena naturally and it provides the detail of analysis not attainable with macro-level equation-based methods (Van Dyke Parunak et al., 1998).

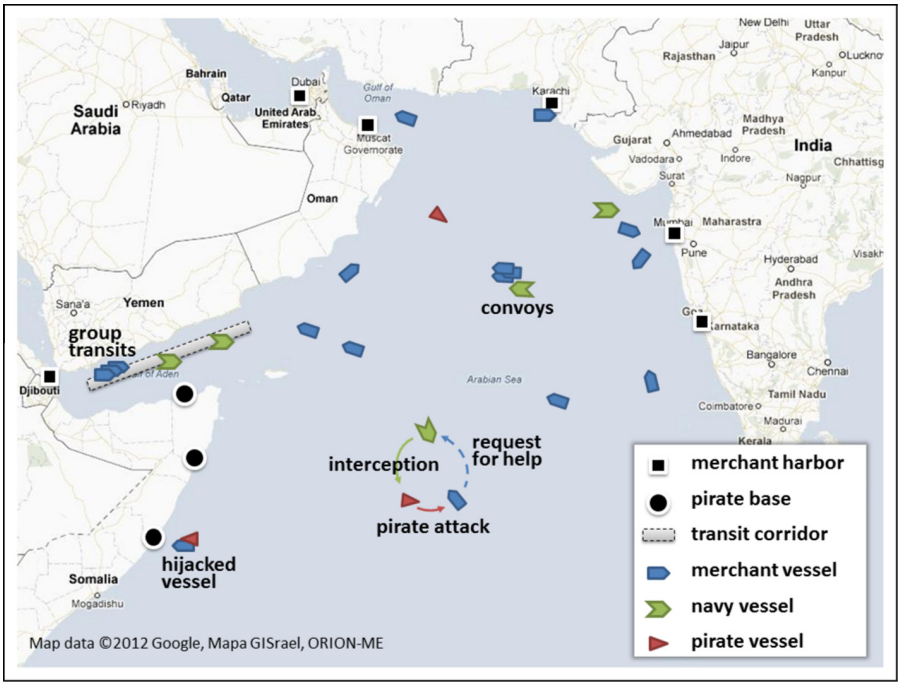


Fig. 1. Key actors, activities and environmental features represented in the AgentC model of piracy affected waters.

In line with the agent-based modeling approach, the model for each class of vessels consists of an individual *vessel behavior model* and a *vessel population model*. The vessel behavior model represents the executable behavior of an individual vessel agent; such behavior can depend on *vessel parameters* assigned to each individual vessel. The vessel population model specifies how many vessels of each class are generated and how the values of vessel parameters are assigned.

The choice of vessel parameters is based on relevant literature (Bruzzone et al., 2011; Tsilis, 2011; Decraene et al., 2010) and was discussed with domain experts; their influence (i.e., importance) was also explored using sensitivity analysis (described in Section 5.2). For each vessel class, we list the parameters in a table together with the intervals of parameter values considered in the simulation. For each simulation instance, depending on the scenario executed, we either sample the value of a parameter uniformly from its respective interval or set the parameter value to a constant from the interval.

3.1.1. Note on Naming

In the military domain, merchant vessels are denoted as MV, pirate vessels—recently mainly operating as a group of one mothership and multiple accompanying speedboats—are denoted as pirate attack group (PAG) and navy vessels as coalition forces (CF). We denote the vessel classes as M, P and N respectively. Furthermore, although we use the term vessel and vessel agent rather interchangeably, we use the latter if we want to stress the behavioral aspect of vessel description. With some simplification, vessel agent can be viewed as the shipmaster controlling a respective vessel.

3.2. Maritime environment model

The environment model represents physical maritime environment in which the vessels operate. It consists of two principal components:

- *geography, bathymetry*—represent the geography of the maritime environment in terms of a set of spherical *obstacle polygons*$\mathcal{O}$ representing land masses, shallow waters and other obstructions that limit navigability. This component also contains locations of ports and anchorages used in merchant shipping and pirate activities.
- *weather*—represents the environmental conditions affecting the behavior of modeled vessels, specifically, wave height, wind speed and currents. Wave height plays an important role in pirate's decision making; currents and wind slightly alter routes of small vessels (e.g., pirate boats).

3.3. Merchant shipping model

Merchant vessels are large ocean-going vessels carrying cargo over long distances between world's major ports. Merchant vessels are the primary targets of pirate attacks. In order to be useful for what-if analysis of different counter-piracy measures and policies, the merchant traffic model has to produce realistic traffic patterns even for situations which diverge from the current status quo in terms of pirate operations and the configurations of piracy countermeasures deployed. The merchant traffic model therefore cannot solely mimic current real-world merchant vessel routes but it has to be capable of generating realistic shipping traffic from more fundamental principles. We therefore adopt the approach of separating the modeling of transportation demand from traffic routing. Demand is considered given and fixed while the routes are generated dynamically, based on the assumption that merchant vessel agents maximize their utility and take the most advantageous route possible. Such an approach is widely used in ground transportation modeling, its application to global maritime shipping, however, is novel.

3.3.1. Merchant shipping origin–destination matrix

The demand for merchant transportation is represented in terms of an *origin–destination matrix (O–D matrix)* which specifies the volume of merchant traffic between world's major ports. The O–D matrix is used to generate origins and destinations for individual merchant vessel voyages and the voyage planning module is then used to find optimum routes connecting voyage endpoints.

Unfortunately, in contrast to ground traffic modeling, no data explicitly and completely capturing the merchant shipping O–D matrix is available; we were therefore forced to estimate the matrix from several partial sources. We have extracted the most important ports in and near the observed area from CI-online database and Ports and Ships[2] portal. We then estimated the O–D matrix by fitting generated traffic to known real-world traffic densities (see Section 5.3).

3.3.2. Voyage planning

A fundamental part of the merchant vessel operation is voyage planning. Voyage planning is primarily used in the model of merchant vessels to generate realistic merchant traffic from the merchant shipping O–D matrix; however, it is also used in the operation of the other two vessel classes.

We model voyage planning as an optimization problem of finding an optimal route C between an *origin* and a *destination* point on a sphere, given vessel-specific route optimality criterion, a set of constraints imposed by geographical boundaries

[2] http://www.ci-online.co.uk, http://ports.co.za/.

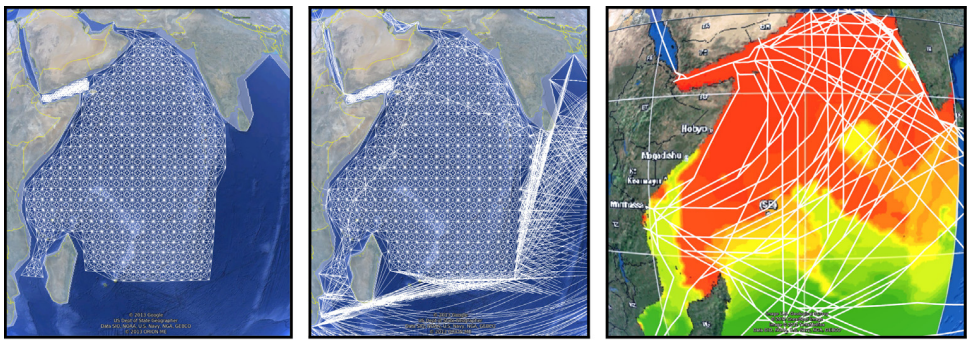


Fig. 2. (a) Rectangular grid representing the piracy risk area, (b) cell-grid connected to the visibility graph, (c) PARS model for October 1st, 2011, provided by NATO Shipping centre—green, yellow, red colors correspond to low, medium and high risk of attack respectively; plans are generated for each O–D matrix entry with risk aversion coefficient $\alpha = 0.5$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and physical properties of the vessel, and a spatial piracy risk function (the latter is only used in voyage planning for merchant vessels).

Mathematically, we formalize the optimal route selection problem as a bi-objective optimization problem

$$\begin{aligned} &\min \quad \mathcal{L}(C), \mathcal{R}(C) \\ &\text{s.t.} \quad C \in \mathcal{C} \end{aligned} \tag{1}$$

where $\mathcal{L}(C)$ is the length of route C on the sphere, $\mathcal{C}$ is the set of all valid routes, i.e., routes going from the origin to the destination respecting geographical boundaries, and $\mathcal{R}(C)$ is the risk of a pirate attack along route C, computed as

$$R(C) = \int_C r ds \tag{2}$$

Here,

$$r(x, y, t) : \mathbb{R}^3 \to \mathbb{R}_0^+ \tag{3}$$

is termed the *piracy risk function* and captures the expected density of pirate attacks for a given time and location, defined by latitude x and longitude y. The piracy risk function can be constructed based on past pirate incident reports,[3] or another risk model in the form of a spatio-temporal or spatial function can be used, e.g., the NATO Shipping Centre[4] pirate activity map.

To solve the optimization problem (1), we transform the length and risk criteria into a scalar criterion function using the aggregation method (Hwang et al., 1979) with a single weight:

$$\min \quad (1 - \alpha)\mathcal{L}(C) + \alpha\mathcal{R}(C) \tag{4}$$

The weight α is termed the *risk aversion coefficient* and it can be set individually for each merchant vessel agent based, e.g., on the level of on-board security, vessel cruising speed or the value of its cargo.

In order to leverage efficient path-finding algorithms, we discretize the ocean surface space and represent it as a graph. First, we define the *piracy risk area* as a spherical polygon $O_{\mathcal{R}}$ which delineates the space in which piracy attacks occur and where we explicitly consider piracy risk in the routing process. For discretization, we divide the piracy risk area $O_{\mathcal{R}}$ into a rectangular *latitude-longitude grid* G_r (Sahr et al., 2003) with predefined cell widths. We create a graph $\mathcal{G}_1(V_1, E_1)$ from the grid G_r: the set of vertices V_1 comprises all vertices from the grid and the set of edges E_1 is the set of all geodesics[5] between any two neighboring vertices (considering 16-connected cells (Boult et al., 1993); see Fig. 2a).

Second, we construct a *spherical visibility graph* $\mathcal{G}_2(V_2, E_2)$. The vertices V_2 of the visibility graph comprise all vertices of all spherical polygons $\mathcal{O} \cup O_{\mathcal{R}}$ ($\mathcal{O}$ is the set of obstacle polygons, see Section 3.2). The set of edges E_2 of the visibility graph is the union of the set of all geodesics between all vertices V_2 which do not intersect any polygon from $\mathcal{O} \cup O_{\mathcal{R}}$ (Fig. 2b) and the set of entry and exit edges, where the entry and exit edges are all such edges connecting the voyage's origin and destination point to the visibility graph $\mathcal{G}_2$ that do not intersect any polygon from $\mathcal{O} \cup O_{\mathcal{R}}$ and any edge from E_1. We set the risk value on edges outside the piracy risk area $O_{\mathcal{R}}$ to 0; for edges inside the piracy risk area $O_{\mathcal{R}}$, we set the risk value to the integral of the piracy risk function (3).

[3] As provided e.g. by the International Chamber of Commerce Piracy Reporting Centre at http://www.icc-ccs.org/.
[4] NATO Shipping Centre website: http://www.shipping.nato.int.
[5] The shortest path between two points on a surface of a sphere.

Table 1
Merchant vessel parameters.

Parameter	Values	Description
Destination	port id	Destination port of vessel voyage
Docking time	[0,3] days	Docking time of a merchant vessel
Cruising speed	[10,20] kn	Vessel travel speed unless participating in a group transit
Ship size	[30,250] m	Size of the ship
Alertness	[0,60] h^{-1}	Frequency of checking for an approaching pirate
Risk aversion	[0,1] –	Risk aversion coefficient used in voyage planning

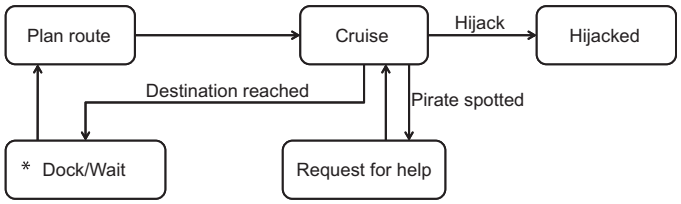


Fig. 3. Merchant vessel agent behavior model. The entry point is the *Dock/Wait* state. After docking in a port, the merchant vessel plans a route and cruises to its destination. If a pirate is spotted, a request for help is sent. In case the vessel is hijacked, the hijacked state is terminal and the merchant vessels is under the control of the pirate.

We compute the optimum vessel route in graph $\mathcal{G}(V_1 \cup V_2, E_1 \cup E_2)$ with respect to criterion (4) using the A* algorithm (Russell et al., 2010) with orthodromic distance[6] heuristics and with the cost function equal to the criterion function. For illustration, vessel routes generated by the route planner between all pairs of ports considered in the AgentC model are shown in Fig. 2c. Risk aversion coefficient $\alpha = 0.5$ and a spatial risk model obtained from the NATO Shipping Centre were used.

3.3.3. Merchant vessel population model

The merchant vessel population model instantiates a population of merchant ships of size *#M* with a realistic distribution of key vessel attributes, i.e., speed and size. It then samples the merchant shipping O–D matrix in order to assign each vessel a port of origin and a destination port to reach. All merchant vessel attributes specified by the population model are listed in Table 1. 2000 merchant vessels with speed and size distribution taken from a data-set of 2700 real-world vessel samples[7] were used in the simulation to replicate real-world shipping density in the Indian Ocean in Year 2011.

3.3.4. Merchant vessel agent behavior model

The behavior of a simulated merchant vessel is straightforward. Given a pair of origin–destination ports at the beginning of the simulation, the merchant vessel agent invokes the route planner to plan vessel's voyage, taking into account corridors and group transit schemes along its route. The merchant vessel then sets on cruising along the route. After the destination port is reached, a new port is sampled from the O–D matrix and a new route is planned. This basic behavior is interrupted if the vessel is attacked by a pirate vessel, in which case the merchant vessel agent reports attack to nearby merchant vessel agents, notifies the closest navy vessel agent and employs self-defense measures (see Section 3.6). Activities and transitions of the merchant vessel agent behavior are depicted in Fig. 3.

3.4. Navy operations model

Navy vessels represent military vessels operating in piracy-affected waters and capable of using force to deter and disrupt pirate activities. AgentC focuses on modeling the part of Navy vessel operation consisting in providing assistance to vessels subject to pirate attacks. It does not model active search and area patrolling operations, although such extensions can easily be incorporated in the model assuming data about such operations (which are typically classified) are obtained.

3.4.1. Navy vessel population model

Navy vessel population model instantiates *#N* navy vessel agents with specified deployment locations. The deployment locations can be specified manually by a human expert or obtained as a result of an optimization process (see below). All navy vessel parameters are listed in Table 2.

[6] The shortest distance between any two points on the surface of a sphere.
[7] The data-set is a subset of the Vesseltracker (http://www.vesseltracker.com) database.

Table 2
Navy vessel parameters.

Parameter	Values	Description
Helicopter	Y/N	Presence of helicopter on board the navy vessel
Patrolling location	GPS Coords	Area at which the navy vessel is located and from where it can respond to nearby pirate attack
Action radius	[100,200] nm	Distance on which the navy vessel reacts to distress calls
Response speed	[20,30] kn	Speed at which the vessel sails to intercept pirate attack
Helicopter Speed	[140,170] kn	Speed of the on-board helicopter

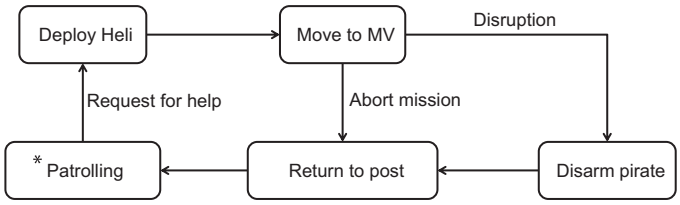


Fig. 4. Navy vessel agent behavior model. The entry point is the *Patrolling* state. The navy vessel reacts on a request for help by deploying a helicopter (if available) and cruises towards the merchant vessel. If the navy vessel or the helicopter arrives before the merchant vessel is hijacked, the pirate is disarmed and the vessel (and helicopter) returns to its assigned location. The intervention terminates unsuccessfully if the pirate successfully completes the hijacks of the merchant vessel.

3.4.2. Navy vessel behavior model

The basic behavior of a navy vessel comprises staying in its deployment location waiting for possible distress calls from nearby merchant vessels threatened by pirates. If a distress call is received, the navy vessel agent responds by dispatching a helicopter (if available) and by moving at its cruising speed to the attacked merchant vessel, trying to intercept the attack. Once the response has been completed, the navy vessel returns to its original deployment position. More details about navy vessel involvement in pirate attacks are given Section 3.6. The behavior model of the navy vessel agent is depicted in Fig. 4.

3.4.3. Navy vessel location assignment model

Similarly to the merchant traffic model, the navy operations model cannot solely replicate existing real-world deployment locations, but it needs to be able to take into account hypothetical merchant traffic flows in diverse evaluated what-if scenarios. We have therefore developed a navy vessel location assignment algorithm which takes the density of merchant traffic as the input and produces navy vessel deployment locations. The choice of locations aims to maximize the proportion of merchant traffic that lies within the action radius of deployed navy vessels.

We formalize the problem of optimal asset allocation as an optimization problem. We consider the *merchant traffic density functionm*$(x,y) : \mathbb{R}^2 \rightarrow \mathbb{R}_0^+$ specifying the average density of merchant vessels at a location specified by latitude x and longitude y. We assume that each navy vessel can protect merchant traffic that lies within the vessel's action radius and that the probability that the navy vessel will prevent the attack decreases exponentially with the distance from the navy vessel. Specifically, we model the probability of attack prevention as

$$p(x,y|\mu_x,\mu_y) = exp\left(-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}\right) \tag{5}$$

where parameters μ_x, μ_y represent the lat/lon location of the navy vessel and we set $\sigma = \frac{1}{3}AR$ to approximate that the probability of successful attack prevention is very high near the navy vessel, declines sharply as the distance from the navy vessel increases and is almost zero once the attack happens outside the vessel's action radius AR.

The navy vessel location assignment optimization problem is then formulated as

$$\underset{\mu_x^1\ldots\mu_x^N,\mu_y^1\ldots\mu_y^N\in\mathbb{R}}{\arg\min} \int_A m(x,y) \prod_{i\in 1\ldots N}\left(1-p(x,y|\mu_x^i,\mu_y^i)\right)dxdy \tag{6}$$

i.e., we search for the positions μ_x^i, μ_y^i of all navy vessels such that the expected number of successfully attacked vessels is minimized.

Similarly to voyage planning, for practical reasons we solve the allocation problem in a discrete formulation. We introduce the *merchant traffic density map* $m_G : G_m \rightarrow \mathbb{R}_0^+$ where G_m is a rectangular latitude-longitude grid covering the navigable ocean waters. For any cell $c \in G_m$, $m_G(c)$ specifies the number of merchant vessels passing through the cell for a specified time interval (e.g., a month or a year). Density maps for each month are provided by, e.g., AMVER[8] or they can be generated for any time span from the AgentC model (see Section 5.3 for details).

[8] Automated Mutual-Assistance Vessel Rescue System (AMVER), http://www.amver.com.

Table 3
Pirate vessel parameters.

Parameter	Values	Description
Home anchorage	base id	Base from which the pirate vessel embarks and to which it returns
Cruising speed	[8,14] kn	Normal speed when traveling long distance between the base and a target location
Pursuit speed	[25,30] kn	Speed during the attack on a merchant vessel
Endurance	[7,21] days	The number of days the pirate vessel can stay at sea
Visibility radius	[5,12] nm	Maximum distance of a merchant vessel which the pirate can spot
Attack time	30 min	Duration of attack attempt
Cool-down time	[1,4] h	Time needed for recovery after an unsuccessful attack
Navy knowledge	[0,1]	Probability of knowledge about navy vessel position
Hijack prob. ρ_u	[0,1]	Probability of successful hijack of a merchant vessel cruising at 10 nm unaware of the pirate attack
Hijack prob. ρ_a	[0,1]	Probability of successful hijack of a merchant vessel cruising at 10 nm aware of the pirate attack

We define functions $\chi_x, \chi_y : G_m \to \mathbb{R}$ specifying the latitude $\chi_x(c)$ and the longitude $\chi_y(c)$ of the central point of cell $c \in G_m$.

The allocation algorithm searches for the set of cells that maximize the protection of merchant traffic. Specifically, we use an iterative greedy algorithm to determine navy vessel locations—in the ith iteration, the ith navy vessel is placed to the center of a cell c^i such that the following expression

$$c^i = \arg\min_{c^* \in G_m} \sum_{c \in G_m} m_G(c) \cdot p(\chi_x(c), \chi_y(c) | \chi_x(c^*), \chi_y(c^*)) \cdot \prod_{j=1}^{i-1} p(\chi_x(c), \chi_y(c) | \chi_x(c^j), \chi_y(c^j)) \quad (7)$$

is minimized. The expression $\prod_{j=1}^{i-1} p(\chi_x(c), \chi_y(c) | \chi_x(c^j), \chi_y(c^j))$ captures the protection provided by navy vessels placed in preceding iterations.

3.5. Pirate activity model

Pirate vessels range from small skiffs up to large motherships acting as a floating base from which speedboats are launched to attack. We model piracy at the level of individual pirate attack groups which are represented by a single pirate vessel agent having its home anchorage and operating in and around main shipping lanes, where it attempts to attack, board and hijack passing merchant vessels.

3.5.1. Pirate population model

The pirate population model is currently simple and is only used to generate *#P* pirate agents and to assign them their *home-anchorage* parameter. The assignment is based on reported estimates of the number of pirate attack groups operating from each known pirate anchorage. All parameters of the pirate are listed in Table 3.

3.5.2. Pirate vessel agent behavior model

The pirate vessel agent behavioral cycle consists of three stages:

1. *Cruising*—the pirate vessel moves directly to its selected target area and looks for a suitable merchant vessel to attack. If the pirate vessel agent spots a navy vessel, it steers away temporarily. When the pirate vessel reaches the target area, it moves at a low speed and changes its course randomly from time to time.
2. *Attack*—If a suitable merchant vessel is spotted, pursuit starts (described in detail in Section 3.6).
3. *Recuperation*—After a successful attack or when running out of supplies (*endurance* parameter), the pirate agent navigates back to its home anchorage. After an unsuccessful attack, the pirate recovers (*cool-down time* parameter) and looks for a merchant vessel again.

Activity diagram of pirate vessel agent behavior is depicted in Fig. 5. Note that we do not model the economic aspect of piracy, such as ransom negotiation and other processes taking places after a hijacked vessel is brought to shore.

3.5.3. Target attack area selection mechanism

A key part of pirate vessel decision-making is choosing its target area where it will look for a merchant vessel to attack. Again, in order to have the pirate activity reflect the simulated scenario, target area cannot be predefined based on attack locations currently observed in the real world but needs to be determined dynamically from more fundamental principles.

AgentC therefore implements an algorithm that determines attack locations dynamically, from the assumption of pirate's rationality and partial knowledge of both merchant vessels and naval patrols: pirate's target area is thus selected based on the following inputs: weather conditions, the merchant traffic density map in a form of a grid and a (partial) knowledge about navy vessel positions.

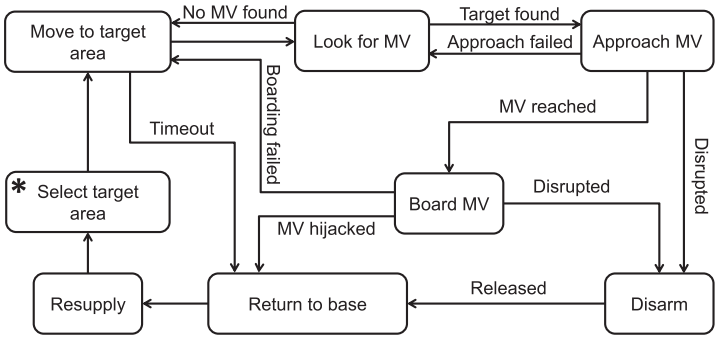


Fig. 5. Pirate vessel agent behavior model. The entry point is the *Select target area* state. After the pirate reaches the target area previously selected, it looks for a suitable merchant vessel to approach and hijack. The hijack attempt can be unsuccessful, interrupted by a navy vessel—in which case the pirate is disarmed and sails home—or successful, in which case he sails with the merchant vessel to its home base.

Algorithm 1. Target area selection algorithm

1: *regions* ← *selectRegions*(*date*)
2: *densityMap* ← *merge*(*NavyPos*,*MerchantMap*)
3: *cList* ← *getCells*(*densityMap*,*regions*)
4: *totalValue* ← *sumValues*(*cells*)
5: **for** *cell* ∈ *cells* **do**
    cell.prob ← *cell.value*/*totalValue*
6: **end for**
7: *cell* ← *sample*(*cell*)
    return *cell*

The algorithm for target area selection is described in Algorithm 1. In the first step, a subset of regions with acceptable weather conditions for a given date is selected.[9] In the second step, navy vessels positions—if known—are combined with the merchant traffic density map (this procedure is equal to the navy vessel location assignment algorithm with positions of navy vessels being already known) and only cells from the merchant traffic density map within the regions with the acceptable weather are considered (*cList* on the 3rd line of the algorithm).

Each cell from *cList* has a probability assigned which is proportional to the value of the cell taken from the merchant traffic density map, i.e., we divide the value of each cell by the sum of values of all cells in *cList* (line 5–6). Finally, a simple random sampling mechanism returns a cell from *cList*, i.e., each cell is chosen with a probability proportional to the value of the cell in the risk map.

The above location selection mechanism approximates the combination of two fairly complex driving forces behind the pirate's decision-making process: (1) the pirate wants to maximize its expected reward and prefers high density cells to low density cells; and (2) the pirate is subject to game-theoretic interaction in the sense that he does not want to be predictable (Tambe, 2011). These two counter-going objectives of pirate are captured by the weighted randomization over the merchant traffic density map with consideration of regions with unsuitable weather and known positions of naval vessels.

3.6. Pirate attack model

Pirate attack is a complex interaction between all three classes of vessels; we therefore provide its standalone description that complements the description of the attack from the perspective of individual vessel agent behaviors. Parameters directly influencing the course and the outcome of the attack are depicted in Table 4 and are a subset of parameters of individual vessel classes, except the M *awareness* binary parameter which is well-defined only during the attack phase. All interactions taking place during a pirate attack are depicted in Fig. 6. The attack consists of three phases:

1. *Pre-attack/approach*—this phase begins after a merchant vessel is spotted by a pirate vessel agent; the pirate vessel agent starts a pursuit at its pursuit speed. The merchant vessel agent checks for a, approaching pirate vessel several times per hour (parameterized by the alertness parameter capturing the probability of spotting an approaching pirate). If a pirate vessel is spotted during its approach, the awareness parameter is set to true, meaning that the mer-

[9] Based on the correlation of attack frequencies and weather conditions in 2011, we have estimated the acceptable wave height for piracy operations to be up to 1.25 m.

Table 4
Parameters affecting the outcome of a pirate attack.

Parameter	Values	Description
M Cruising speed	[10,20] kn	Cruising speed of the merchant vessel
M Alertness	[0,60] h^{-1}	How often the merchant vessel checks for pirate presence in its vicinity
M Awareness	Y/N	Merchant vessel's knowledge about an approaching pirate (element of surprise)
P Visibility radius	[5,12] nm	Maximum distance at which a merchant vessel can be spotted
P Pursuit speed	[25,30] kn	Cruising speed of the pirate
P Attack time	30 min	Maximum time for which the pirate attacks a merchant vessel
P Hijack prob. ρ_u, ρ_a	[0,1]	Probability of hijack of (un) aware merchant vessel
N Helicopter	Y/N	Presence of helicopter on board the navy vessel
N Action radius	[100,200] nm	Navy vessel distress call response radius
N Helicopter speed	[140,170] kn	Speed of navy vessel's on-board helicopter
N Cruising speed	[20,30] nm	Speed of a navy vessel

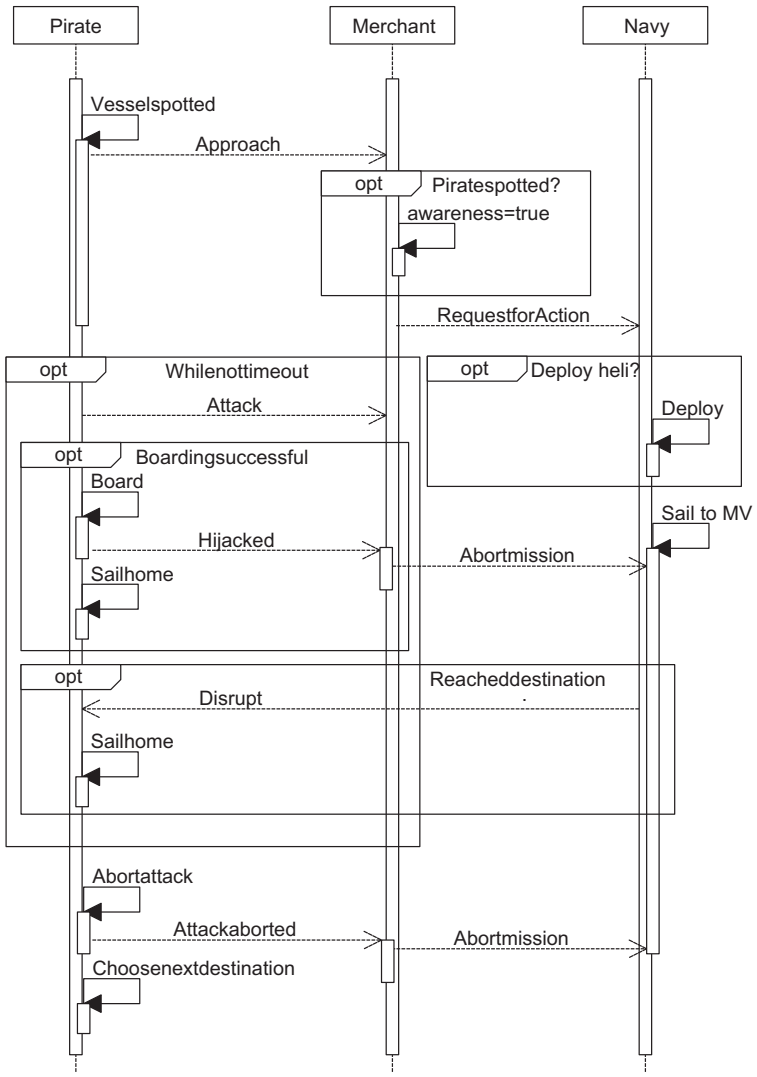


Fig. 6. Sequence diagram of the pirate–merchant–navy vessel interaction during a pirate attack. The attack takes a predefined amount of time and is terminated either by the successful hijack of the merchant vessel or by the arrival of a navy vessel or its helicopter.

chant vessel is not taken by surprise by the attacking pirate and can deploy self-defensive countermeasures. Furthermore, upon spotting the attack, the attacked merchant vessel agent broadcasts a distress call and notifies nearby

merchant vessel agents about the danger (their awareness is then set to true). If there is an idle navy vessel within the navy vessel action radius, the navy vessel responds by moving towards the attack. If the navy vessel carries an on-board helicopter, it dispatches the helicopter to prevent the pirate from hijacking the merchant vessel.

2. *Attack*—the pirate vessel agent attempts (repeatedly, for a time period, specified by the *attack-time* parameter) to board the merchant vessel and seize control. The probability of success depends on the speed s of the merchant vessel, its alertness a and subsequently on its awareness. The average probability of a merchant vessel being hijacked without any navy vessels present in the model can be computed as $p_h = a \cdot p_a(s) + (1 - a) \cdot p_u(s)$, where $p_a(s)$ and $p_u(s)$ are the probabilities of hijacking an aware and unaware merchant vessel, respectively, traveling at speed s. The hijacking probabilities are linear functions (with threshold) of merchant vessel speed: $p_a(s) = max\{0,(2 - s/v)\rho_a\}$, $p_u(s) = \max\{0,(2 - s/v)\rho_u\}$, where ρ_a and ρ_u are base probabilities specifying the probability of hijacking a merchant vessel cruising at $v \geqslant 10$ kn (minimum cruising speed of a merchant vessels in our model is 10 kn).
3. *Post-attack*—if the attack is successful, the hijacked vessel is taken to the pirate's home anchorage; if the attack is aborted by the pirate (after attacking unsuccessfully for a period given by pirate's *attack time* parameter), the merchant vessel continues its voyage according to the original plan; the pirate vessel recovers from the attack for a specified period of time (*cool-down time* parameter) and then it looks for another target to attack. If the attack is interrupted by the navy, the pirate is disarmed and sails to its home anchorage without further trying to attack any merchant vessels.

3.7. Piracy countermeasures model

Merchant and navy vessels can engage in various piracy countermeasures designed to increase the security of passage through piracy-affected waters. Most of such measures require cooperation between multiple vessels and can be viewed as multi-agent coordination mechanisms that augment standard, single-agent vessel behaviors. Based on discussions with the maritime security community, we support the following operational piracy countermeasures in the AgentC model:

- *Recommended transit corridors*, which concentrate merchant traffic along a defined route connecting a sequence of waypoints. Such concentration of traffic facilitates protection from navy vessels; however, it also makes targeting transiting vessels easier for pirates. The corridors are modeled as extensions of the merchant voyage planner. Case study in Section 6 examines the effectiveness of corridor systems closer.
- *Group transit schemes*, which coordinate the timing of merchant vessel transit so that vessels pass high-risk piracy areas in groups. This improves mutual awareness and facilitates navy response; however, it makes the transit take longer as vessels have to follow a predefined schedule and may have to reduce their cruising speeds to match the speed of their respective transit group. The group transit schemes are modeled as an extension of the voyage planner; they assign time marks to a subset of waypoints in a plan and the merchant vessels is then required to be at those waypoints at given time.
- *Navy vessel deployments*, which deploy navy vessels in strategic locations from where they can provide assistance to nearby merchant vessels in case of a pirate attack. We consider only stationary deployments (see Section 3.4).
- *On-board security teams* consists in deploying armed security personnel on-board of vessels transiting high-risk areas capable of deterring attackers and denying them access to the vessel. This countermeasure is currently modeled by the alertness and awareness parameters of the merchant vessel.

Each countermeasure is parameterized by a set of parameters (see Table 5). Except for route randomization, all above measures are currently actively used, although convoy schemes are operated rather sporadically by national navies on an ad hoc basis. The usage of transit corridors and group transits is currently limited to the Gulf of Aden.

3.8. Simulation model outputs

By its very nature, the agent-based micro-simulation model allows recording and evaluating, at different levels, multitude of information about the behavior of the modeled maritime transportation system.

At the lowest level, each simulation run produces a detailed log of all events generated by vessel agents and their interactions among themselves and with the modeled maritime environment. One year of simulated maritime traffic generates hundreds of thousands of events recording vessel locations in time, state-transition events (e.g. *destination-reached* event,

Table 5
Piracy countermeasures considered, with sets of parameters by which they are specified.

Countermeasure	Parameters
Transit corridor	Sequence of GPS waypoints
Navy vessel deployment	Set of locations
Group transit scheme	Corridor, Speed levels, transit schedule (per speed level)
On-board security team	Alertness of merchant vessels

target-area-selected event, etc.) and events generated during vessel interactions (e.g., *pirate-spotted* event, *pirate-attack* event, *pirate-attack-disrupted* event and many others). Events logs can be used for studying micro-level behaviors involving one or more individual vessels. They can be also used for on-line visualization of individual simulation runs, which is crucial for presentation and face validation purposes.

Micro-level event logs are aggregated in time and/or space into meso-level spatio-temporal output reports describing the occurrence of specific event or a set of events over time or in geographical space. A particularly important type of output report at this level are *density maps* which capture the frequency of occurrence of a specific event in grid-discretized space (as an example, see Fig. 9a for a density map of pirate attacks). Meso-level reports are useful for understanding how a certain phenomena is geographically distributed, how it is changing in time or both. It can also be used for calibration and validation of the model.

Finally, at the highest-level, event logs and spatio-temporal reports are aggregated into simple numerical statistics summarizing the activity in the piracy-affected waters. Both security-related and operational output quantities are evaluated. The former includes *pirate attack count*, i.e, the number of all attacks over a given time period (e.g., a year), and *attack success ratio*, i.e, the number of successful attacks (i.e. hijacks) divided by the number of all attacks; the later includes *average transit distance* and *average transit duration*, which can be used to estimate operational shipping costs. The high-level statistics are used to gain insight into the global behavior of the modeled system under different circumstances and are also crucial for model calibration.

4. Model implementation

Agent-based maritime transportation simulation requires agent control architecture capable of expressing required individual and collective vessel behaviors. Vessel agents have to be able to execute long-running actions while reacting to interruptions. The minimum intelligent agent architecture that can handle such requirements is the model-based reflex agent architecture (Russell et al., 2010) with encapsulated deliberative modules handling route-planning and other complex reasoning tasks. The required class of behaviors should be implementable in a modular and extensible way, facilitating sharing of common behavior fragments between different classes of vessels. At the same time, the agent control architecture should be computationally efficient enough to handle thousands of simulated agents. Unfortunately, none of existing agent architectures or simulation platforms supports these requirements. AnyLogic[10] simulation software comes closest but is not suitable due to its commercial closed-source nature. We have therefore implemented our own agent architecture for executing individual agent behavior models.

The implemented architecture decomposes agent behaviors into individual activities (depicted as boxes in Figs. 3–5) which correspond to the principal activities of the vessel agent (such as cruise, board, and patrol) and their associated actions. Each activity stores its context when deactivated so that when reactivated, the context can be restored to continue the previously interrupted action. The transition between activities is conditioned by external (e.g., request for help) or internal events (e.g., cool-down time passed). Although limited (e.g., not capable of executing concurrent activities), this approach provides a good trade-off between expressiveness, modularity and computational efficiency.

Behavioral models are embedded and executed in a Java-based agent-based simulator built partially using the lightweight ALITE (Novák et al., 2012) multi-agent simulation toolkit and employing Google Earth for geo-spatial visualization (see Fig. 7). The simulator provides suitable abstractions for implementing the model of the maritime environment, environment-to-agent sensor interfaces and agent-to-agent communication protocols. Time-stepped simulation execution model is used although migration to the discrete event-based model is possible.

5. Model calibration and validation

The AGENTC model contains a wide range of parameters that needs to be specified. Most of these parameters were set based on consulting domain sources and experts. There are, however, also parameters which significantly affect the behavior of the model and for which no reliable sources exist—the values of these parameters were therefore determined through calibration against real-word data.

In the following sections, we describe the calibration and validation process employed. Specifically we describe the calibration methodology selected, sensitivity analysis, calibration of the merchant traffic sub-model, and calibration and validation of the complete model.

5.1. Calibration methodology

The purpose of the calibration step is to set the values of key model parameters so that the behavior of the model most closely reflects the behavior of the real system; this closeness of the model is measured in terms of several fitness criteria. Due to limited supply of computing resources, we performed greedy iterative calibration—in each iteration, we chose a subset of parameters most influencing the fitness criteria, found the optimal value of these parameters and fixed them in subsequent iterations. To further speed up calibration, we used different calibration fitness metric in each step. The ordering of

[10] AnyLogic http://www.xjtek.com/.

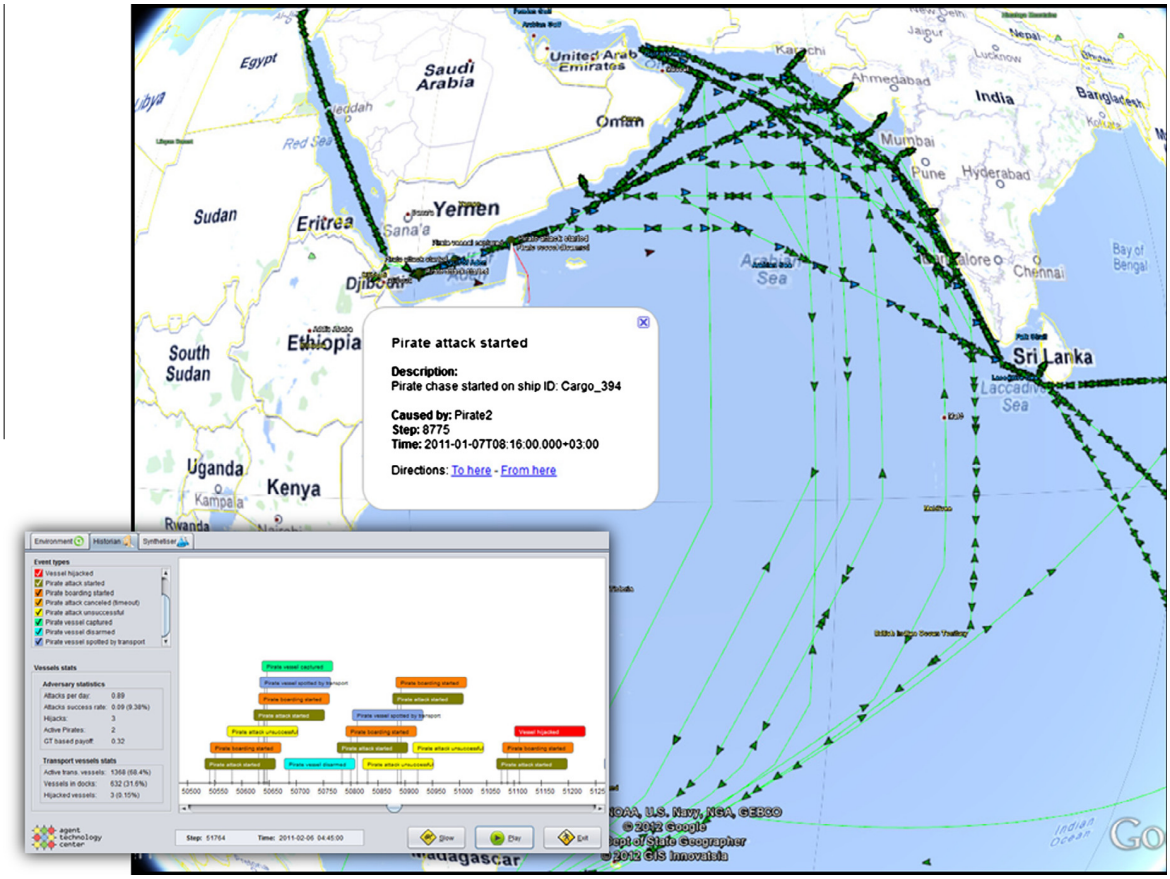


Fig. 7. Screenshot of the AgentC simulation. Google Earth-based visualization of the simulation execution and graphical user interface for simulation control and inspection of internal state of the model.

Table 6
Coefficients of variation for each criteria and model parameter. The bold-faced values correspond to parameters which were varied when calibrating the model for each criterion.

Parameter	Attack dist.	Attack freq.	Hijack ratio
#N	**0.15**	0.24	0.32
#P	0.046	**0.74**	0.041
P Visibility radius	0.052	**0.26**	0.11
M Alertness	0.053	0.075	**0.20**
P Hijack prob. ρ_a, ρ_u	0.057	0.078	**0.16**
P Navy knowledge	**0.1**	0.085	0.14

steps and the choice of calibrated variables and fitness metrics in each step was based on the results of the sensitivity analysis. Depending on the standard deviation of the selected fitness criterion, we executed between 50 and 100 simulation runs (with a different random generator seed) for each model configuration (i.e., each combination of model parameters – see Table 6).

In order to compare the spatial outputs of the model, we used spatial *success rate (SR) curves* as described by Chung and Fabbri (2003). Spatial success rate curves give a concise account of the performance of a spatial model. Specifically, the curve specifies what percentage of space a given spatial model needs in order to cover a given percentage of real-world occurrences of the event of interest (e.g., pirate attacks). The smaller the area required to cover a given percentage of the events, the tighter the fit and the better the model. The SR curve is constructed by discretizing the modeled area into a finite-size cells and then sorting the cells according to the relative frequency of the event occurring in the cell (e.g., relative frequency of pirate attacks). The SR percentage value for x percent of covering cells is determined by taking the x percent of the highest-frequency cells and counting the percentage of events occurring within these cells. SR curves can be modified in order to be used for comparing spatial models against spatial event density maps rather than sets of discrete events. In this case,

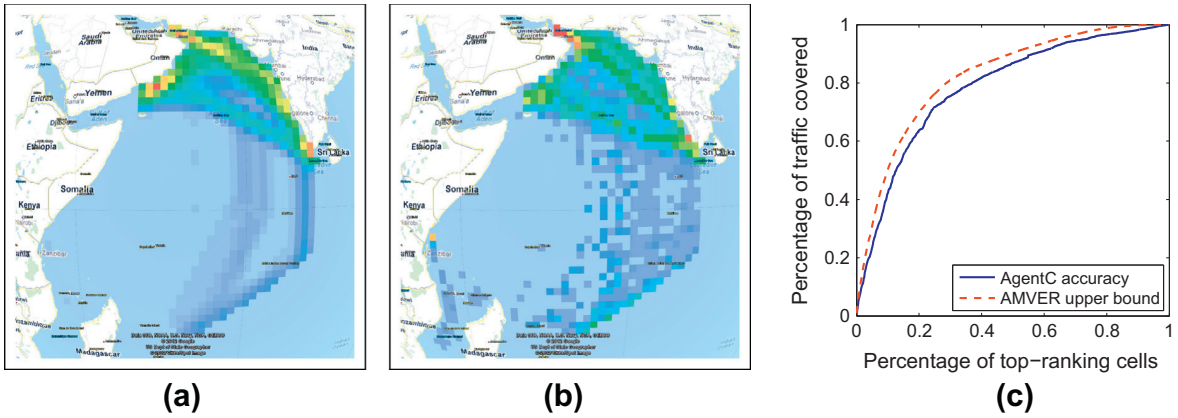


Fig. 8. Merchant traffic sub-model calibration. (a) Simulated merchant traffic density map G_m^S for merchant traffic sub-model. (b) Reference AMVER 2011 merchant traffic density map G_m^A. (c) SR curves for G_m^S (blue solid line) and G_m^A (red dashed line). The red SR curve of the AMVER map captures the theoretical upper-bound achievable for a given spatial resolution of the model: 20% of the AMVER map top ranking cells cover 70% of the AMVER map; 20% of the AgentC merchant traffic sub-model top ranking cells cover approximately 64% of the AMVER traffic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

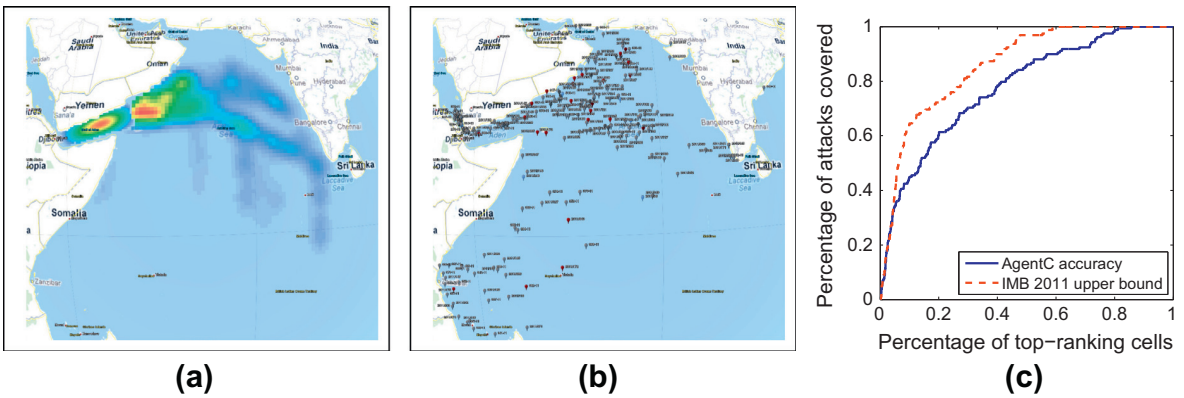


Fig. 9. Complete model calibration—attack spatial distribution fitting. (a) Density map for the complete model. (b) IMB 2011 Reports. (c) SR curves for the complete model (blue solid line) and IMB 2011 reports (red dashed line). The red SR curve for IMB 2011 reports was measured by transforming the incidents into a density map on which the SR curve was measured. This IMB density map thus serves as a theoretical upper-bound 20% of the most dense cells in the IMB density map covers approximately 72% and 20% of the AgentC model density map covers 61%. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

coverage is counted as the fraction of the overall mass[11] of the density map covered by a given proportion of highest-raking cells. We further define the *SR curve index* as the percentage of the area under the SR curve with respect to the overall rectangular plot area (i.e., the area below and above the curve). The interpretation of SR curves is illustrated in the captions of Figs. 8 and 9. To our knowledge, this is the first use of SR curves for calibration of simulation models.

5.2. Sensitivity analysis

Before the actual calibration, we performed sensitivity analysis in order to understand how the variation of key model parameters affects model outputs: (1) attack distribution, (2) attack frequency and (3) attack success ratio. Table 6 summarizes the sensitivity of each output to the variation of the given model parameter, measured in terms of the coefficient of variation[12] of a given output variable when varying a given model parameter. For each of the model outputs, we have selected the most sensitive parameters which were then varied while the rest of the parameters remained fixed.

[11] Where the mass of a density map is the sum of values in all cells. Note that we assume the space is discretized into finite-sized cells, e.g., a latitude-longitude grid utilized for voyage planning or navy vessel location assignment (see Sections 3.3.2 and 3.4).

[12] Coefficient of variation is defined as a ratio of standard deviation and mean: $c_v = \frac{\sigma}{\mu}$.

5.3. Merchant traffic sub-model calibration

In the first calibration step, we calibrated the merchant traffic sub-model of the AgentC model, i.e., the model consisting solely of merchant vessels. The calibration involved estimating all entries in the O–D matrix (see Section 3.3.1) together with the risk aversion parameter $\alpha \in [0,1]$ (see Section 3.3.2) such that we minimize a difference between the simulated and real-world merchant traffic. We used the SR curve index between the simulated merchant traffic density map in form of the latitude-longitude grid G_m^s and reference 2011 merchant traffic density map grid G_m^A provided by AMVER (see Fig. 8b) as the calibration fitness metric in this step. The simulated merchant traffic density map G_m^s is obtained in the following way: we discretize the observed area into a latitude-longitude grid with cell size equal to 1°, similarly to the AMVER grid. We let the simulation run for one simulated year and for each cell, we record the number of transits by any ship. Note that the two maps do not have to be normalized for the computation of the SR index. We used a canonical piracy risk function modeling the piracy risk as a time-independent function of the distance from main pirate anchorages for risk-aware routing.

In the calibration of the O–D matrix, we search for a number of merchant vessels sailing between any two of 20 major world ports. We assume, that the traffic is symmetric, i.e. the flow between two locations is the same in both directions (the O–D matrix is symmetric) and no vessels sail from one location to the same location (the diagonal values are zero); i.e. we can consider only the upper triangular matrix excluding the diagonal. The O–D matrix contains 20 rows and columns, resulting into 180 parameters which are interdependent and together with the non-linear fitness criterion (the SR index) pose a difficult problem to be solved optimally. We thus use local search with restarts to estimate the value of entries in the O–D matrix.

The algorithm is listed in Algorithm 2. The O–D matrix is initialized as a zero matrix. All entries from the upper triangular matrix excluding the diagonal (denoted as *U*(*ODmatrix*) on the 7th line) are selected into a list *entries*, which is randomly shuffled. Then, for each entry from *entries*, the algorithm tries to increment the entry's value by a *step* as long as it leads to the improvement of the SR index. If it is not possible to improve the SR index by incrementing any of the entries above a predefined *max* value, the algorithm tries to improve the SR index by *decrementing* the entries by *step* (switching to decrements on 25th line). The SR index is computed in each step by simulating one year of merchant traffic sampled from the O–D matrix, creating a merchant traffic density map and comparing it to the reference map.

Algorithm 2. O–D matrix calibration

```
1:  input: max,step
2:  SRindex ← ∞
3:  refMap ← AMVER density map
4:  ODmatrix ← U(zeros)
5:  best ← 0
6:  changed ← true
7:  entries ← shuffle(U(ODmatrix))
8:  while changed do
9:    changed ← false
10:   for entry ∈ entries do
11:     while (entry ⩾ 0)&(entry ⩽ max) do
12:       entry ← entry + step
13:       simMap ← simulate(ODmatrix)
14:       SRindex ← getFit(refMap,simMap)
15:       if SRindex < best then
16:         entry ← entry − step
17:         break
18:       else
19:         best ← SRindex
20:         changed ← true
21:       end if
22:     end while
23:   end for
24:   if step > 0 then
25:       step ← − step
26:       changed ← true
27:   end if
28: end while
```

The O–D matrix calibration was repeated 100 times for each risk aversion coefficient $\alpha = \{0, 0.1, \ldots, 1\}$. The best fit was achieved for $\alpha^* = 0.6$; the resulting traffic density map for α^* and the associated SR curve are given in Fig. 8a and c, respectively. Minor discrepancies can be observed around Kenyan and Tanzanian coast and in the Mozambique channel; overall, the fit is very good.

5.4. Complete model calibration

Following the calibration of merchant traffic sub-model, we calibrated the complete model containing all three categories of vessels. The complete model was calibrated to fit the situation in the Indian Ocean in 2011, where there were 181 attacks (source: IMB 2011 reports), from which 28 were hijacks (15.4% hijack success rate). Even though some of the attacks are unreported and thus the IMB 2011 reports are incomplete, it is to our best knowledge the most comprehensive report source. The calibration consisted of the following three steps:

5.4.1. Attack spatial distribution fitting

First, the complete model was calibrated with regards to the spatial distribution of pirate attacks. The number of navy vessels $\#N$ $(0 - 500)$ and pirate's *P-navy-knowledge* were the calibrated variables; the SR curve index (see Section 5.1) between the attack density map produced by the model and the IMB 2011 reports was used as the fitness metrics. The best fit was found for $\#N = 50$ and *P-navy-knowledge* = 0.4. The attack density map produced by the model and the SR curve for the best values of calibrated parameters are given in Fig. 9, along with the reference IMB 2011 reports.

Two directly observable discrepancies between the AgentC model and the situation in 2011 can be observed: the attacks in the AgentC model are concentrated in the East Arabian sea, not spreading to the North. This difference is caused by the fact that the AgentC pirates sailing to the Northern Arabian sea encounter a merchant vessel during their voyage prior reaching the target area and decide to attack this vessel instead of continuing along its original route. Additionally, due to the sparse AgentC merchant traffic in the West Indian Ocean, there are no simulated attacks in that area, the AgentC pirates shift their presence under the southern tip of India, causing another small discrepancy. Even though the fit of the calibrated model on the available data is good, the described discrepancies have to be taken into account when using the model for decision making.

5.4.2. Pirate attack frequency fitting

Second, the complete model was calibrated with regards to the overall number of attacks. The number of pirates $\#P$ (0–5) and pirate's *P-visibility-radius* (5–12 nm) were the calibrated parameters; the fitness metric was a difference between the overall number of attacks produced by the model and the reference real-word value of 181 based on the IMB 2011 reports. The best fit was obtained for $\#P = 2$ and *P-visibility-radius* = 6 nm, producing on average 182 attacks with the standard deviation of 16.1.

5.4.3. Pirate attack success ratio fitting

Finally, the complete model was calibrated with regards to the attack success ratio. Two parameters influencing the outcome of the merchant vessel-pirate interaction were calibrated: *M-alertness* $\in [0,1]$ and *P-base-hijack-probability* ρ_a, $\rho_u \in [0,1]$ (defined in Section 3.6); the fitness metric was the difference to the reference success ratio based on IMB 2011 report statistics, was 0.15. Best fit was obtained for *M-alertness* = 0.5, $\rho_a = 0.2$ and $\rho_u = 0.5$, estimating the probability of a hijack with navy vessels ($\#N = 50$—fixed in the previous calibration phase) to be 0.15 and without any navy vessels to be $p = 0.35$. The probability of a pirate being disrupted by a navy vessel is then 43%, according to our model—this is an example of an insight which cannot be directly inferred from the collected attack reports alone.

5.5. Validation

Face validation was performed repeatedly throughout the model development process. We consulted experts and officials from the industry, government and military, including International Maritime Organization, US Naval Research Lab, US Naval Postgraduate School and several maritime security providers. Feedback received on structural walkthroughs and visualized simulation runs confirmed structural and behavioral plausibility of the proposed model.

Unfortunately, due to lack of data on the behavior of the maritime transportation system under varying circumstances (e.g., the exact number of pirate attack groups and/or deployed naval warships), we were unable to statistically validate the model. The model should not therefore be treated as reliable for quantitative prediction and should be used for gaining qualitative insights only.

6. Case study

We have applied the developed model to several real-world use cases, based in part on discussions with maritime domain stakeholders. Here we present one particular case study focusing on analyzing the possibility of introducing transit corridor system in the Indian Ocean.

The existing *International Recommended Transit Corridor (IRTC)*, established in 2009, has proven—in combination with the deployment of navy vessels—a very effective tool in reducing the number of successful pirates attacks in the Gulf of Aden. The maritime security community has been discussing the possibility of establishing additional corridors in the Indian Ocean, where most pirate activity takes place following pirates' displacement from the Gulf of Aden. In contrast to the Gulf of Aden, which is an elongated, narrow area with a simple bidirectional traffic flow, the Indian Ocean is much larger and crisscrossed, in all directions, by a multitude of traffic flows. This makes the design of an effective corridor system a complicated task.

6.1. Scenarios

We studied the effect of two possible layouts of Indian Ocean corridor systems: (1) single *west-east corridor* channeling the large amount of west- and east-bound traffic (denoted as *Single-IO*), and (2) a more extensive *multi-corridor system* covering all the main traffic flows in the Indian Ocean (denoted as *Multi-IO*). See Fig. 10 for a scheme of corridor layouts. We compared the results with the current setup where no corridors are used in the Indian Ocean (denoted as *None-IO*). The existing IRTC corridor was considered in all three configurations.

In addition to the corridor layout, we were interested in assessing synergies between corridors and other countermeasures, specifically in assessing employing group transit schemes within the corridors and deploying of navy vessels alongside corridors. In addition to the corridor layout, we therefore included the number of deployed navy warships ($\#N = \{20, 30, 40, 50, 60, 80, 100\}$ and the use of group transit (*group-transit* = {*YES, NO*}) as additional study parameters. In order to make the assessment more robust with respect to the variation of future pirate activity, we also included the number of active pirates ($\#P = \{1, 2, 3, 4\}$) as a study parameter.

6.2. Results

The results given are for one year of simulated maritime traffic. Due to probabilistic nature of part of the model, we simulated each configuration for 50 runs and present average values together with standard errors.

The values of the average transit distance (in nautical miles) and average transit duration (in hours) only depend on the layout of the corridor system and amounted to 2153 nm/141 h for the None-IO setup with no corridors in the Indian Ocean, 2162 nm/142 h for the Single-IO and 2213 nm/145 h for the Multi-IO corridor setups. The small difference between different corridor settings is due to the positioning of corridors copying main natural shipping lanes. The traffic is not re-routed

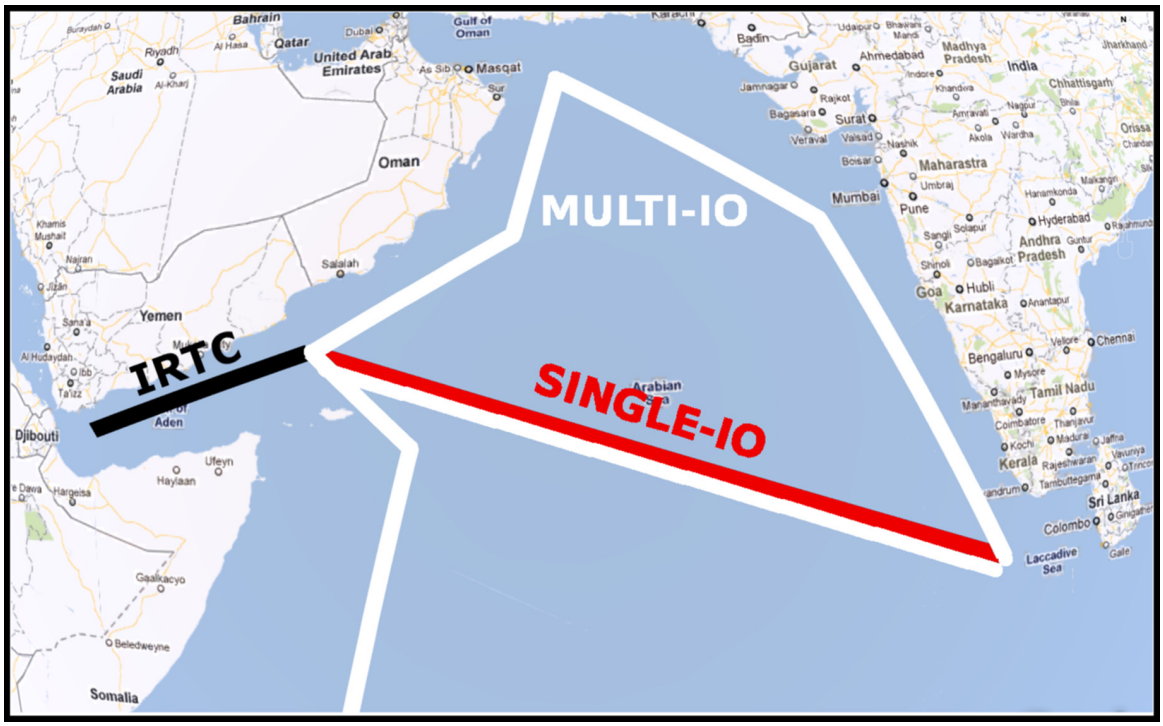


Fig. 10. Corridor layouts for the Indian Ocean corridor system. The Single-IO layout only uses IRTC with the red east–west corridor; the Multi-IO layout utilizes all depicted corridors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

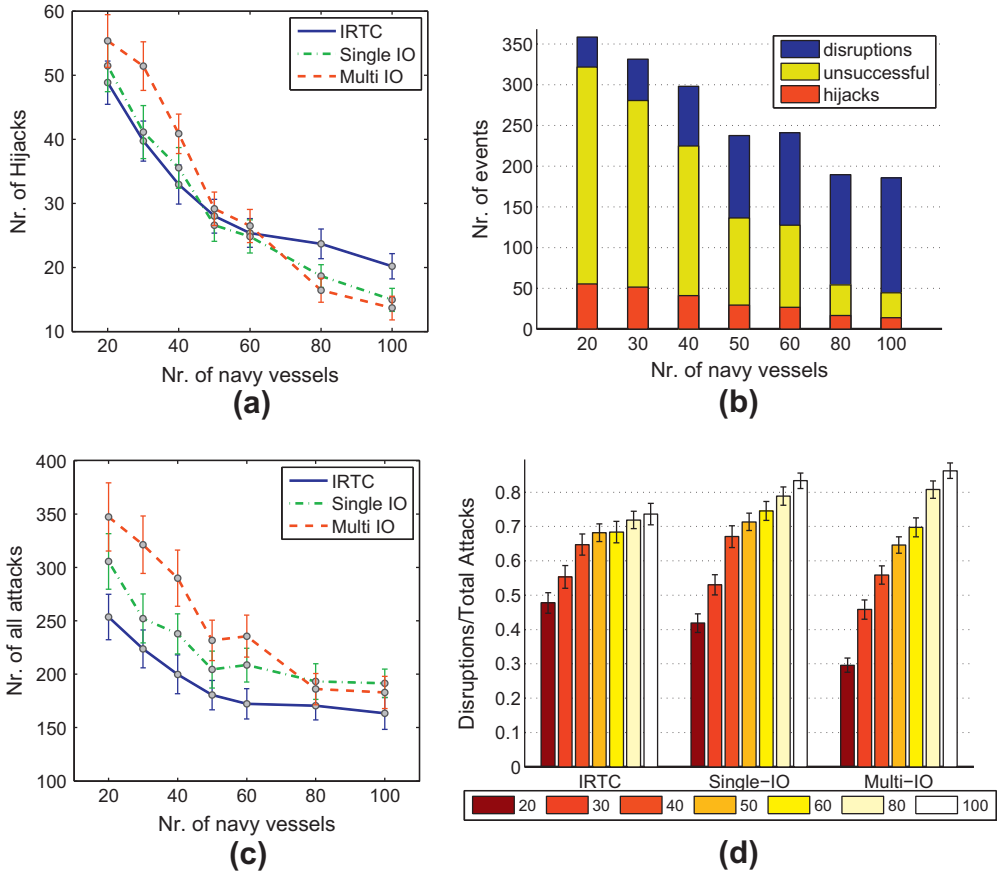


Fig. 11. Results of the Corridor system study. (a) Dependency of the number of hijacks on the corridor system and the number of navy vessels (lower is better). (b) Breakdown of the different attack types for Multi-IO corridor system. (c) Dependency of the number of all attacks on the corridor system (lower is better). (d) Dependency of the ratio of intercepted attacks on the number navy vessels—we can observe boost of navy vessel efficiency by the introduction of extended corridor systems, when enough navy vessels are available (higher is better).

significantly by the introduction of the corridors. Although some of the routes are longer, others can actually be shorter—the risk inside corridors is considered zero[13] and the corridors (especially the south-north corridor) can therefore act as shortcuts to long risk-avoiding detours.

Fig. 11a captures the dependency of the number of hijacks on the number of navy vessels for each corridor system, averaged over different numbers of pirates. As expected, increasing the number of navy vessels decreases the number of both attempted and successful attacks. The reduction in attempted attacks is caused by a denser navy presence, which causes the pirates not to launch attacks when a navy vessel is nearby. The reduction in successful attacks is then caused, additionally, by more frequent attack disruption allowed by the higher density of navy vessels—this can be seen from Fig. 11b which depicts a detailed breakdown of attack outcomes for the Multi-IO corridor system. When the number of navy vessels is increased to a certain level, most of the attacks are successfully disrupted.

What is more interesting is the finding that in order to have positive impact on reducing hijacks, the extended corridor systems (Single-IO and Multi-IO) have to be patrolled by a high number of navy vessels (approximately 70 and more). For fewer than 40 navy vessels, the introduction of the corridors in the Indian Ocean actually worsens transit security (keep in mind that, as suggested in the validation section, quantitative results are only indicative). This is because the better predictability and higher concentration of merchant traffic inside the extended corridors systems makes targeting vessels easier for pirates. This can be seen from Fig. 11c—the number of attempted attacks for the Multi-IO remains higher than for the None-IO setup even for very high numbers of navy vessels. The ratio of disrupted attacks to the number of all attacks (depicted in Fig. 11d) can be seen as navy vessel efficiency. This efficiency rises with increasing the number of deployed navy vessels.

[13] Estimating the value of risk inside a corridor opens another case study by itself—what is the actual risk in an established corridor, given its positioning and a specified number of patrols? Are the merchant vessels incentivized enough to transit through the corridor? AgentC can be used to estimate the risk value by repeatedly running the simulation, compare the number of incidents inside and outside the corridor, quantify the pirate target area preferences, assess the risk and run another simulation with modified risk values, until convergence. Unfortunately, this study is out of scope of this paper.

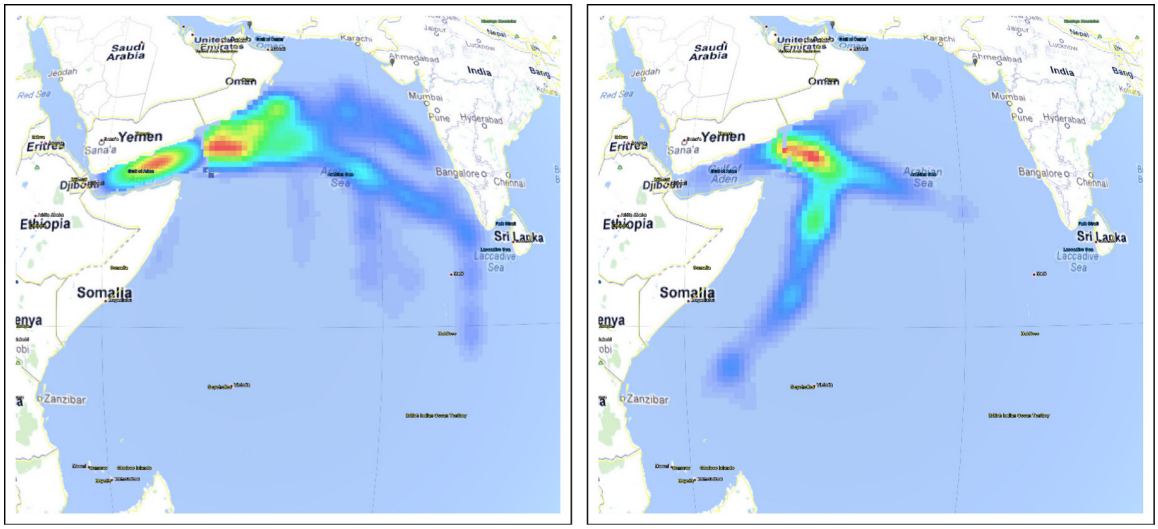


Fig. 12. Geographical distribution of hijacks for the None-IO (left) and Multi-IO (right) corridor system configurations.

More interestingly, the Single-IO and Multi-IO systems lead to higher navy vessel efficiency (though this increase is not enough to counter the increase of attempted attacks). The use/not use of group transit has insignificant effect in the current model—this may change if more sophisticated patrolling strategies which coordinate navy vessels with transit groups are employed.

Finally, in Fig. 12 we compare the geo-spatial distribution of vessel hijacks between None-IO and Multi-IO corridor setups for 50 navy vessels. The distribution of attacks in the Multi-IO corridor system is more concentrated along the corridors and the pirate activity is shifted from the Gulf of Aden and from the East Arabian Sea to the corridor along the Somali coast.

Overall, the results suggests that the positive effect of transit corridors is not directly transferable from the small and narrow Gulf of Aden into the vast Indian Ocean. This is not surprising given the complex nature of the inter-dependencies in the maritime transportation system; it is exactly the kind of conclusions that is difficult to reach without in-depth simulation modeling (e.g., by employing data analysis techniques only).

Key limitation of the study lies in the simple static deployment of navy vessels. More elaborated patrolling and convoy formation strategies could be more effective and allow the extended corridor systems to be successfully patrolled with fewer vessels. The agent-based design and implementation of the simulator makes introduction of such strategies into the model straightforward assuming the description of the strategies can be obtained.

7. Conclusions

We presented AgentC, a simulation model of the maritime transportation system affected by piracy. The model employs agent-based modeling approach—the behavior of the overall system is represented as a composition of thousands of micro-level behaviors of individually simulated vessels. To our knowledge, AgentC is the first model representing deep sea shipping and pirate activity at such a level of detail.

The ability of the model to provide insight into the complex dynamics of piracy-affected waters was demonstrated on a real-world use case of designing transit corridors in the Indian Ocean. Although direct extrapolation of the experience from the Gulf of Aden would suggest that corridors will boost transit security, the simulation of several corridor layouts revealed that this is not necessarily the case and that additional factors play a decisive role. Many other policy decisions can be analyzed using the AgentC model, either out of the box or after small extensions. In fact, applications of the model are not limited to maritime piracy—the merchant traffic sub-model alone is a valuable result and can be used for studying the impact of other factors on maritime shipping (e.g., fuel costs, opening of northern shipping lines, etc.).

The major obstacle to building the model was a severe lack of data on almost all aspects of the behavior of the maritime transportation system. This made proper statistical validation of the model impossible and, consequently, all conclusions have to be interpreted with caution. On the positive side, the work on the model allowed us to clearly identify missing datasets and such information can now serve as a motivation and direction for future maritime data acquisition activities.

Although the AgentC model is the main contribution, the methodology using which the model was developed and calibrated is also a valuable contribution. The application of the agent-based simulation engineering process in the maritime security domain is novel and required several iterations to refine. The trialed-and-tested methodology can be enacted repeatedly to model other maritime transportation scenarios or to improve the existing model when new data become available.

Acknowledgements

This research is funded by the Office of Naval Research (Grant No. N000140910537) and by the Czech Ministry of Education, Youth and Sports (Grant No. LH11051).

References

Barton, D., Stamber, K. 2000. An agent-based microsimulation of critical infrastructure systems, Tech. Rep., Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).
Boult, T., Melter, R.A., Skorina, F., Stojmenovic, I. 1993. G-neighbors. In: Proc. SPIE Conf. on Vision Geometry II, pp. 96–109.
Bourdon, S., Gauthier, Y., Greiss, J. 2007. MATRICS: A Maritime Traffic Simulation, Tech. Rep., Defence R& D Canada.
Bowden, A., Hurlburt, K., Aloyo, E., Marts, C., Lee, A. 2011. The Economic Costs of Maritime Piracy, Tech. Rep., Oceans Beyond Piracy, One Earth Future Foundation.
Bruzzone, A., Massei, M., Madeo, F., Tarone, F., Gunal, M., 2011. Simulating marine asymmetric scenarios for testing different C2 maturity levels. In: Proceedings of the 16th International Command and Control Research and Technology Symposium, pp. 12–23.
Chawdhry, P., 2009. Risk modeling and simulation of airport passenger departures process. In: Proceedings of the 2009 Winter Simulation Conference. IEEE, pp. 2820–2831.
Chung, C., Fabbri, A., 2003. Validation of spatial prediction models for landslide hazard mapping. Natural Hazards 30 (3), 451–472.
Crainic, T.G., Gendreau, M., Potvin, J.-Y., 2009. Intelligent freight-transportation systems: assessment and the contribution of operations research. Transportation Research Part C: Emerging Technologies 17 (6), 541–557.
Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., Wernstedt, F., 2005. An analysis of agent-based approaches to transport logistics. Transportation Research Part C: Emerging Technologies 13 (4), 255–271.
Decraene, J., Anderson, M., Low, M. 2010. Maritime counter-piracy study using agent-based simulations. In: Proceedings of the 2010 Spring Simulation Multiconference, 165.
Earnest, D., Yetiv, S. 2009. Economic globalization and national insecurity: vulnerabilities in the global intermodal shipping network. In: Proceedings of the 50th Annual Convention "Exploring the Past, Anticipating the Future, pp. 1–28.
Ghanmi, A., Boukhtouta, A., Sebbah, S. 2011. Modeling and Simulation of Military Tactical Logistics Distribution.
Hansen, J., Hsu, L., Dykes, J., Dastugue, J., Allard, R., Barron, C., Abramson, M., Russell, S., Mittu, R. 2011. Information domination: dynamically coupling METOC and INTEL for improved guidance for piracy interdiction. In: NRL Review, Naval Research Laboratory, pp. 109–115.
Hasegawa, K., Hata, K., Shioji, M., Niwa, K., Mori, S., Fukuda, H. 2004. Maritime traffic simulation in congested waterways and its applications. In: 4th Conference for New Ship and Marine Technology, Chine, pp. 195–199.
Hidas, P., 2002. Modelling lane changing and merging in microscopic traffic simulation. Transportation Research Part C: Emerging Technologies 10 (5), 351–371.
Hwang, C.L., Masud, A.S.M., et al, 1979. Multiple Objective Decision Making-Methods and Applications. vol. 164. Springer.
Koch, D., 2007. PortSim – a port security simulation and visualization tool. In: Proceedings of 41st Annual IEEE International Carnahan Conference on Security Technology. IEEE, pp. 109–116.
Lauren, M., Stephen, R., 2002. Map-aware non-uniform automata (MANA)–A New Zealand approach to scenario modelling. Journal of Battlefield Technology 5, 27–31.
Norstad, I., Fagerholt, K., Laporte, G., 2011. Tramp ship routing and scheduling with speed optimization. Transportation Research Part C: Emerging Technologies 19 (5), 853–865.
Novák, P., Komenda, A., Lisý, V., Bošanský, B., Pěchouček, M., et al. 2012. Tactical operations of multi-robot teams in urban warfare. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, IFAAMAS, pp. 1473–1474.
Onuoha, F.C., 2010. Piracy and Maritime Security off the Horn of Africa: connections, causes, and concerns. African Security 3 (4), 191–215.
Øvstebø, B.O., Hvattum, L.M., Fagerholt, K., 2011. Routing and scheduling of RoRo ships with stowage constraints. Transportation Research Part C: Emerging Technologies 19 (6), 1225–1242.
Russell, S., Norvig, P., Davis, E., Russell, S., Russell, S., 2010. Artificial Intelligence: A Modern Approach. Prentice Hall, NJ.
Sahr, K., White, D., Kimerling, A.J., 2003. Geodesic discrete global grid systems. Cartography and Geographic Information Science 30 (2), 121–134.
Slootmaker, L. 2011. Countering Piracy with the Next-Generation Piracy Performance Surface Model (Master thesis), Tech. Rep., NPS, Monterey California.
Tambe, M., 2011. Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. Cambridge University Press.
Tang, J., Alam, S., Lokan, C., Abbass, H., 2012. A multi-objective approach for Dynamic Airspace Sectorization using agent based and geometric models. Transportation Research Part C: Emerging Technologies 21 (1), 89–121.
Tsilis, T., 2011. Counter-Piracy Escort Operations in the Gulf of Aden (Master thesis), Tech. Rep., NPS, Monterey California.
Van Dyke Parunak, H., Savit, R., Riolo, R., 1998. Agent-based modeling vs. equation-based modeling: a case study and users guide. In: Proceedings of the 1998 Workshop on Multi-Agent Systems and Agent-Based Simulation. Springer, pp. 277–283.
Waraich, R.A., Galus, M.D., Dobler, C., Balmer, M., Andersson, G., Axhausen, K.W., 2013. Plug-in hybrid electric vehicles and smart grids: investigations based on a microsimulation. Transportation Research Part C: Emerging Technologies 28, 74–86.

Appendix B

Modular Framework for Simulation Modelling of Interaction-Rich Transport Systems

M. Jakob and Z. Moler. Modular framework for simulation modelling of interaction-rich transport systems. In *16th IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2152–2159, 2013.

Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, October 6-9, 2013

Modular Framework for Simulation Modelling of Interaction-Rich Transport Systems

Michal Jakob[1] and Zbyněk Moler[1]

Abstract— **The increasing pervasiveness of information and communication technology (ICT) in transport systems changes the requirements on techniques and tools for transport simulation modelling. Novel ICT-powered responsive mobility services, such as real-time on-demand transport, are *interaction-rich* in a sense that they rely on frequent, ad hoc interactions between various entities of the transport system. These interactions have to be properly captured in the model if it is to accurately represent the dynamics of the modelled transport system. Unfortunately, existing modelling tools are not well suited for modelling interaction-rich transport systems. We have therefore developed a novel modular simulation framework designed specifically for modelling transport systems in which ad hoc interactions and decision making play an important role. The framework provides an extensible library of modelling elements based on a unifying ontology of agent-based modelling abstractions, a high-performance discrete-event simulation engine and suite of tools supporting real-world deployment and utilization of implemented models. By fully leveraging the conceptual foundation of multiagent systems, our framework provides flexibility and extensibility that is difficult to achieve by existing approaches. We demonstrate the applicability of the framework on the models of five distinct interaction-rich transport systems.**

I. Introduction

The increasing deployment of ubiquitous location-aware and internet-connected devices is changing the way transport is organized and managed. Novel ICT-powered mobility services, such as real-time on-demand transport, peer-to-peer car sharing or dynamically priced taxis, are on the rise. A common feature of these services is the intensive use of (semi-)automated, electronic communication for coordination, in order to improve the efficiency and convenience and to reduce the financial and environmental costs of the service. In the case of shared collective taxi services, for example, the explicit, real-time coordination between the riders and the service provider allows using fewer vehicles and, consequently, road space compared to when the same demand was served in an uncoordinated fashion. The newly introduced coordination interactions, however, increase the complexity of the transport system and, consequently, make its operation more difficult to analyse and foresee.

Simulation modelling is an established approach for analysing the behaviour of complex socio-technical systems and is therefore also applicable for analysing transport systems employing ICT-powered services. Unfortunately, existing simulation toolkits do not support the simulation of ICT-powered transport systems well – in particular, they lack the support for modelling anytime, ad hoc interactions among the entities of the transport system and the just-in-time decision making required for participating in such interactions. Capturing both well is essential for accurately modelling the behaviour of ICT-powered systems and, in fact, of the wider class of *interaction-rich transport systems*, i.e., systems whose overall behaviour is strongly affected by ad hoc interactions among their constituent entities.

In our work, we aim to remedy this situation by providing a simulation modelling framework, termed *AgentPolis*[1], designed from its inception to support the modelling of interaction-rich transport systems. Key to achieving this objective is the use of the concept of *multiagent systems*[12] as the basis of the framework's design. Multiagent systems capture the interaction-centricity of ICT-powered transport systems very well – putting them in the core of the modelling framework therefore minimizes the structural and behavioural gap between the target interaction-rich system and its model.

In this paper, we present the main results of our research, describing the four pillars of the AgentPolis framework – the ontology of modelling abstractions, library of ready-to-use modelling elements, discrete-event simulation engine and simulation tools – along with our experience of employing the framework to implement models of five distinct instances of interaction-rich transport systems.

II. Related Work

In the last decade, simulation modelling has become an indispensable tool for studying the behaviour of ICT-powered, interaction-rich transport systems. In [8], the authors employed an agent-based simulation, developed completely from scratch, to study operational characteristics of a multimodal transport system integrating scheduled and flexible on-demand services. Demand-responsive transport systems were also studied in [13].

Taxi operations were also evaluated using simulations, both in their standard form (e.g. [4]) or employing a real-time taxi sharing scheme (e.g. [10], [7]). In all three cases, model-specific simulation tools had to be developed and used, with [4] explicitly stating that existing simulation toolkits, including MATSim and SUMO, were not suitable for the task. Another type of transport systems evaluated using simulations are car sharing services. In [3], the authors evaluated a car sharing scheme under real-world conditions

[1]{jakob, moler}@agents.fel.cvut.cz, Agent Technology Center, Dept. of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, Praha, Czech Republic.

[1]The AgentPolis framework can be obtained from http://agentpolis.org.

978-1-4799-2914-613/$31.00 ©2013 IEEE

of a Californian resort community, again employing a simulation tool developed internally from scratch.

A very interesting approach is presented by Wainer in [14]. The author developed a general language for describing simulation models that allows decoupling the model description from the simulation engine used for model execution. The objectives of Wainer's work – flexibility and ability to rapidly develop simulation models – are close to our goals. His approach is, however, based on discrete-event cellular automata and directed towards vehicle-centric low-level traffic simulations.

A common attribute of the majority of simulations of ICT-powered transport systems is that these simulations were developed from scratch using general-purpose programming languages (most often C++ or Java). There are exceptions – [5] and [6] used the MATSim simulation framework [1] for evaluating car sharing and collective taxi schemes, respectively. Furthermore, in [11] the authors used the general-purpose AnyLogic simulation toolkit to model a taxi sharing scheme in Lisbon. In all of the above cases, however, model developers faced considerable difficulties expressing and implementing required model behaviour using their chosen toolkit; this resulted in long development times and/or reduced fidelity of implemented models.

III. Background and Motivation

Although there are many differences between services such as collective taxis and car sharing, there are also many elements (e.g. the concept of road networks, vehicles, passenger demand, or coordination protocols) that are similar and can be shared between the models of all such transport systems and services. Judging from the observed low use of general toolkits for the simulation modelling of interaction-rich transport systems, it seems that such similarities have not been sufficiently exploited. We believe – and, as we shall see, this belief has been confirmed by our results so far – that the difficulties in employing general simulation toolkits, and the consequent lack of reuse in modelling interaction-rich transport systems, stems from the fact that existing toolkits do not take into account the multiagent nature of the ICT-powered transport systems sufficiently and, consequently, fail to provide abstractions for modelling such systems in a direct, natural way.

Before explaining how we have solved the problem, let us briefly introduce the very concept of multiagent systems (see e.g. [12] for an in-depth discussion). With an acceptable level of simplification, the *multiagent system* can be defined as a system composed of multiple autonomous entities, termed *agents*, situated in a shared environment. The *environment* represents the physical space surrounding the agents and the agents can interact with it in two ways. First, agents perform *actions* that modify the state of the environment; second, in the opposite direction, agents are informed about the state of the environment through *perceptions*. We assume that the agents are endowed with intelligence that allows the agents to select and execute actions that bring them closer to their goals. However, as the environment is one and the agents are many, the actions of individual agents can mutually interact and produce results that, for better or worse, cannot be achieved by individual agents alone. In addition to implicit interaction through the environment, agents can also interact directly, i.e., bypassing the environment, through message-based *communication*. See Figure 1 for a scheme relating the above concepts in a high-level conceptual model of a multiagent system.

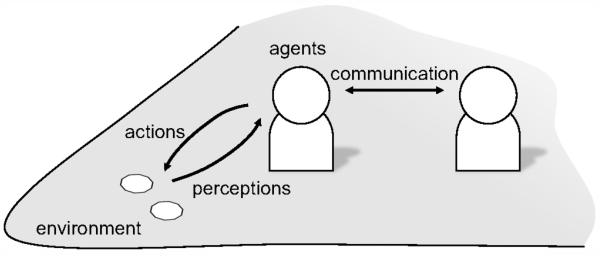


Fig. 1: High-level conceptual model of a multiagent system.

In transport systems, a large number of autonomous entities, such as passengers, drivers or transport operators, pursue their transport-related objectives within the context of a shared and capacity-constrained transport infrastructure. The individual entities interact among themselves and with the transport infrastructure (e.g. queuing on junctions), and produce complex, emergent global behaviours (e.g. congestion). In traditional transport systems, interactions among entities are mostly implicit, mediated by the transport environment. In ICT-powered transport systems, implicit interactions are complemented by explicit ICT-mediated interactions that are often central to driving the overall system behaviour.

Due to their structural and dynamic properties, ICT-powered, interaction-rich transport systems therefore essentially *are* multiagent systems. Consequently, to model them, the (multi)agent-based modelling paradigm should be employed as it offers the most direct conceptual mapping between the model and the system. Unfortunately, existing transport modelling toolkits support the agent-based modelling paradigm only to a limited extent. Although MATSim [1], for example, uses individual-level modelling, it treats individuals as passive data structures whose state can only be updated synchronously by central modules at infrequent, predefined points in time. Despite some practical advantages, such a centralized approach contradicts the nature of multiagent systems and consequently introduces a significant modelling gap – in reality, agents in transport systems make just-in-time decisions asynchronously at different occasions throughout a day, often in reaction to external observations or communication.

To eliminate the modelling gap and issues it creates, our AgentPolis framework employs the agent-based modelling approach fully. AgentPolis does not impose constraints on when and how decision making, activities and interactions can occur in the model, and it is therefore suitable for modelling ICT-powered transport systems with ad hoc interactions and just-in-time decision making.

978-1-4799-2914-613/$31.00 ©2013 IEEE

IV. Framework Overview

The proposed AgentPolis framework provides abstractions, code libraries and software tools for building and using agent-based models of interaction-rich transport systems. More specifically, the framework consists of the following four components:

1) *Modelling abstraction ontology* which provides a unifying set of concepts for expressing agent-based simulation models. The abstractions refine the more general multiagent systems concepts and make them expressible in object-oriented programming languages.
2) *Modelling element library* which contains concrete implementations of the modelling abstractions chosen so as to represent the elements frequently used in real-world transport models.
3) *Simulation engine*, based on the discrete event simulation approach, which provides the runtime functionality for simulating AgentPolis models.
4) *Simulation tools* which support the deployment and use of AgentPolis models in real-world conditions by providing data import, scenario configuration and simulation result analysis and visualization capabilities.

In the following two sections, we describe the framework components in more detail.

V. Modelling Abstractions and Elements

In designing the AgentPolis framework, our aim was to provide a framework that provides maximum ready-to-use transport modelling functionality out of the box while offering enough flexibility to adapt to initially unforeseen requirements. A key tool for achieving this objective was the explicit separation between well-defined modelling abstractions, based on the multiagent conceptual model (see Section III), and concrete modelling elements for building specific application models. By requiring that any modelling element is an instance of one of the modelling abstractions, we enforce design and implementation decisions that promote interoperability among different elements and facilitate addition of new application-specific modelling elements.

The AgentPolis framework currently has eight modelling abstractions (see Figure 2) and several tens of modelling elements – these evolved through several iterations during which the abstractions were used to define concrete modelling elements that were, in turn, used to build specific simulation models.

In the rest of the section, we describe individual modelling abstractions along with the corresponding modelling elements. Due to limited space, we omit some technical details and focus on the features that best convey the overall idea of the framework. Also note that due to circular dependencies between concepts and elements, we sometimes refer to concepts or elements that will only be defined later.

A. Agents

Agents are the central entities of agent-based models and are the main drivers of model dynamics. Somewhat surprisingly, the concept of the agent is only loosely defined in the AgentPolis framework. This is primarily because of the large variation in the behaviour of agents between different models, which makes standardization of agent behaviour difficult and, in fact, counterproductive. Each agent in the AgentPolis framework is therefore only required to have defined its *lifecycle*, which is a top-level activity governing the agent's behaviour.

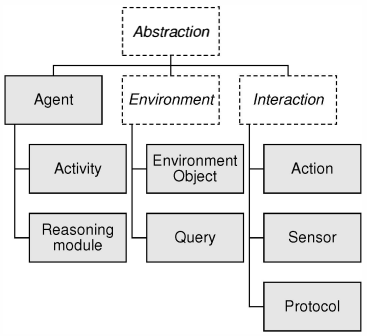


Fig. 2: Modelling abstractions of the AgentPolis framework. The concepts in the white, dashed-outline boxes only provide grouping and are not used as modelling abstractions.

Two predefined lifecycles are nevertheless provided in the framework and can be utilized for defining new agents. The `PTDriver` lifecycle represents the top-level behavioural loop of the agent serving as a public transport vehicle driver; the `UrbanTraveller` lifecycle can be used to implement an agent generating and executing basic activity-driven travel patterns[2].

B. Activities

Activities provide the abstraction for defining agent behaviour. Technically, activities are reactive control structures implementing the logic determining which actions or nested activities the agent executes at a certain point in time or in response to sensor information or messages received from other agents.

For example, the `DriveVehicle` activity moves a vehicle along a predefined route. The route to follow, expressed as a sequence of nodes of an underlying transport network, is given as an input parameter of the activity. The `DriveVehicle` activity then sequentially, for each edge of the transport network, invokes the `MoveVehicle` action to change the location of the vehicle (as well the driver and any passenger inside the vehicle) on the network. After the vehicle reaches the final waypoint, the activity notifies the caller about its successful conclusion and finishes. The list of activities currently provided by the AgentPolis framework is given in Table I.

C. Actions

Actions provide the abstraction for modelling how agents manipulate the environment. Each action defines the logic

[2]Because of their defining role in specifying agent behaviour, we sometimes refer to agents by the name of their assigned lifecycle, e.g., calling an agent employing the `PTDriver` lifecycle as a `PTDriver` agent.

978-1-4799-2914-613/$31.00 ©2013 IEEE

Activity	Description
`Walk`	The agent walks between locations according to a specified journey plan.
`RideInVehicle`	The agent travels as a passenger of an individual transport vehicle according to a journey plan.
`RideOnPT`	The agent travels by public transport according to a journey plan.
`DriveVehicle`	The agent drives a vehicle according to a journey plan.
`ParkVehicle`	The agent parks a vehicle at or near a specified location.
`Wait`	The agent spends a specified time waiting.

TABLE I: Core activities in the AgentPolis framework.

Action	Description
`MoveVehicle`	Moves a vehicle across an edge of the road network, taking possible congestion in the account.
`MoveAgent`	Moves an agents across an edge of the road network.
`TeleportAgent`	Moves an agent instantly to a specified location (used e.g. for initializing agent's position).
`GetInVehicle`	Moves a passenger into a vehicle (the passenger will be linked with the vehicle and move automatically whenever the vehicle moves).
`GetOffVehicle`	Removes a passenger from a vehicle (unlinks the passenger from the vehicle).
`WaitForVehicle`	Waits until a specified vehicle arrives.

TABLE II: Core actions in the AgentPolis framework.

determining action duration and the logic defining which state attributes of which environment objects should be modified as the effect of executing the action.

For example, the `MoveVehicle` action moves a vehicle along a transport network edge by changing the vehicle's location from one transport network node to another, adjacent network node. The `MoveVehicle` action interacts with the queuing logic implemented by the `TransportNetwork` environment object. The state of the `TransportNetwork` object can affect the duration of the `MoveVehicle` action and can even make the action fail if the queue associated with the traversed network edge is full. The list of actions currently provided by the framework is given in Table II.

D. Sensors

Sensors process percepts from the environment and allow agents (and their activities) to be informed about events in the course of simulation, in particular about the changes of the environment state and the execution of action and activities. Together with messages received from other agents, sensor notifications can provide the main triggers for starting, terminating or changing activities executed by agents.

For example, the `PositionUpdate` sensor notification is sent to the `DriveVehicle` activity after the vehicle has reached a new position; after receiving the notification, the `DriveVehicle` activity decides where to move the vehicle next and invokes the next `MoveVehicle` action accordingly. The list of all sensors implemented in the framework is given in Table III.

E. Environment Objects

The environment models the physical context in which agents are situated and perform their activities. In the AgentPolis framework, the environment is decomposed into and, consequently, represented as a collection of *environment objects*. Each environment object represents a fragment of the modelled physical reality and its associated state. The state of an environment object is represented by its attributes and it can only be changed by actions or by the object's internal update logic. Environment objects notify agents through sensors about changes in their state.

Sensor	Description
`PositionUpdated`	Informs about a new position of a specific agent or an environment object.
`NextVehicleLoc.`	Informs about the upcoming next location of a vehicle.
`DrivingFinished`	Informs that a vehicle driver has reached the destination specified by the plan.
`WaitingFinished`	Informs that a specified waiting time has elapsed.
`VehicleArrived`	Informs that a vehicle arrived to a given node.

TABLE III: Core sensors in the AgentPolis framework.

Environ. Object	Description
`TransportNetwork`	A network of roads, railways, cycle paths and/or pedestrian pathways with the associated queuing logic.
`PTStops`	A list of public transport stops or stations.
`Attractor`	A location acting as a destination for trips with specific purpose (i.e. schools, offices, shops etc.).
`Vehicle`	A vehicle that can move along a transport network (car, bus, tram, train etc.).

TABLE IV: Core environment objects in the AgentPolis framework.

For example, the `TransportNetwork` environment object represents a transport network (road, cyclepath, footpath or railway). It consists of a graph of junctions and connecting network segments with associated queues and update logic for modelling congestion. The queue is used by the `MoveVehicle` action to determine how much time a vehicle needs to move along the respective network segment. The list of the environment objects provided by the AgentPolis framework is given in Table IV.

F. Queries

Queries are used by agents to obtain information about the state of the environment. Queries read, filter or aggregate but do not change the state of any environment objects. In contrast to sensors, queries are invoked by the agents (or, typically, by activities)[3]. Although not strictly necessary – calls to queries could be replaced with direct calls to respective environment objects – queries improve encapsulation by providing a layer that hides environment's internal implementation from agents.

For example, given an agent identifier, the `AgentPosition` query returns the position of the agent as the identifier of the transport network node on which the agent is located. The list of queries implemented in the framework is given in Table V.

[3]Queries can therefore be viewed as information *pull* requests, while sensors correspond to information *push* requests.

978-1-4799-2914-613/$31.00 ©2013 IEEE

Query	Description
`AgentPosition`	Returns the current position of an agent or an environment object.
`PTStopPosition`	Returns the position of a (public transport) stop or station.

TABLE V: Core queries in the AgentPolis framework

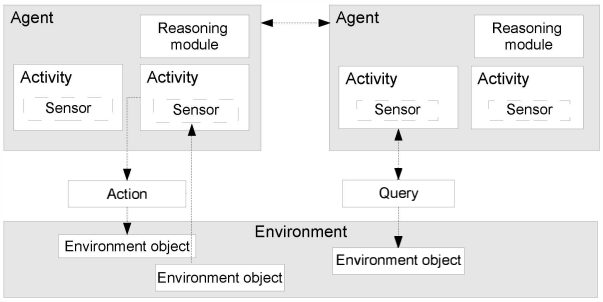


Fig. 3: Simplified architecture of AgentPolis models.

G. Communication Protocols

Communication protocols are the abstraction for modelling inter-agent communication by means of message passing. At the moment, the framework core only provides simple protocols: `1-to-1 messaging` and `1-to-many messaging`. Additional, more complex protocols (e.g., tendering and auctions) have, however, been implemented as part of application-specific models (see SectionVII).

H. Reasoning Modules

As part of their behaviour, agents may need to make decisions that require executing complex algorithms. In the AgentPolis framework, such algorithms can be encapsulated into *reasoning modules* and reused in different activities.

At the moment, the only reasoning module provided in the framework core is the *JourneyPlanner* module encapsulating the fully multimodal journey planner developed in [9]. The module, given an origin and destination location and time constraints, finds a shortest-duration journey plan that can subsequently be executed by agent activities. Additional reasoning modules have been implemented as part of application-specific models (see Section VII).

Figure 3 shows how all modelling abstraction relate to each other in AgentPolis simulation models.

VI. Simulation Engine and Tools

The library of modelling elements and the underlying ontology of modelling abstractions form the fundamental part of the AgentPolis framework. Additional functionality is, however, required for practically using developed models as part of simulation-based evaluation and decision support processes. To this end, the AgentPolis framework comprises software components that support the whole modelling lifecycle from importing real-world data, executing simulation models and analysing and visualizing simulation results.

A. Data Import Tools

To facilitate the incorporation of real-world data into AgentPolis models, the framework provides data importers for converting external datasets into framework's internal data models. At the moment, the framework supports importing data in the *OpenStreetMap (OSM)*[4] and *General Transit Feed Specification (GTFS)*[5] formats, including automated cross-referencing between both formats (e.g., mapping the corresponding public transport stops between OSM and GTFS files). Through the importers information about road, cyclepath and footpath networks, public transport routes and timetables and basic land use can easily be incorporated in AgentPolis models. Files imported by the framework tools are checked for consistency in order to prevent the hard-to-trace errors caused by invalid data during simulation execution.

AgentPolis models can incorporate additional categories of data, such as socio-demographic data or origin-destination matrices representing travel flows. However, as no established standards exist for these data categories, importers for such datasets are scenario-specific and need to be developed or customized for each model.

B. Simulation Engine

The simulation engine for executing AgentPolis simulation models is an essential part of the framework. The AgentPolis framework employs the *discrete event simulation (DES)* approach [2] in which the operation of the target system is modelled as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state of the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation can directly jump in time from one event to the next, which makes it computationally more efficient than the time-stepped approach that is mostly used in transport models.

In AgentPolis models, events provide the low-level causal link between actions, model updates and sensor invocations. Whenever an agent executes an action, the action inserts an event into the event queue; the event has a state update logic attached specifying which environment objects should be updated as the effect of action execution. The state update logic is executed only after the simulation time corresponding to the duration of the action has elapsed. The modification of the environment state caused by the update logic triggers sensor notifications which are received by agents (activities); the agents (activities) can consequently react by invoking further actions, thus closing the model update loop.

The AgentPolis uses the discrete event-queue implementation provided by *Alite*[6], a general purpose lightweight toolkit for building multiagent systems. A screenshot of a running AgentPolis simulation is given in Figure 4.

[4] `http://openstreetmap.org`

[5] `https://developers.google.com/transit/gtfs/reference`

[6] `http://alite.agents.cz`

978-1-4799-2914-613/$31.00 ©2013 IEEE

Fig. 4: High-level view of a running AgentPolis simulation model. Road (black), pedestrian (grey), tram (yellow) and metro (red) networks and UrbanCitizen (green) and PTDriver (yellow) agents are shown. Simulation events are depicted in the overlay window.

C. Result Reporting, Analysis and Visualization Tools

Recording simulation progress and results is a necessary part of simulation execution. AgentPolis provides a customizable logging mechanism employing the Java event bus programming concept that allows detailed recording of low-level simulation events (e.g. the start and end of the execution of activities and actions). From the recorded events, higher-level, aggregate performance metrics can be calculated and visualized using a customizable reporting pipeline. The pipeline is based on the open-source GIS software stack employing the PostGIS[7] spatially enabled database and the OpenGeo[8] interactive geovisualization framework. Powerful aggregation and filtering functions can easily be specified using the spatial extension of the SQL language supported by PostGIS. In addition to OpenGeo, export to Google Earth is also supported and is particularly useful for interactively exploring temporal geospatial data. Together, the above tools allow analysing and browsing simulation results at different spatial and temporal resolution.

VII. Example Models

We have successfully used the AgentPolis framework to implement several simulation models. The models cover a wide range of interaction-rich transport systems that differ in a number of important characteristics, including the type and number of agents, the complexity of agent decision making, the type and number of transport modes present and the complexity of agent-to-agent interactions. The basic information about the implemented models is given in Table VI – below we describe each model in more detail. In Table VII, we then list the main modelling elements used in each of the models.

[7] http://postgis.net
[8] http://opengeo.org

Model	# agents	Types of agents
Multimodal mobility	$10^5 - 10^7$	Urban citizen, PT driver, Driver
Ridesharing	$10^2 - 10^3$	Passenger, Driver, Dispatcher
Dynamic pricing	10^2	Passenger, Driver
Fare inspection	$10^4 - 10^5$	Passenger, Inspector
Parcel logistics	10^2	Dispatcher, Van driver

TABLE VI: List of implemented AgentPolis models with the overall number and the types of agents used.

A. Multimodal Urban Mobility

The multimodal urban mobility model is the most comprehensive and the largest model built using the AgentPolis framework, covering areas up to thousands of square kilometres and simulating populations of up to millions of inhabitants. Employing the activity-centric approach, the model aims to reproduce travel in a multimodal urban transport system. The model is similar in purpose and scope to other activity-based mobility models but it is internally implemented in the fully agent-based way – this gives it the benefits associated with the agent-based approach, in particular the ability to model within-the-day decision making and to include ICT-powered mobility services relying on ad hoc inter-agent interactions in the activity model.

Technically, the model utilizes most of the core AgentPolis modelling elements with the UrbanTraveller lifecycle being the basis of the agents representing the population of the modelled region.

B. Real-time Ridesharing

The real-time ridesharing model has been implemented for studying the performance of ridesharing services under different deployment conditions. The model comprises three types of agents: vehicle drivers (corresponding to drivers of collective taxis, flexible buses or shared private vehicles), passengers of the ridesharing service, and the dispatcher, who matches passengers with drivers and vehicles. While the dispatcher agent is new, the driver and the passenger agents largely reuse the core AgentPolis activities. New, model-specific logic consists of the negotiation protocol used to arrange shared rides and the associated decision logic on the side of participating agents. Extension on lower-level of the model, i.e. actions and sensors, were not required.

In its basic configuration, the ridesharing model only employs hundreds of agents directly participating in the modelled ridesharing service. Thanks to its fully agent-based design, it is, however, possible to combine the ridesharing model with the multimodal urban mobility model and to study interactions between ridesharing services and other mobility modes and services.

C. Auction-based Dynamic Taxi Pricing

The dynamic taxi pricing model has been implemented for studying the effect of auction-based dynamic pricing of taxi services. In contrast to the previous model, the modelled dynamic taxi pricing scheme relies on peer-to-peer interactions and only contains two types of agents: passengers

978-1-4799-2914-613/$31.00 ©2013 IEEE

Abstraction	Element	Multimodal mob.	Ridesharing	Dynamic pricing	Parcel logistics	Fare inspection
Activities	Walk	●	●	●		●
	RideInVehicle	●	●	◇		
	RideOnPT	●				●
	DriveVehicle	●	●		●	●
	ParkVehicle	●				
	Wait	●	●	●	●	●
	DriveTaxi			+		
	PatrolInStation					+
	PatrolInVehicle					+
Env. Objects	TransportNetwork	●	●	●	●	●
	PTStops	●				●
	Attractor	●				
	Vehicle	●	●	●	●	●
	Warehouse				+	
	DeliveryPoint				+	
	VehicleInspectArea					+
	StationInspectArea					+
Actions	MoveVehicle	●	●	●	●	●
	MoveAgent	●	●	●	●	●
	TeleportAgent	●				
	GetInVehicle	●	●	●		●
	GetOffVehicle	●	●	●		●
	WaitForVehicle	●	●	●		●
	RideInTaxi			+		
	TaxiWaitForJob			+		
	LoadParcel				+	
	UnLoadParcel				+	
	UnLoadParcel				+	
	InspectPassengers					+
	ExistInspectArea					+
	EnterInspectArea					+
Sensors	PositionUpdated	●	●	●	●	●
	NextVehicleLoc.	●	●	●	●	●
	DrivingFinished	●				●
	WaitingFinished	●	●	●	●	●
	VehicleArrived	●	●	●		●
	PassengerInSight					+
	InspectorInSight					+
Queries	GetAgentPosition	●	●	●	●	
	GetPTStopPosition	●				
Protocols	1-to-1 Messaging		●	●	●	
	Auction		+			
Reasoning modules	JourneyPlanner	●	●	●		●
	EuclideanAStar				+	
	DistanceTripFinder				+	

TABLE VII: The use of modelling elements in the example AgentPolis models. Core elements printed using normal font; newly added in italics. (● reused core element, ◇ modified core element, + newly added modelling element).

and taxi drivers. Similarly to the ridesharing model, the taxi pricing model reuses a large part of framework's core modelling elements, with the majority of newly developed code concerning the auction protocol and the associated decision logic. In contrast to the ridesharing model, new activities related to travelling by taxi were added. Again, the taxi pricing model can be combined with the multimodal urban mobility model to study mutual interactions.

D. Urban Parcel Logistics

The urban parcel logistics model has been implemented for studying the performance of parcel delivery services. The model comprises two types of agents: van drivers and dispatchers. Because of its focus on the transport of goods rather than people, the model lies outside the main focus of the AgentPolis framework and, consequently, provided an interesting test of the flexibility of the framework's design. The framework has passed the test successfully – although the model required the implementation of several model-specific elements at the environment level, these elements could be expressed using the AgentPolis abstractions. Specifically, we added depots and delivery locations as new types of environment objects together with actions and sensors related to parcel loading and unloading.

E. Public Transport Fare Inspection

Finally, the fare inspection model has been implemented for studying the effectiveness of different strategies for conducting ticket inspection patrols in public transport networks. The model takes travel demand, ticket options and inspector patrol schedules as the input and produces inspection and fare evasion statistics as the output. Different passenger and fare evasion strategies, including the ability of passengers to avoid inspection through learning and communication, are modelled. The model uses two types of agents: passengers and ticket inspectors. The implementation of the model reused a significant portion of the core AgentPolis elements but also required the addition of a number of elements related to performing ticket inspections.

Because of their strong reliance on modelling ad hoc interactions and just-in-time decision making, security models, such as this one, are another important category of interaction-rich transport systems that can benefit from the fully agent-based modelling supported by the AgentPolis framework.

F. Additional Models

We are currently considering the implementation of models of other ICT-powered transport systems, including demand-responsive fleets of driverless cars, smart parking schemes and electrical vehicles sharing services. We believe that in their implementation, similarly to the models already implemented, it would be possible to reuse a large number of AgentPolis core modelling elements and that the extensions and additions required would be expressible using the abstractions of the modelling ontology.

VIII. Discussion

The positive experience with the development of several models confirmed the viability of the fully agent-based approach, and the AgentPolis framework in particular, to modelling interaction-rich transport systems. The five models implemented represent a diverse set of models, each testing the flexibility of the framework in a different way. The framework proved capable of supporting models with a low number of computationally intensive agents (e.g. ridesharing or parcel logistics) as well as models with millions of lightweight agents (multimodal urban mobility). The latter is important because it shows that the higher flexibility of the

978-1-4799-2914-613/$31.00 ©2013 IEEE

fully agent-based approach does not come at the expense of degraded runtime performance of fully agent-based models. Furthermore, despite the diversity of the implemented models, the ratio between the reused and the newly developed code remained good, with the newly developed code mostly focusing on the logic specific to each model. Although in some cases significant extensions were necessary (in particular for parcel logistics and fare inspection models), they were easily accommodated by the framework.

There are still a number of open issues, though. The development of AgentPolis models remains a non-trivial task and requires model developers with good software design and implementation skills. In some cases, there are multiple ways in which a certain behaviour can be expressed in the framework but only some of them allow the model to fully leverage the strengths of the framework and its tools. At the moment, the modeller can refer to the example models for guidance on which abstractions should be employed for which purposes; in the future, we plan to make such guidance explicit in a set of model design patterns.

The above issue is also related to the fact that the simulation logic concerning a certain fragment of the modelled phenomena typically cuts across several modelling abstractions (in particular activities, actions, sensors and environment objects); the implementations of these abstractions thus need to be kept consistent, which is not easy. Although such a mutual dependency problem cannot be fully solved and affects all extensible simulation platforms, there are ways in which the burden on the modeller can be reduced and which we consider for the future versions of the framework. A usual way to address the dependency problem would be to provide a set of well-defined and encapsulated extensions points, which would reduce the need to modify core modelling elements and consequently shield the developer from having to understand their exact interdependencies. This approach would be particularly efficient if the scope of the framework is narrowed. Focusing, e.g., solely on modelling on-demand mobility services (such as ridesharing) would allow fixing the majority of lower-level modelling elements; the model developer would then only implement higher-level model logic governing the arrangement of rides but not their actual execution. In a longer run, the maintainability and extensibility of the framework could be improved by employing more modular programming abstractions – such as traits or lambda expressions – available in some progressive programming languages now and coming to Java in a near future.

The AgentPolis framework currently provides the strongest support for modelling the environment and agent-to-environment interactions. The support for modelling agent behaviour, on the other hand, is relatively basic, with activities and reasoning modules as the only supporting abstractions. This is partly intentional because of the diversity of agent behaviours and the notorious difficulty to provide flexible abstractions for programming general agent behaviour. That said, we plan to improve the support for behaviour modelling by providing simple yet proven behaviour programming abstractions such as finite state machines.

IX. Conclusions

We have developed a modular framework for the implementation, execution and analysis of simulation models of interaction-rich transport systems. The framework fully adopts the agent-based modelling paradigm, which makes it very versatile and capable of modelling systems with complex ad hoc interactions and just-in-time decision making. We have used the framework to implement models of five different transport systems. The positive experience obtained has confirmed the effectiveness of the fully agent-based approach in general, and of the AgentPolis framework in particular, in quickly building models of different kinds of interaction-rich transport systems.

Acknowledgments

This work was funded by the Ministry of Education, Youth and Sports of Czech Republic (grants no. TE01020155 and 7E12065) and by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067).

References

[1] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. W. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of MATSim-T. In *TRB Conference on Innovations in Travel Modeling*, 2008.

[2] J. Banks, J. S. Carson, B. L. Nelson, D. M. Nicol, et al. *Discrete-event system simulation*. Pearson Prentice Hall Upper Saddle River, NJ, 2005.

[3] M. Barth and M. Todd. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7(4):237–259, 1999.

[4] S.-F. Cheng and T. D. Nguyen. Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 14–21, 2011.

[5] F. Ciari, M. Balmer, and K. W. Axhausen. Concepts for large-scale carsharing system: Modeling and evaluation with agent-based approach. In *Transportation Research Board 88th Annual Meeting*, number 09-1888, 2009.

[6] F. Ciari, M. Balmer, and K. W. Axhausen. Large scale use of collective taxis. Technical report, ETH, Eidgenössische Technische Hochschule Zürich, IVT, Institut für Verkehrsplanung und Transportsysteme, 2009.

[7] P. M. d'Orey, R. Fernandes, and M. Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems*, pages 140–146. IEEE, 2012.

[8] M. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188, 2002.

[9] J. Hrnčíř and M. Jakob. Generalised time-dependent graphs for fully multimodal journey planning. In *Proceedings of 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2013.

[10] E. Lioris, G. Cohen, and A. de La Fortelle. Overview of a dynamic evaluation of collective taxi systems providing an optimal performance. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 1110–1115. IEEE, 2010.

[11] L. M. Martinez, G. Correia, and J. Viegas. An agent-based model to assess the impacts of introducing a shared-taxi system in Lisbon (Portugal). In *Proceedings of the 7th International Workshop on Agents in Traffic and Transportation*, 2012.

[12] F. Michel, J. Ferber, A. Drogoul, et al. Multi-agent systems and simulation: a survey from the agents community's perspective. *Multi-Agent Systems: Simulation and Applications*, 2009.

[13] L. Quadrifoglio, M. M. Dessouky, and F. Ordóñez. A simulation study of demand responsive transit system design. *Transportation Research Part A: Policy and Practice*, 42(4):718–737, 2008.

[14] G. Wainer. Developing a software toolkit for urban traffic modeling. *Software: Practice and Experience*, 37(13):1377–1404, 2007.

978-1-4799-2914-613/$31.00 ©2013 IEEE

Appendix C

Mixed-Reality Testbeds for Incremental Development of HART Applications

M. Jakob, M. Pěchouček, M. Čáp, P. Novák, and O. Vaněk. Mixed-reality testbeds for incremental development of HART applications. *IEEE Intelligent Systems*, 27(2):1541–1672, 2012.

Mixed-Reality Testbeds for Incremental Development of HART Applications

Michal Jakob, Michal Pěchouček, Michal Čáp, Peter Novák and Ondřej Vaněk, *Czech Technical University*

An incremental process for developing human-agent-robot applications uses mixed-reality testbeds of varying fidelity and size.

Thanks to technological progress in recent years, autonomous robotic assets now play a role in many real-world endeavors. With continued demand for applications involving a mixture of human, agent, and robot teams (HARTs), there's a growing need for efficient methods and processes to develop such applications. Because of the requirements of efficiency, reliability, and robustness of such systems in real-world conditions, no development methodology will be effective without strong support for realistic evaluation and testing.

In contrast to standalone software systems, the operation of HART applications depends on factors beyond the actual software logic—in particular, on the characteristics of the hardware (sensors, actuators, and communication links), the dynamics of the environment, and the behavior of the humans involved. A reliable assessment requires a *testbed* that approximates these factors with a sufficient level of fidelity. In general, testing an application in the full target configuration (that is, with the complete set of hardware assets and human individuals operating within the target physical environment) provides the most reliable assessment. Unfortunately, such full-configuration tests are expensive in terms of money, time, and resources, and could carry substantial risks—for example, testing a collision avoidance functionality between unmanned aerial vehicles (UAVs). This makes full-configuration testing impractical in the early stages of application development, when developers must perform a lot of evaluation and testing quickly to assess multiple design options.

To reduce costs and risks, application developers can employ simplified testbeds that approximate the target application setup. These testbeds can fully or partially replace the environment, hardware, and human actors with computational models, albeit at the expense of introducing potential assessment errors. Using computational models is common in many areas of engineering, including the development of robotic systems.[1]

1541-1672/12/$31.00 © 2012 IEEE
Published by the IEEE Computer Society

Figure 1. The AgentScout project focuses on tracking mobile targets and patrolling areas and perimeters with teams of unmanned aerial vehicles. Mixed-reality testbeds helped develop the algorithms for coordinating multiple vehicles.

In the case of HART applications, however, determining which parts of the application to approximate with computational models is difficult due to the large number of involved entities and their dependencies. In this article, we address this issue and lay the foundations for an approach that balances assessment cost and accuracy throughout the development process. We propose and formalize the concept of *incremental multilevel mixed-reality development*, which lets us use mixed-reality testbeds of various sizes and virtualization levels in a way that maximizes the effectiveness of HART application development.

This approach arose from our experience with the AgentScout project (see Figure 1), which focuses on developing coordination algorithms for mixed teams of mobile robots, static sensors, and human patrols.[2] Our work dealt with implementing the ability for teams of UAVs to track mobile targets autonomously, with only high-level supervision by a human operator. We have discussed some of the ideas underlying the proposed framework in previous writings about using simulations to accelerate the development of multiagent applications.[3] We also followed this approach in the development of a piracy countermeasure coordination system for the AgentC project, as well as in the process of porting collision-avoidance algorithms to UAVs in the AgentFly project.[4] For other work in the field, see the sidebar "Related Work on Mixed Reality."

Multilevel Mixed-Reality Testbeds

In general, there are different ways to approximate target deployment setup to make application tests faster and less expensive. Because our interest is in multi-entity systems involving several human actors and robotic assets, we consider two principal dimensions in which the approximation can proceed:

- the level of virtualization at which the target setup is represented, and
- the number of entities in the test scenarios.

We could introduce other approximation dimensions as long as they allow us to trade test costs for test accuracy.

Approximation Dimensions

The *virtualization level* denotes how much the target application setup is virtualized in a given testbed configuration—that is, how many parts are replaced with synthetic computational models. At one extreme, the full target setup uses no virtualized entities—only physical hardware platforms and human actors. Starting from the zero-virtualization setup l_0, we replace individual entities of the application with computational models. This process of gradual virtualization gives rise to a sequence of virtualization levels, labeled $l_0, l_1, \ldots, l_n$, in which l_n is a completely virtualized setup with fully synthetic computational representations of system entities. Between the two extremes lie mixed-reality testbeds, such as hardware-in-the-loop simulations or testbeds involving human actors in virtual reality. Within a single testbed configuration, we can assign different virtualization levels to different entities. In most cases, a lower virtualization level facilitates more reliable testing but consumes more time and resources. The core idea underlying our development process is to start at a relatively high system virtualization level and then iteratively decrease it until we reach the target deployment setup.

The other dimension along which we can approximate the target application setup is the number of autonomous entities with which the application is tested. Instead of having the same full number of robots and/or humans as in the full application, we can perform initial development and testing with only a subset of entities. We call the number of entities used the *size* of a testbed configuration.

Reducing the testbed size generally leads to cost savings, especially when physical hardware or human actors are involved. Reducing the number of entities below a certain threshold,

Related Work on Mixed Reality

Virtual reality deals with ways for a human user to observe and interact with nonexistent virtual worlds.[1] In mixed reality, the agent observes a world that is partly real and partly virtual. The concept of a *reality-virtuality continuum* embodies a continuous scale, ranging from fully real to fully virtual worlds.[2] Between the two extremes stands *mixed reality*, typically implemented either as *augmented reality* (in which perception of the real world is augmented with virtual objects) or *augmented virtuality* (the virtual world is augmented with elements of physical reality). Current research in mixed reality mostly concerns interface devices that let a human user observe and interact with mixed-reality worlds.

Developers of control programs for autonomous robotic systems routinely use sophisticated simulators to test their programs prior to deployment on target hardware. To further increase the fidelity of such testing, they can include one or more physical hardware assets in the simulation. Such simulations are, depending on the context, termed *hardware-in-the-loop simulations* (HILs), *hybrid simulations* (HS), or *mixed-reality simulations* (MRSs).[3–5]

The idea of evaluating control algorithms for multirobot systems in environments that mix both real and simulated entities is over 20 years old. An initial attempt dealing with simulation of industrial robots was published in 1989.[6] More recently, researchers have used mixed-reality simulations in the development of autonomous robotic assets. Ian Chen and his colleagues introduced a mixed-reality simulation library for the Gazebo 3D mobile robot simulator.[5] Using this library, a real hardware robot can interact with the Gazebo simulated world, which can both augment the real robot's environment with virtual objects and provide visual feedback on the state of the robot's perception through 3D visualization. Others have used hybrid and hardware-in-the-loop simulations for autonomous underwater vehicles.

References

1. G. Burdea and P. Coiffet, "Virtual Reality Technology," *Presence: Teleoperators and Virtual Environments*, vol. 12, no. 6, 2003, pp. 663–664.
2. P. Milgram et al., "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum," *Proc. Telemanipulator and Telepresence Technologies*, 1994, pp. 282–292.
3. D. Lane et al., "Interoperability and Synchronization of Distributed Hardware-in-the-Loop Simulation for Underwater Robot Development: Issues and Experiments," *Proc. Int'l Conf. Robotics and Automation* (ICRA 01), IEEE, 2001, pp. 909–914.
4. B. Davis, P. Patron, and D. Lane, "An Augmented Reality Architecture for the Creation of Hardware-in-the-Loop & Hybrid Simulation Test Scenarios for Unmanned Underwater Vehicles," *Proc. OCEANS 2007*, 2007, pp. 1–6.
5. I.Y.H. Chen, B. MacDonald, and B. Wünsche, "Mixed Reality Simulation for Mobile Robots," *Proc. Int'l Conf. Robotics and Automation* (ICRA 09), IEEE, 2009, pp. 232–237.
6. F. Cao and B. Shepherd, "Mimic: A Robot Planning Environment Integrating Real and Simulated Worlds," *Proc. Int'l Symp. Intelligent Control*, IEEE, 1989, pp. 459–464.

however, can undermine the ability to test collective behavior properties.

Testbed Configurations and Fidelity

To specify the distribution of virtualization levels in compact form, we define *testbed configuration* σ as a vector $\sigma = (s_0, s_1, \ldots, s_n)$, where n is the number of virtualization levels, and s is the number of entities modeled at the corresponding virtualization level. The testbed configuration's size (the number of entities used) then corresponds to $\eta(\sigma) = \sum_{i=0}^{n} s_i$. *Target configuration* σ_τ represents the entire target system, with no virtualization. Typically, σ_τ will be of the form $\sigma_\tau = (k, 0, \ldots, 0)$ for some $k \geq 0$.

We can visualize the space of all possible testbed configurations in a plane, as shown in Figure 2. The horizontal axis denotes the testbed configuration sizes; the vertical axis represents the aggregate virtualization level, defined as a weighted average of virtualization levels over all the entities in the given testbed configuration.

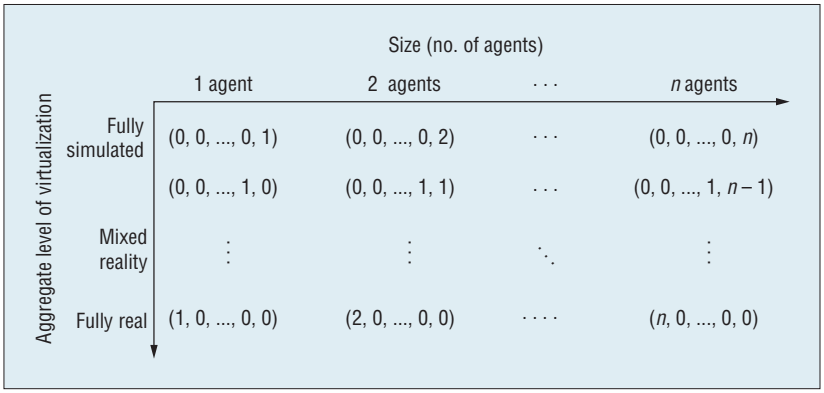


Figure 2. The space of testbed configurations with different combinations of testbed size and level of virtualization. The target configuration corresponds to the bottom right corner.

Unless the testbed corresponds to the target configuration σ_τ, the application's test performance can differ from its performance in the target configuration. To capture this difference, we introduce the concepts of *testbed error* and *testbed fidelity*. The testbed error $\varepsilon(\sigma) \in [0, \infty)$ is the distance between state space execution traces produced using the given testbed configuration and those using the target configuration, averaged over all possible application runs. The testbed

fidelity $\varphi(\sigma) = 1$ if $\varepsilon(\sigma) = 0$—that is, when a testbed configuration fully replicates the behavior of the target setup. Otherwise, $\varphi(\sigma) = \tanh[1/\varepsilon(\sigma)]$.

The higher the fidelity of a testbed, the more accurate the assessments obtained on it. We can use a different sigmoid function for the fidelity calculation, as long as it maps the testbed error in the (0, 1] interval.

Comparing Testbeds

The core of the proposed approach to HART application development is iterative evaluation in gradually more and more realistic setups. Expressed in terms of testbed configurations, the incremental process should use a sequence of testbed configurations with increasing fidelity. Unfortunately, in practice, determining such a sequence isn't directly feasible, because we can't determine testbed fidelity without performing tests on the full target configuration and comparing results.

Instead, we use an approximate fidelity ordering based on the following assumption, which is valid in most domains: a testbed configuration σ' is expected to have higher fidelity than σ if

- testbed size increases while the virtualization level does not increase, or
- the virtualization level decreases while the testbed size does not decrease.

More formally, we define the *approximate fidelity ordering relation* $\succ$ of testbed configurations as follows: for configurations $\sigma = (s_0, \ldots, s_n)$ and $\sigma'' = (s'_0, \ldots, s'_n)$, we denote $\sigma' \succ \sigma$ if at least one of the following holds:

- there exists a testbed configuration σ'' such that $\sigma' > \sigma'' > \sigma$ (transitivity);
- for all i, $s'_i \geq s_i$, and there exists j such that $s'_j > s_j$ and thus $\eta(\sigma) > \eta(\sigma)$; or
- $\eta(\sigma') = \eta(\sigma)$, and there exists j such that $s'_j > s_j$, and for all $i < j$, $s'_i \geq s_i$.

We use the approximate testbed fidelity ordering to navigate the space of testbed configurations during the development process.

Virtualization Levels in HART Applications

For applications involving human-agent-robot teams, we can use specific virtualization techniques. For robotic hardware entities, these include the following, listed in decreasing order of level of virtualization (L):

- *Fully simulated hardware* (L_{FS}). A hardware asset is replaced by a fully synthetic computational model, such as an out-of-the-box robotic simulator like Gazebo, able to simulate the physical and electronic properties of several different robotic platforms.
- *Hardware-in-the-loop* (L_{HIL}). Even the most sophisticated robotic simulators don't capture all the phenomena that could arise in a real hardware asset. We can use a hardware-in-the-loop setup to increase the testbed's hardware fidelity. We test a hardware asset in a laboratory setting, with sensory input provided by a simulator, and actuator signals controlling a simulated physical model. This approach is often used to verify the function of hardware platform electronic and communication subcomponents and is therefore useful in single-robot scenarios.
- *Augmented reality or hybrid simulation* (L_{HS}). As a next step toward the full hardware testbed, we can use an augmented reality or hybrid simulation. Hardware assets operate in the target physical environment, but the simulator augments their sensory inputs with objects that exist only in the simulation. This approach is particularly useful in multiagent scenarios where we want to test interaction between multiple robots but fewer than the target number of physical robots are available.
- *Full hardware* (L_0). Robot assets are represented by target robotic platforms operating and interacting in the target physical environment.

Virtualization techniques involving human entities include the following, again listed in decreasing order of virtualization:

- *Computational behavior models* (L_{BM}). We substitute computational behavior models for human actors. We can construct the models manually according to an expert input, or they can be learned automatically from past observations of human behavior. We can employ different models of human decision-making, such as prospect theory, bounded rationality, or quantal-response equilibrium.
- *Data feeds about human behavior* (L_{DF}). The testbed replays data feeds of past real human behavior. This type of virtualization typically provides only a unidirectional link between the human actors and the rest of the application; in other words, the humans cannot react to the output of the application.
- *Virtual reality* (L_{VR}). Real humans are involved, but instead of operating directly in the target environment, they work in its virtual reality approximation. This lets us test aspects of human behavior that are hard to capture with computational models and facilitates testing of phenomena such as bounded rationality, irrational behavior, or the effect on decision-making of immediate mental attitudes, such as stress, fear, anger, or joy.

- *Simulated human-machine interaction* (L_{SI}). Real humans operate in the target environment, but some parts of their interaction with other entities remain simulated. This lets us test coordination algorithms even with incomplete hardware capability (sensory or other). The setup provides a human-oriented equivalent of hybrid simulations from the perspective of hardware assets.
- *Full human involvement* (L_0). Real humans act in the target environment.

Multilevel Incremental Development

The main reason for the concept of a testbed configuration is to provide a framework for describing iterative strategies for HART application development. A key idea in determining such strategies is to use a sequence of testbeds with different test accuracy and cost so that each iteration can provide the maximum feedback on design and implementation choices.

Cost

To explain this idea more accurately, we introduce the notion of *iteration cost,* expressed as $cost(\sigma_1, \sigma_2)$, which represents the total cost for getting from an application that works correctly on testbed configuration σ_1 to an application that works correctly on configuration σ_2. In general, the overall iteration cost breaks down into.

- the *testbed cost* of providing a testbed with configuration σ_2;
- the *development cost* of modifying the application logic to work correctly on testbed σ_2; and
- the *test cost* of verifying that the modified application works correctly on testbed σ_2.

Iteration Strategy

The ultimate challenge in our proposed methodology is to find the optimal iteration strategy through the space of testbed configurations—that is, a sequence of configurations $\sigma_0, \sigma_1, \ldots, \sigma_n$ such that $\sigma_n = \sigma_\tau$ and $\sum_{i=1}^{n} \text{cost}(\sigma_{i-1}, \sigma_i)$ is minimal.

Unfortunately, except for trivial cases, determining the optimal strategy a priori is not feasible in real-world cases because it depends on the specifics of each application and on the uncertainty of cost estimates. However, assuming that iterations are faster on more highly virtualized testbeds, a reasonable strategy is to start with those, even though they might have lower fidelity. After that, as uncertainty about application design and underlying logic decreases, development should move toward higher-fidelity testbeds, even though they're likely to have higher test cost and therefore allow fewer tests. The following algorithm illustrates such an iteration strategy:

1. Choose an arbitrary starting configuration σ.
2. Develop a testbed with the configuration σ.
3. Develop, modify, and debug the application until it works correctly on σ.
4. If $\sigma = \sigma_\tau$, end.
5. If not, choose another configuration σ' such that $\sigma' \succ \sigma$, and start again from step 2 with σ'.

Typical scenarios involving several virtualization levels and multirobot systems will offer many ways to construct the sequence of testbed configurations. For instance, the developer can choose whether to first scale the algorithms with respect to the number of simulated robots and only then start to port the system to real hardware, or the other way around. Either way, the core of the iterative-development strategy should remain: to gradually refine the system setup so as to approach the target deployment scenario.

In general, the space of testbed configurations has as many dimensions as there are virtualizations levels, making the number of possible iteration strategies huge. A better understanding of the structure and properties of the error, fidelity ordering, and iteration cost functions is therefore essential for deriving intuitive or possibly even more formal rules for determining the iteration strategy.

Iterative Development of a Multi-UAV Tracking Application

We have used our approach to develop a coordination mechanism for a team of UAVs cooperatively tracking a number of humans in an urban environment. Besides a fully simulated setup (L_{FS}) and a fully physical target setup (L_0), we also considered an intermediate augmented-reality setup (L_{HS}). The intermediate setup was necessary because we only had two hardware platforms available—Unicorn UAVs from Procerus Technologies—whereas we had to test the coordination mechanism with larger teams of UAVs.

Starting from a fully simulated setup, there are numerous ways to traverse the space of testbed configurations; Figure 3 depicts a fragment of the configuration space. The underlying directed graph connecting the configurations corresponds to the approximate fidelity-ordering relation described earlier. The two development strategies shown correspond to the following two extreme cases.

Following the first strategy, we would first scale the coordination algorithms to the target number of simulated UAVs in a fully simulated

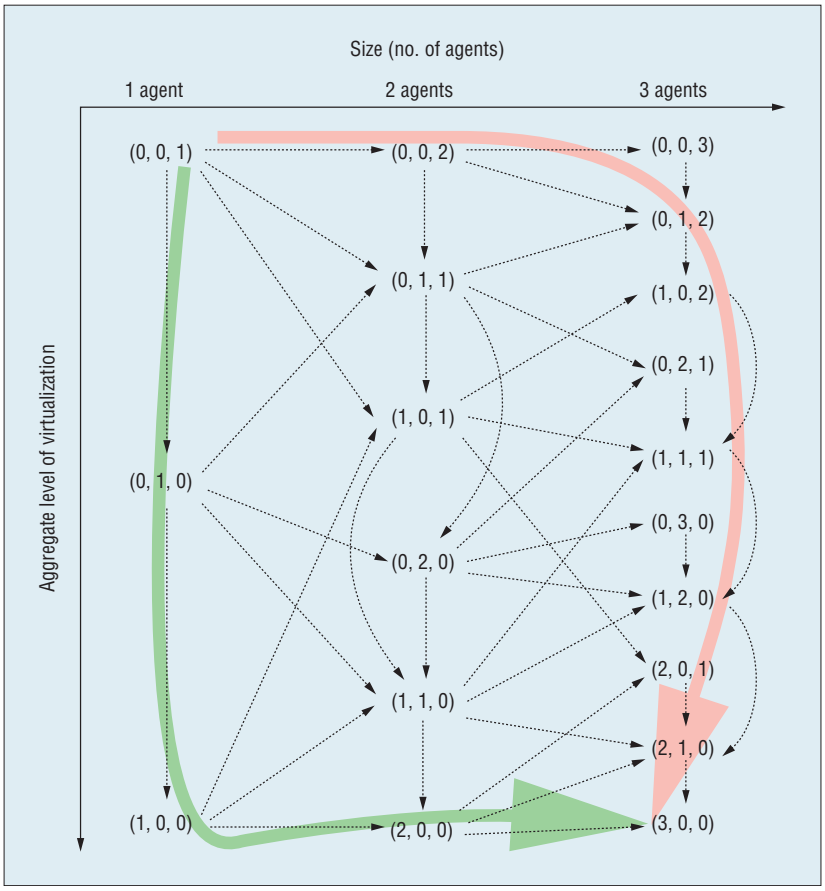


Figure 3. Space of testbed configurations for the multi-UAV tracking application. The arrows describe two possible development strategies.

environment. Then we would port the control code running on individual UAVs from the simulation to physical UAVs, one by one, embedded in an augmented-reality environment. At each step, we would go through the full port-extend or adapt-test-evaluate cycle until we reach the desired performance level. This way, development would eventually result in all aircraft control agents deployed and working correctly on target hardware UAV platforms with augmented sensory input feeds. In the final round of iterations, we would disconnect individual UAVs from the augmented-reality environment and place them in the target physical environment.

With the other strategy, we would initially work with only one UAV. Immediately after getting its control logic working correctly in the full simulation, we would port it to the UAV embedded in the augmented reality and then, after adapting it to work correctly, to a physical UAV. Once the system is working correctly on a single hardware UAV in the physical environment, we would add additional UAVs.

The choice between the two strategies typically depends on which is more uncertain: realizing collective team behavior or deploying the control logic on the target hardware platform. In this particular case, we chose the first approach.

Similarly, human entities that are the tracked objects can also be involved at different virtualization levels. Initially, we can represent all human subjects with computational behavior models. In the second step, we can approximate some human actors in the minimal virtual reality settings (for example, using a joystick with a GUI). The third step would involve real humans moving in the target physical environment but with simplified interaction with the robotic assets.

The approach described is just a first step toward a comprehensive methodology for incrementally developing HART applications using mixed-reality testbeds. Further research is needed to better understand how different combinations of testbed sizes and virtualization levels affect testbed fidelity and individual components of iteration cost. Although it is likely that strong, prescriptive iterative-development guidelines can be found only for specific subcategories of HART applications, the proposed common conceptual framework allows for the comparison of different guidelines and the transfer of methodological knowledge across domains.

Acknowledgments

This article originated during the sabbatical of Michal Pěchouček at the University of Southern California supported by the Fulbright Commission. Furthermore, the presented work was supported by US Army Communications-Electronics Research, Development and Engineering Center grants W911NF-10-1-0112 and W911NF-11-1-0252, Office of Naval Research grant N000140910537, and the Grant Agency of the Czech Technical University in Prague grant SGS10/189/OHK3/2T/13.

References

1. N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-robot Simulator," *Proc. Int'l. Conf. Intelligent Robots*

and Systems (IROS 04), IEEE, 2004, pp. 2149–2154.
2. J. Vokřínek, P. Novák, and A. Komenda, "Ground Tactical Mission Support by Multi-agent Control of UAV Operations," *Proc. 5th Int'l. Conf. Industrial Applications of Holonic and Multi-agent Systems* (HoloMAS 11), LNCS 6599, Springer, 2011, pp. 225–234.
3. M. Pěchouček, M. Jakob, and P. Novák, "Towards Simulation-Aided Design of Multi-agent Systems," *Proc. 8th Int'l Workshop Programming Multi-agent Systems* (PROMAS 10), LNCS 6599, Springer, 2012.
4. M. Jakob, O. Vaněk, and M. Pěchouček, "Using Agents to Improve International Maritime Transport Security," *IEEE Intelligent Systems*, vol. 26, no. 1, 2011, pp. 90–96.

THE AUTHORS

Michal Jakob is a senior researcher in the Agent Technology Center at the Czech Technical University in Prague. His research interests include agent-based simulation, computational game theory, and multiagent systems. Jakob has a PhD in artificial intelligence and biocybernetics from the Czech Technical University in Prague. Contact him at michal.jakob@agents.fel.cvut.cz.

Michal Pěchouček is a full professor in cybernetics and the head of the Agent Technology Center at the Czech Technical University in Prague, where he is also the deputy head of the Department of Computer Science and Engineering and head of the Agent Technology. His research interests include multiagent simulation and modeling; coordination; social-knowledge representation; multiagent planning; multiagent prototypes and testbeds; and applications of agent-based computing into security-related applications, unmanned-aerial-vehicle robotic coordination, and air traffic control. Pěchouček has a PhD in artificial intelligence and cybernetics from the Czech Technical University in Prague. Contact him at michal.pechoucek@agents.fel.cvut.cz.

Michal Čáp is a researcher and a PhD student in the Agent Technology Center at the Czech Technical University in Prague. His research interests include multiagent systems, especially agent-oriented programming, agent-based simulations, and distributed coordination and design of multirobot systems. Čáp has an MSc in agent technology from Utrecht University. Contact him at michal.cap@agents.fel.cvut.cz.

Peter Novák is a postdoctoral researcher in the Agent Technology Center at the Czech Technical University in Prague. His research interests include interaction of cognitive agents with dynamic environments, especially cognitive robotics, multiagent planning, and multiagent coordination. Novák has a PhD in computer science from the Clausthal University of Technology. Contact him at peter.novak@agents.fel.cvut.cz.

Ondřej Vaněk is a researcher and PhD student in the Agent Technology Center at the Czech Technical University in Prague. His research interests include multiagent simulations and application of cooperative and noncooperative game theory to securing complex critical infrastructures. Vaněk has an MSc in technical cybernetics from the Czech Technical University in Prague. Contact him at ondrej.vanek@agents.fel.cvut.cz.

Appendix D

Generalised Time-Dependent Graphs for Fully Multimodal Journey Planning

J. Hrnčíř and M. Jakob. Generalised time-dependent graphs for fully multimodal journey planning. In *16th IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2138–2145, 2013.

Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, October 6-9, 2013

Generalised Time-Dependent Graphs for Fully Multimodal Journey Planning

Jan Hrnčíř[1] and Michal Jakob[1]

***Abstract*— We solve the fully multimodal journey planning problem, in which journey plans can employ any combination of scheduled public transport (e.g., bus, tram and underground), individual (e.g., walk, bike, shared bike and car), and on-demand (e.g., taxi) transport modes. Our solution is based on a generalised time-dependent graph that allows representing the fully multimodal earliest arrival problem as a standard graph search problem and consequently using general shortest path algorithms to solve it. In addition, to allow users to express their journey planning preferences and to speed up the search process, flexible journey plan templates can be used in our approach to restrict the transport modes and mode combinations permitted in generated journey plans. We have evaluated our solution on a real-world transport network of the city of Helsinki and achieved practically usable search runtimes in the range of hundreds of milliseconds.**

I. Introduction

The growing number of transport options available in modern cities raises the importance of tools that support travellers in finding journey itineraries that make the best use of available transport services while respecting traveller's individual needs. Existing journey planners fulfil such requirements only to a limited degree, in particular as they often only consider a certain subset of transport modes and their combinations, and as they only provide limited ways for users to express their preferences.

The journey planning problem is most often formalised as the *earliest arrival problem (EAP)*, i.e., the problem of finding the earliest arrival at a destination given a departure date and time from an origin. The earliest arrival problem has been widely studied and numerous algorithms and speed-up techniques exist for solving it on road network graphs and networks of public transport (PT) services. However, very limited work has been done on solving the earliest arrival problem for journey plans allowing general combinations of individual and public transport modes, the work of Horn [7] and Yu and Lu [13] being notable exceptions.

In this paper, we aim to address this gap. More specifically, we focus on solving the *fully multimodal* variant of the EAP. We use the term fully multimodal in order to stress that we consider modes and combinations thereof that go beyond what is supported in existing multimodal journey planners. In our approach, a journey can consist of any combination of scheduled PT modes (e.g., bus, tram and underground), individual modes (e.g., walk, bike, shared bike and car), and on-demand (e.g., taxi) modes.

[1]{hrncir,jakob}@agents.fel.cvut.cz, Agent Technology Center, Dept. of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Praha, Czech Republic

We adopt a representation-centric approach to solving the fully multimodal EAP. Thus, instead of providing complex, purpose-specific journey planning algorithms, we introduce a *generalised time-dependent (GTD) graph* that allows representing the fully multimodal EAP as a standard graph search problem and consequently use general shortest path algorithms to solve it. We treat the problem in a deterministic setting assuming no uncertainty in any of the attributes of the planning graph.

Along with the GTD graph representation, we also introduce the concept of *journey plan templates*. Journey plan templates provide a powerful way of parameterising the operation of the planner and allow the user or the administrator of the journey planner to obtain plans that best meet their constraints and preferences. In addition, the templates constrain the search space and therefore speed up journey planning.

II. Related Work

As already indicated, the earliest arrival problem is a widely studied problem when considered separately for planning on road networks and for planning on networks of scheduled PT services. Existing work covers the whole spectrum from formal models of the problem, through solution algorithms up to practical consumer-oriented planning tools and services.

The road network variant of the EAP typically employs the direct graph representation of the road network. The road network is represented as a weighted directed graph $G = (V, E, \rho)$ where the set of nodes V represents junctions and the set of edges E represents roads. Each edge $(u, v) \in E$ is assigned a weight $\rho((u, v))$ specifying the time needed to travel across this edge. The road network graph is very sparse, almost planar, and usually has hierarchical properties. These properties are used to enhance basic shortest-path algorithms, such as A*, with speed-up techniques that accelerate graph search. The best known speed-up techniques include *SHARC* [2], *Landmark A** (ALT) [6], *highway hierarchies* [11], and *transit-node routing* [1].

For the scheduled PT variant of the EAP, there are two main ways to represent public transport timetables as a search graph. In the *time-expanded approach* [9], each event at a stop, e.g., the departure of a train, is modelled as a node in the graph; in the *time-dependent approach* [3], the graph only contains one node for each stop. To accelerate the search process, many speed-up techniques for basic shortest-path algorithm, e.g., Dijkstra's algorithm, have been proposed, including the *multi-level graph* approach [12], *access-node*

978-1-4799-2914-613/$31.00 ©2013 IEEE

routing [4], and *core-ALT* [8]. Many of these techniques have been implemented as part of public online travel planning services.

In contrast to the EAP for road networks and for networks of scheduled PT services, only very limited research exists on the fully multimodal variant of the EAP. One of few exceptions is the planner proposed by Horn [7] which supports combinations of scheduled PT and on-demand transport services. A limitation of the Horn's approach is that the on-demand mode can only appear as the first or the last non-walk leg of a journey, i.e., the on-demand mode can only serve as a feeder service. The second attempt at solving the fully multimodal EAP is provided by Yu and Lu [13] who use a genetic algorithm to construct the sequence of transport modes in a journey plan. In their experiments, Yu and Lu permit walk, bus, underground, and taxi modes. However, the individual modes of transport (bike, shared bike and car) are not used.

Despite the limited research on the fully multimodal EAP, several online services exist capable of planning journeys employing non-trivial combinations of transport modes. For example, the AnachB[1] planner supports the combination of car and scheduled PT services. A major weakness is that the parking place (P+R) is not chosen optimally by the planner but needs to be selected manually by the user. The OpenTripPlanner[2] supports the combination of walking and riding a shared bike borrowed and later returned to one of the many city's bike sharing stations. However, the technical approaches behind these services have not been published and no guarantees about their optimality are known.

III. Generalised Time-dependent Graph

As mentioned in the introduction, our approach to solving the fully multimodal EAP relies on the newly proposed generalised time-dependent (GTD) graph, which allows representing the combined road network (for individual and on-demand modes) and PT network (for PT modes) in a single structure. The GTD graph is a generalisation of the time-dependent graph with constant transfer times defined by Pyrga et al. [9] (the time needed to make a transfer between two lines at a stop is defined as a constant for each stop). The generalised time-dependent graph G is constructed from the following three structures: (1) *time-dependent graph* G^T for the PT network; (2) *network graph* G^N for the network of pavements, cycleways, and roads; (3) *graph connector* D of the time-dependent graph G^T and the network graph G^N. The GTD graph's structure is shown in Figure 1. Below, we describe each part of the construction in detail.

A. Time-dependent Graph

To model the network of scheduled PT services (e.g., bus, tram, underground), we use a time-dependent graph $G^T = (V^T, E^T, \rho^T)$ with constant transfer times [9]. We have chosen this model for its better performance than the time-expanded model [10]. Let S be the set of *stop nodes* corresponding to the stops that are physically present in the PT network. A stop node can be served by one or more routes. A *route* is a set of PT vehicle trips that are known to the public under the same route number identifier, e.g., the tram line number 3. Assuming that n is the number of routes using a stop $u \in S$, then n *route nodes* $R_u = \{r_1^u, \ldots, r_n^u\}$, one for each route, are associated with stop u. Route nodes are virtual nodes without corresponding counterparts in the real world and they are used to model constant transfer times. Without route nodes, it would not be possible to model non-zero transfer times between different routes at the same stop. The set of all route nodes is denoted as $R = \cup_{u \in S} R_u$. The set of nodes V^T of the time-dependent graph G^T is then defined as $V^T = S \cup R$.

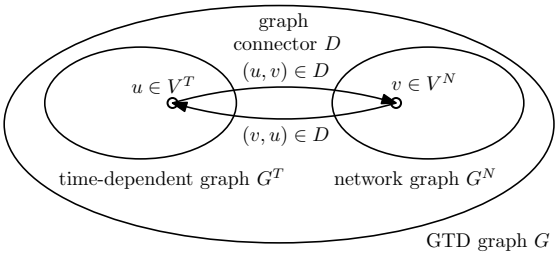


Fig. 1: The structure of a GTD graph

The set of edges E^T of the time-dependent graph G^T is defined as $E^T = A \cup B \cup C$ where A denotes the set of edges between route and stop nodes, B denotes the set of edges between stop and route nodes, and C denotes the set of *route edges* between route nodes of the same route. Edges $(v, w) \in A \cup B$ are called *transfer edges*. Formally, the sets are defined as follows:

$$\begin{aligned} A = &\ \cup_{u \in S}\{(r^u, u) | r^u \in R_u\} \\ B = &\ \cup_{u \in S}\{(u, r^u) | r^u \in R_u\} \\ C = &\ \cup_{u,v \in S}\{(r^u, r^v) | r^u \in R_u \wedge r^v \in R_v\} \text{ where } r^u \\ &\ \text{and } r^v \text{ are visited successively by the same route} \end{aligned}$$

The *link-traversal function* ${f'}_{(v,w)} : \mathbb{N} \to \mathbb{N}$ is associated with each edge $(v, w) \in C$ and defined as ${f'}_{(v,w)}(t) := t'$ where t is the departure time from v and $t' \geq t$ is the earliest possible arrival time at stop w. We assume that overtaking of vehicles on edges of the same route is not permitted. This means that the earliest arrival of a PT vehicle to a route node r_j^w corresponds to the earliest departure from an adjacent departure route node r_i^v.

Let the function g_v return the constant transfer time at stop v. For example in Figure 2, the transfer from a route node r_0^v to r_1^v and vice versa takes time g_v. Then the travel duration ${\rho^T}_{(v,w)} : \mathbb{N} \to \mathbb{N}$ of traversing an edge $(v, w) \in E^T$ from v at the departure time t is defined as

$${\rho^T}_{(v,w)}(t) := \begin{cases} 0 & \text{if } (v, w) \in A \\ g_v & \text{if } (v, w) \in B \\ {f'}_{(v,w)}(t) - t & \text{if } (v, w) \in C \end{cases}$$

B. Network Graph

To model the network for individual modes of transport (e.g., walk, bike, shared bike and car) and on-demand modes

[1] http://anachb.at/
[2] http://emtvalencia.es/geoportal/?lang=en_otp

978-1-4799-2914-613/$31.00 ©2013 IEEE

of transport (e.g., taxi), we use the *network graph* $G^N = (V^N, E^N, \rho^N)$ defined as a weighted directed graph, where the set of nodes V^N represents junctions and the set of edges E^N represents roads, pavements, and cycleways. The length of each edge $(v, w) \in E^N$ is given by the weight function $\rho^N : E^N \to \mathbb{R}_0^+$.

C. Graph Connector

In order to plan multimodal journeys using combinations of individual, on-demand, and PT modes of transport, the time-dependent graph G^T and the network graph G^N need to be interconnected. Let $\theta : S \to \mathcal{P}(\mathcal{V}^{\mathcal{N}})$ be a mapping that associates with each stop $v \in S$ a set of nodes $\theta(v) \in \mathcal{P}(\mathcal{V}^{\mathcal{N}})$ from the network graph. For the underground stops and large PT stations, the mapping assigns a stop a set of corresponding entrances from the network graph G^N. For the other PT stops, the mapping assigns to a stop a nearest pavement node from the network graph G^N.

Then the *graph connector* D of graphs G^T and G^N is defined as a set of interconnecting edges:

$$D = \{(v, w) | (v \in S \wedge w \in \theta(v)) \vee (v \in \theta(w) \wedge w \in S)\}$$

A length in metres $\rho_d((v, w)) = |v, w|$ is assigned to each $(v, w) \in D$ (the Euclidean distance between v and w is used).

D. GTD Graph

Finally, we can use the described structures to construct a unified network graph that supports multimodal journeys that use *any combination* of PT, individual, and on-demand modes of transport. Before defining the GTD graph, we define the edge weight ρ, the permitted modes function μ, and the permitted mode change predicate χ.

Firstly, let t be the departure time from node $v \in V$ and $vel \in \mathbb{R}^+$ the travel speed in metres per second. Then the *edge weight* $\rho_{(v,w)} : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ returns the travel duration (in seconds) of traversing the edge $(v, w) \in E$ at time t using travel speed vel:

$$\rho_{(v,w)}(t, vel) := \begin{cases} \rho^T((u, w), t) & \text{if } (v, w) \in E^T \\ \rho^N((v, w))/vel & \text{if } (v, w) \in E^N \\ \rho_d((v, w))/vel & \text{if } (v, w) \in D \end{cases}$$

Secondly, assuming $M = \{m_1, \ldots, m_l\}$ is the set of all l supported modes of transport, the function $\mu : E \to \mathcal{P}(\mathcal{M})$ returns the set of permitted modes of transport $\mu((v, w)) \in \mathcal{P}(\mathcal{M})$ at an edge $(v, w) \in E$. In our approach, we currently use the following modes of transport: *walk (W), bike (I), shared bike (S), car (C), taxi (X), bus (B), tram (T)*, and *underground (U)*. Especially in the network graph G^N, there are usually several modes of transport permitted to use a given edge, e.g., car, taxi, and bike.

Thirdly, we need to capture the fact that certain changes of mode of transport are possible only at some nodes. For example, changing from walk to shared bike or vice versa is only possible at bike sharing stations. Formally, the *permitted mode change* predicate $\chi_v : M \times M$ is associated with each node $v \in V$ and $\chi_v(m_1, m_2)$ returns true if it is possible to change the mode of transport from m_1 to m_2 at node v.

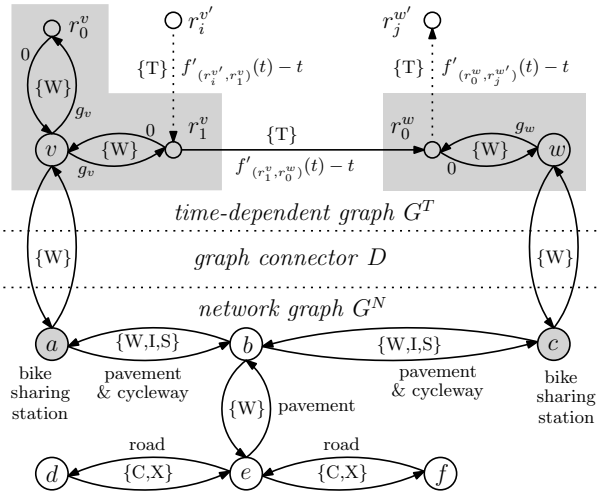


Fig. 2: An example of the GTD graph. Edges are annotated with the permitted modes of transport. Stop nodes $v, w \in S$ represent two tram stops that are connected by one tram route connecting four route nodes $(r_i^{v'}, r_1^v, r_0^w, r_j^{w'})$. Route nodes $R_v = \{r_0^v, r_1^v\}$ and $R_w = \{r_0^w\}$ associated with the respective stop nodes v and w are highlighted with grey background. Edges from the time-dependent graph G^T are also annotated with their weight (edge traversal time).

As an example, let $S^N \subset V^N$ be the set of bike sharing stations and $P^N \subset V^N$ be the set of park and ride (P+R) parking places. For the modes of transport currently used, the predicate $\chi_v(m_1, m_2)$ for each $v \in V$ and $m_1, m_2 \in M$ is defined as follows (`t` denotes true, `f` denotes false):

$$\chi_v(m_1, m_2) := \begin{cases} \texttt{t} & \text{if } v \in V^T & (1) \\ \texttt{t} & \text{if } v \in S^N \wedge ((m_1 m_2 = \text{WS}) & \\ & \quad \vee (m_1 m_2 = \text{SW})) & (2) \\ \texttt{t} & \text{if } v \in P^N \wedge ((m_1 m_2 = \text{CW}) & (3) \\ \texttt{t} & \text{if } m_1 = m_2 & (4) \\ \texttt{f} & \text{otherwise} & (5) \end{cases}$$

The defined predicate captures the five following rules: (1) change of mode of transport is not restricted for any stop $v \in V^T$; (2) change from walk to shared bike or vice versa is possible only at bike sharing stations $v \in S^N$; (3) change from car to walk is possible only at P+R parking places; (4) change from m_1 to $m_2 = m_1$ is not a change (it is always permitted to continue with the same mode of transport); (5) change of modes is not permitted in all other cases.

Finally, we define the *generalised time-dependent graph* as a weighted directed graph $G = (V, E, \rho, \mu, \chi)$ where $V = V^T \cup V^N$ and $E = E^T \cup E^N \cup D$. An example of a GTD graph is shown in Figure 2.

IV. Journey Planning Problem

In this section, we first describe the notions of a journey leg and a journey plan. Then, we define the fully multimodal earliest arrival problem.

978-1-4799-2914-613/$31.00 ©2013 IEEE

A. Journey Plan

Let the journey leg be a part of a journey plan that is either covered by the traveller on foot or by a movement by one and only one vehicle from one location to another. Formally, the *journey leg* $L = ((v_1, w_1), \ldots, (v_k, w_k))$ is defined as a sequence of $|L| = k$ edges $(v_j, w_j) \in E$. Edges are a finer-grained decomposition of a journey leg and represent the lowest-level, atomic parts of any journey plan. Then the *journey plan* is a quadruple $\pi = (P, \sigma, \phi, \psi)$:

- $P = (L_1, \ldots, L_n)$ is a sequence of $|P| = n$ journey legs L_i.
- Function σ denotes the mode of transport $\sigma(L_i) \in M$ that is used for journey leg L_i.
- Function $\phi : E \to \mathbb{N}$ returns the departure time from v for each edge $(v, w) \in E$.
- Function $\psi : E \to \mathbb{N}$ returns the arrival time at w for each edge $(v, w) \in E$.

Let $L[j]$ be the j-th element of a sequence of elements L and $|L|$ be the number of elements in L. Let $\xi(P)$ be the *flattened plan* constructed as the concatenation of all edges in all journey legs $L_i \in P$:

$$\xi(P) = (L_1[1], \ldots, L_1[|L_1|], \ldots, L_n[1], \ldots, L_n[|L_n|])$$

B. Fully Multimodal Earliest Arrival Problem

The *fully multimodal earliest arrival problem* is a pair $J = (G, r)$, where:

- $G = (V, E, \rho, \mu, \chi)$ is a *GTD graph*
- $r = (o, d, t)$ is a *journey request* specifying an origin $o \in V$, a destination $d \in V$, and a time of departure $t \in \mathbb{N}$

A *journey plan* $\pi = (P, \sigma, \phi, \psi)$, where $P = (L_1, \ldots, L_n)$, is then a solution of the fully multimodal earliest arrival problem $J = (G, r)$ if and only if all the following conditions hold:

1) Journey plan starts at the origin:
   $o = v$ where $(v, w) = L_1[1]$
2) Journey plan ends at the destination:
   $d = w$ where $(v, w) = L_n[|L_n|]$
3) All edges are present in the GTD graph:
   $\forall (v, w) \in \xi(P) : (v, w) \in E$
4) Edges form a path in the GTD graph:
   $\forall j \in \{1, \ldots, |\xi(P)| - 1\} :$
   $(v, w) = \xi(P)[j] \wedge (w, x) = \xi(P)[j + 1]$

V. Journey Planning Problem with Templates

For the fully multimodal earliest arrival problem with templates, we introduce the notion of a *journey plan template*. As mentioned in the introduction, journey plan templates give users and journey planner administrators a powerful way of parameterising the journey planner to obtain plans that best meet their constraints and preferences. For instance, a journey plan template that prefers environmentally friendly modes of transport can be designed by a journey planner administrator (e.g., a combination of walk and shared bike).

A. Journey Plan Template

A journey plan template constrain the journey plan in the permitted combination of modes on the level of journey legs. A *journey plan template* τ is defined as a regular expression over the transport modes alphabet M. As an example, we list three templates[3]:

- Taxi only: `^X$`
- Walk and PT: `^W((B|T|U)W)*$`
- Walk and shared bike: `^W(SW)?$`

We define several notions related to the journey plan templates. Let the word $\sigma(L_1) \ldots \sigma(L_n)$ be the *mode sequence* $\kappa(P)$ of a sequence of journey legs $P = (L_1, \ldots, L_n)$. Empty mode sequence $\kappa(\emptyset) = \epsilon$. We say that a sequence of journey legs P match a journey plan template τ if and only if the mode sequence $\kappa(P)$ matches the regular expression τ. Next, let $modes(\tau)$ be the set of modes of transport that are present in a template τ. Finally, the binary operator $\|$ over a mode sequence $m_1 \ldots m_n$ and a mode of transport $m \in M$ is defined as follows:

$$m_1 \ldots m_n \| m := \begin{cases} m_1 \ldots m_n & \text{if } m = m_n \\ m_1 \ldots m_n m & \text{otherwise} \end{cases}$$

B. Fully Multimodal EAP with Templates

The fully multimodal EAP with templates adds the notion of journey plan template to the fully multimodal EAP. Thus, the *fully multimodal earliest arrival problem with templates* is a triple $J = (G, r, \tau)$, where:

- $G = (V, E, \rho, \mu, \chi)$ is a *GTD graph*
- $r = (o, d, t)$ is a *journey request*
- τ is a *journey plan template*

A *journey plan* $\pi = (P, \sigma, \phi, \psi)$ is then a solution of the fully multimodal earliest arrival problem with templates $J = (G, r, \tau)$ if and only if all the following conditions hold:

1) Journey plan π is a solution of the fully multimodal earliest arrival problem $J = (G, r)$.
2) Journey legs $P = (L_1, \ldots, L_n)$ match the journey plan template, i.e., $\sigma(L_1) \ldots \sigma(L_n)$ matches τ.

VI. Solution Method

In this section, we present a method to solve the fully multimodal earliest arrival problem with templates using the GTD graph representation. The method uses a contextual view over the underlying GTD graph in order to use general shortest path algorithms to find the journey plans in the search space. This is enabled by storing the node context, i.e., the time of arrival and the modes of transport sequence used, in the contextual GTD graph.

A. Contextual GTD Graph

The contextual GTD graph is a *view* over an underlying GTD graph. The contextual GTD graph serves two main purposes. First, it allows filtering the available edges in the GTD graph with respect to the permitted modes of transport specified by a given journey plan template τ. Second, it

[3]POSIX Extended Regular Expression syntax is used.

978-1-4799-2914-613/$31.00 ©2013 IEEE

Function 1 Outgoing edges of a contextual node

Input: A contextual node (v, t_a, m_s) and a template τ

Output: A set of outgoing edges from (v, t_a, m_s) given τ

```
1:  function OUT((v, t_a, m_s), τ)
2:      O := ∅
3:      for all (v, w) ∈ E do
4:          for all m ∈ μ((v, w)) do
5:              m'_s := m_s ‖ m
6:              m_prev := m_n where m_s = m_1 ... m_n
7:              a := χ_v(m_prev, m)
8:              b := m'_s matches τ
9:              if m ∈ modes(τ) ∧ a ∧ b then
10:                 t' := t_a + ρ_(v,w)(t_a, λ(m))
11:                 O := O ∪ ((v, t_a, m_s), (w, t', m'_s))
12:             end if
13:         end for
14:     end for
15:     return O
16: end function
```

Function 2 Path to journey plan transformation

Input: A path K in G_τ

Output: A journey plan $\pi = (P, \sigma, \phi, \psi)$

```
1:  function DERIVEJOURNEYPLAN(K)
2:      i := 0
3:      for all ((v, t_a, m_s), (v', t'_a, m'_s)) ∈ K do
4:          if m_s ≠ m'_s then
5:              i := i + 1
6:              L_i := ()
7:              σ(L_i) := m_j where m'_s = m_1 ... m_j
8:          end if
9:          L_i := L_i ∘ (v, v')
10:         φ((v, v')) := t_a
11:         ψ((v, v')) := t'_a
12:     end for
13:     P := (L_1, ..., L_i)
14:     return (P, σ, φ, ψ)
15: end function
```

allows checking that the current partial journey plan matches a given journey plan template τ during the search process.

Let us define the graph formally. The *contextual GTD graph* G_τ over a GTD graph $G = (V, E, \rho, \mu, \chi)$ using a journey plan template τ is defined as $G_\tau = (V_\tau, E_\tau, \rho, \mu, \chi, \lambda)$. V_τ is a set of *contextual nodes* defined as triples (v, t_a, m_s) where:

- $v \in E$ is a node in the GTD graph
- $t_a \in \mathbb{N}$ is the arrival time at v
- m_s is a mode sequence $\kappa(P')$ of a sequence of journey legs P' from origin o (taken from the input journey request r) to node v

The context of contextual nodes corresponds to a GTD graph traversal at certain time using specific modes of transport, cf. Figure 3.

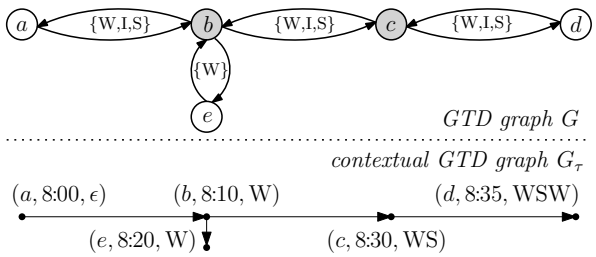


Fig. 3: An example of a GTD graph and its corresponding contextual GTD graph searched using the walk and shared bike template. Origin is set to a at 8:00; destination is set to d. Grey nodes b and c represent bike sharing stations. The bottom part of the figure shows how the contextual information is represented using the contextual nodes in the contextual GTD graph G_τ.

Let the function $\lambda : M \to \mathbb{R}^+$ returns the travel speed $\lambda(m)$ for a mode of transport $m \in M$. Then the set of contextual nodes V_τ and the set of edges E_τ is constructed using the origin contextual node (o, t, ϵ) and the function $\text{OUT}((v, t_a, m_s), \tau)$ (cf. Function 1) that returns the outgoing edges for a contextual node (v, t_a, m_s) and a template τ. At line 9 of Function 1, it is checked that a mode of transport m is present in the template τ, that a mode change from m_{prev} to m is permitted and that the current mode sequence m'_s matches the journey plan template τ.

The advantage of the contextual GTD graph is that unmodified general shortest path algorithms (e.g., A* or Dijkstra) can be used to find journey plans. This is enabled by embedding the domain information (e.g., permitted modes of transport and checking against a journey plan template) in the contextual GTD graph.

From the implementation point of view, the contextual GTD graph can be constructed *on request*. The nodes and edges are created on request only when they are needed during the search process of the respective shortest path algorithm.

B. Algorithm Specification

Now we present how the contextual GTD graph is used to solve the fully multimodal EAP with templates $J = (G, r, \tau)$. The input of the algorithm is an instance of the problem $J = (G, r, \tau)$ and the output is a journey plan $\pi = (P, \sigma, \phi, \psi)$ that solves the problem $J = (G, r, \tau)$. The algorithm works in two phases:

1) Shortest path algorithm on contextual GTD graph
2) Journey plan derivation

In the first phase, a general shortest path algorithm (e.g., A* or Dijkstra) is used to find a path $K = ((x_1, x_2), (x_2, x_3), \ldots, (x_k, x_{k+1}))$ of length $|K| = k$ in the contextual GTD graph $G_\tau = (V_\tau, E_\tau, \rho, \mu, \lambda)$ from the origin contextual node (o, t, ϵ) to the destination contextual node $(d, \cdot, \cdot)$. The edge weight function $\rho_{(v,w)}$ at line 10 of Function 1 returns the duration of traversing an edge

978-1-4799-2914-613/$31.00 ©2013 IEEE

TABLE I: Size of the Helsinki GTD graph

Graph name	Graph	Nodes	Edges
Time-dependent graph	G^T	50,320	112,127
Network graph	G^N	207,240	585,937
Graph connector	D	-	14,980
GTD graph	G	257,560	713,044

$(v, w) \in E$, therefore the journey plan is optimised with respect to its duration (i.e., the earliest arrival problem is solved).

In the second phase, the path K found in the contextual GTD graph G_τ is transformed into a journey plan $\pi = (P, \sigma, \phi, \psi)$. This is done using the DERIVEJOURNEYPLAN(K) function, cf. Function 2. The function iterates over edges $((v, t_a, m_s), (v', t'_a, m'_s)) \in K$. Every time the mode sequence is changed, a new journey leg L_i is created and its mode $\sigma(L_i)$ set. The edge $(v, v') \in E$ is then added to the current journey leg L_i using the operator $\circ$ that appends an element to a sequence and the departure $\phi(v, v')$ and arrival $\psi(v, v')$ is set.

It is important to note that if the shortest path algorithm used in the first phase of the algorithm is optimal, then the solution of the fully multimodal EAP with templates is optimal with respect to journey plan duration and the journey plan template τ.

VII. EVALUATION

Our proposed approach has been evaluated on real-world PT and road network data for Helsinki. The main purpose of the evaluation was to confirm that the GTD graph representation is flexible enough to allow successfully planning fully multimodal journeys with a variety of mode combinations. We were also interested in measuring how fast the GTD graph can be searched using standard algorithms – the runtimes results should, however, be treated as preliminary because we have not yet applied any speed-up techniques or other optimisation methods.

A. Data

Helsinki covers the area of 600 square kilometres. Kalkati.net XML database dump[4] provided by the Helsinki Regional Transport Authority (HSL) has been used as the data source for scheduled PT services. The data has been converted to the widely used General Transit Feed Specification (GTFS)[5] data format which is then used to construct the time-dependent graph G^T. OpenStreetMap[6] has been used as a data source for the network graph G^N. Basic statistics about the size of the GTD graph and its components are given in Table I. A fragment of the GTD graph is visualised in Figure 4. Note that in Helsinki, there are currently no bike sharing stations. For experimentation purposes, 150 bike sharing stations have therefore been added – the locations of the stations were chosen randomly with the uniform distribution over the nodes V^N of the network graph G^N. In addition, P+R parking places are not properly set in the OpenStreetMap data. For experimentation purposes, 10 P+R parking places were manually inserted into the map at the border of the Helsinki city centre.

[4] http://developer.reittiopas.fi/pages/en/kalkati.net-xml-database-dump.php

[5] https://developers.google.com/transit/gtfs/

[6] http://openstreetmap.org/

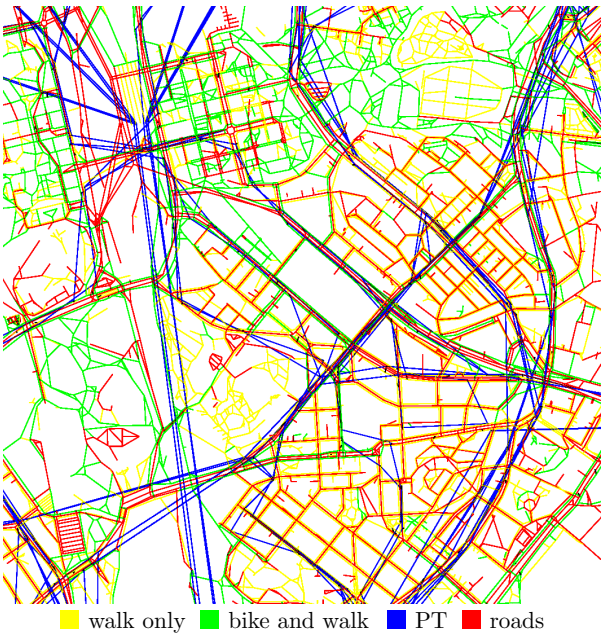


Fig. 4: Visualisation of a 2.4 km by 2.4 km fragment of the Helsinki GTD graph. Edge colours denote the modes of transport permitted at each edge, cf. legend. All other combinations of modes (e.g., car and taxi, bike only) are marked red. There are approximately 9,500 nodes and 27,700 edges in the visualisation.

B. Experiment Settings

We used seven journey plan templates $\tau \in T_7$ for the evaluation, cf. Table II. The templates have been chosen to reflect the typical combinations of modes used in modern multimodal transport systems. To allow a unified description of the results, we treat the fully multimodal EAP (without templates) as equivalent to the fully multimodal EAP with templates using the *empty template* permitting any combination of modes.

A* and Dijkstra's algorithms have been used to find a journey plan π given $J = (G, r, \tau)$. A* uses a duration heuristic $h(v)$ calculated as $h(v) = |v, d|/vel_{max}$ where $|v, d|$ is the Euclidean distance between current node v and destination node d, vel_{max} is the speed of the underground set to 120 km/h.

Following initial experiments, the better algorithm of the two has been chosen for each template $\tau \in T_7$. The templates $\tau \in T_7$ and their corresponding chosen algorithms are listed in Table II. Consequently, all templates use A* except the walk and PT template and car, walk and PT template where the Euclidean distance heuristic slows-down the A* search process [6] so the Dijkstra's algorithm is used.

978-1-4799-2914-613/$31.00 ©2013 IEEE

TABLE II: Journey plan templates used in the evaluation, along with the best performing algorithm for each template

Template name	Template regexp	Algorithm
Walk only	`^W$`	A*
Bike only	`^I$`	A*
Taxi only	`^X$`	A*
Walk and PT	`^W((B|T|U)W)*$`	Dijkstra
Car, walk and PT	`^CW((B|T|U)W)*$`	Dijkstra
Walk and shared bike	`^W(SW)?$`	A*
Empty template	N/A	A*

TABLE III: Average runtimes in milliseconds

Template name	Short	Medium	Long
Walk only	24	135	417
Bike only	15	60	178
Taxi only	31	103	239
Walk and PT	488	817	939
Car, walk and PT	384	477	504
Walk and shared bike	87	223	440
Empty template	376	758	891

The set of instances of the fully multimodal EAP with templates Q for the experiment were created in the following way. First, $n = 10{,}000$ origin-destination-departure triples $Q_t = ((o_1, d_1, t_1), \ldots, (o_n, d_n, t_n))$ were sampled using the uniform distribution over the coordinates of Helsinki area and the uniform distribution over the time interval from 8:00 to 18:00 on 17 Jan 2013. The maximum origin-destination distance was set to 40 km to exclude long trips that are not usual in the urban setting.

Then the origin and destination coordinates were converted to origin and destination nodes from graph G. Let $\delta(c, m)$ be a function that returns the nearest node in the GTD graph G given a coordinate c and a mode of transport m. For example, for the walk mode, the nearest node on a pavement is returned. Then the set of $|Q| = 70{,}000$ instances of the fully multimodal EAP with templates is constructed. Each of the origin-destination-departure triples Q_t is combined with all journey templates as follows:

$$Q = \{(G, (\delta(o, m_1), \delta(d, m_n), t), \tau) | \\ (o, d, t) \in Q_t \wedge \tau = m_1 \ldots m_n \in T_7\}$$

C. Implementation

The algorithm is implemented in JAVA 7. The results obtained are based on running the algorithm on one core of a 3.2 GHz Intel Core i7 processor of a Linux desktop computer with OpenJDK IcedTea7 2.3.7. The PostgreSQL 9.1 database spatially enabled with PostGIS 2.0.1[7] was used for storing and retrieving the data for the time-dependent graph G^T and the network graph G^N. The Osmosis 0.41[8] tool has been used to cut the Helsinki area from the OSM data dump and to put the data in the PostgreSQL database. Both the A* and Dijkstra's algorithm use the Fibonacci heap [5] implementation from the JGraphT 0.8.3[9] library.

[7] `http://postgis.net/`
[8] `http://wiki.openstreetmap.org/wiki/Osmosis`
[9] `http://jgrapht.org/`

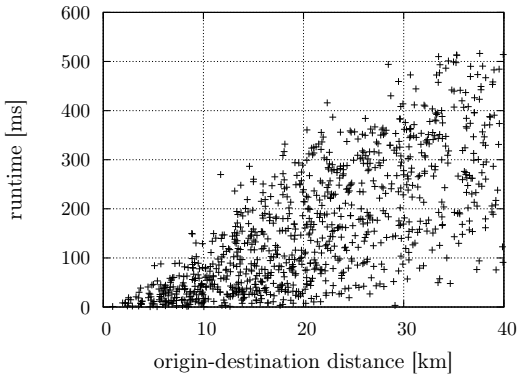


Fig. 5: Runtime against origin-destination distance (taxi only template, 1000 randomly selected requests)

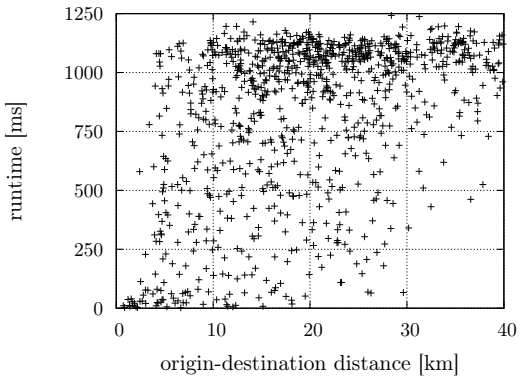


Fig. 6: Runtime against origin-destination distance (walk and PT template, 1000 randomly selected requests)

Geographical locations of all nodes in the OSM data and stops in the GTFS data are represented as their longitude and latitude values using the World Geodetic System (version WGS 84). WGS 84 is a *geographic* coordinate system type identified by SRID 4326[10] (Spatial Reference System Identifier). In order to simplify the complex calculation of the Euclidean distance between two nodes expressed in the WGS 84 coordinates (the calculation is very frequently used in the A* Euclidean distance heuristic), we use a *projected* coordinate system. The projected coordinate system is regional and projects the location from a spheroid to a plane. For locations in Helsinki, the spatial reference system "KKJ / Finland zone 2" with SRID 2392[11] is used.

D. Results

A solution for each problem instance $J \in Q$ has been computed. All instances are divided into three sets based on the distance of their origin and destination location: short (below 10 km), medium (10–20 km), and long (20–40 km). Average runtimes in milliseconds for each journey plan template and origin-destination distance interval are shown in Table III.

[10] `http://spatialreference.org/ref/epsg/4326/`
[11] `http://spatialreference.org/ref/epsg/2392/`

978-1-4799-2914-613/$31.00 ©2013 IEEE

Runtimes for all journey plan templates (except templates containing PT) are better than the runtimes for the empty template. This empirically confirms that the journey plan templates constrain the search space of the planner, which results in lower runtimes than when the empty template, which permits any combination of modes, is used. Runtimes for the empty template are better than the runtimes for templates containing PT because the heuristic of the A* algorithm leads the planner well into the destination using the taxi mode (for the majority of requests, taxi is the fastest mode of transport with the lowest journey plan duration).

In general, the templates containing more than one mode of transport are more difficult for the planner (higher branching factor and a larger contextual GTD graph) resulting in higher runtimes than the single-mode templates. Template with the lowest runtimes is the bike only template where the average runtimes range from 15 ms for the short requests up to 178 ms for the long requests. Template with the highest average runtimes is the walk and PT template where the average runtimes ranges from 488 ms for the short requests up to 939 ms for the long requests. The runtimes of car, walk and PT template are lower than the runtimes of the walk and PT template because a significant part of the journey is covered by car and only the last part from the P+R parking place to the destination by walk and PT modes.

Figures 5 and 6 show scatter plots of the search runtime versus the origin-destination distance for 1000 randomly selected requests. It can be observed that runtimes for the taxi only template in Figure 5 are more strongly correlated on the origin-destination distance than the runtimes of the walk and PT template in Figure 6.

E. Discussion

Compared to the algorithms employing state-of-the-art speed-up techniques specifically designed for road network and public transport network variant of EAP, the search times of our method are high. There are several reasons for such a behaviour. First and most importantly, the GTD graph representation is significantly more expressive and flexible, enabling searching for plans from a much richer family of journey plans, which necessarily increases the method's computational cost. Second, no speed-up techniques have yet been applied to accelerate the search of the contextual GTD graph. Last, the algorithm is currently implemented in JAVA whereas the search algorithms employing state-of-the-art speed-up techniques are usually implemented in C++. That said, even without the use of speed-up techniques and other optimisations, our method achieves practically usable runtimes.

So far, seven journey plan templates have been used in the evaluation. In the future, we plan to add the following useful plan templates:

- Taxi and PT: `^X?W((B|T|U)W)*X?$`
  A taxi can be used for covering the first, the last, or both first and last journey legs.
- Bike and PT: `^IW(UW(IW)?)*I?$`
  A traveller uses his or her own bike to get from an origin to a destination. Where possible and beneficial, PT mode of transport that permits taking bike along is used (in this example only the underground permits it).

VIII. Conclusion

We have presented a novel method for multimodal journey planning that allows finding multi-leg journeys utilising transport modes and combinations thereof not supported by existing journey planners. At the core of our method is a novel, generalised time-dependent graph representation which allows representing the fully multimodal journey planning problem with templates as a graph search problem that can be solved by general graph search algorithms. Experiments on realistic network data about the Helsinki transport system confirmed the viability of the approach – the planner was able to find a diverse set of journey plans and achieve runtimes which, although noticeably higher compared to algorithms optimised for basic variants of the earliest arrival problem, are generally usable and are likely to be significantly improved after the preliminary implementation of the approach is optimised.

Acknowledgment

This work was supported by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067), by the Ministry of Education, Youth and Sports of Czech Republic (grant no. LD12044 and 7E12065) and by the Czech Technical University (grant no. SGS13/210/OHK3/3T/13).

References

[1] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast Routing in Road Networks with Transit Nodes. *Science*, 316(5824):566, 2007.

[2] R. Bauer and D. Delling. SHARC: Fast and robust unidirectional routing. *ACM Journal of Experimental Algorithmics*, 14, 2009.

[3] G. S. Brodal and R. Jacob. Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. *Electronic Notes in Theoretical Computer Science*, 92(0):3–15, 2004.

[4] D. Delling, T. Pajor, and D. Wagner. Accelerating Multi-modal Route Planning by Access-Nodes. In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 587–598. Springer, 2009.

[5] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[6] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, USA, 2005.

[7] M. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188, 2002.

[8] T. Pajor. Multi-Modal Route Planning. Master's thesis, 2009.

[9] E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Efficient models for timetable information in public transportation systems. *Journal of Experimental Algorithmics (JEA)*, 12, 2008.

[10] E. Pyrga, F. Schulz, D. Wagner, and C. D. Zaroliagis. Experimental Comparison of Shortest Path Approaches for Timetable Information. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 88–99, 2004.

[11] P. Sanders and D. Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2005.

[12] F. Schulz. *Timetable information and shortest paths*. PhD thesis, 2005.

[13] H. Yu and F. Lu. A Multi-Modal Route Planning Approach With an Improved Genetic Algorithm. In *Joint International Conference on Theory, Data Handling and Modelling in GeoSpatial Information Science*, pages 343–348, 2010.

978-1-4799-2914-613/$31.00 ©2013 IEEE

Appendix E

Exploring Pareto Routes in Multi-Criteria Urban Bicycle Routing

Q. Song, P. Žilecký, M. Jakob, and J. Hrnčíř. Exploring pareto routes in multi-criteria urban bicycle routing. In *17th IEEE Intelligent Transportation Systems Conference (ITSC)*, 2014.

Exploring Pareto Routes in Multi-Criteria Urban Bicycle Routing

Qing Song, Pavol Zilecky, Michal Jakob and Jan Hrncir

Abstract— **To properly account for a broad range of route-choice factors in bicycle route planning, a multi-criteria optimization framework is needed. Unfortunately, in contrast to other categories of routing problems, optimal multi-criteria search has not yet been developed for bicycle routing. In this paper, we address this gap and provide a multi-criteria formulation of the bicycle routing problem and an optimum multi-label correcting algorithm for finding a full set of Pareto routes. To reduce the potentially very large number of Pareto solutions, we introduce a route selection algorithm, based on hierarchical clustering, for extracting a small representative subset of Pareto routes. We empirically evaluate our approach on a real-world cycleway network. We explore the size and structure of the set of Pareto routes and demonstrate the capability of our method to generate a practical set of bicycle routes in realistic conditions.**

I. Introduction

Utility cycling, i.e., using the bicycle as a mode of transport, is the original and the most common type of cycling in the world [7]. Cycling provides a convenient and affordable form of transport for most segments of the population. It has a range of health, environmental, economical, and societal benefits [5] and has therefore been promoted as a modern, sustainable mode of transport.

In contrast to car drivers, cyclists consider a significantly broader range of factors while deciding their routes. By employing questionnaires and GPS tracking, researchers have found that besides travel time and distance, cyclists are sensitive to slope, turn frequency, junction control, noise, pollution, scenery, and traffic volumes [1], [15]. Moreover, the relative importance of these factors varies among cyclists and can also be affected by weather conditions and the purpose of the trip [1]. Such a user- and context-dependent multi-criteriality makes bicycle routing a particularly difficult category of routing problems.

Several papers have been published on bicycle routing with multiple objectives. In [14], the authors discuss the design of a web-based tool that helps cyclists to determine safe and efficient routes. It uses a weighted combination of five different metrics to determine routes that optimise a trade-off among various safety factors and distance. In [12], the authors developed a web-based cycling route planner for Metro Vancouver, Canada. It enables users to find a cycling route based on one of the selectable preferences: shortest path route, restricted maximum slope, least elevation gain, least traffic pollution and most vegetated route. Hochmair et al. [9] proposed a bicycle trip planner for Broward County, Florida that enables users to select among five criteria: fast, safe (least interaction with traffic), simple, attractive and short. The criteria were decided based on the observed route choice behaviour of cyclists [8].

The authors are with the Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic, emails: song@agents.fel.cvut.cz, zilecky@agents.fel.cvut.cz, jakob@agents.fel.cvut.cz, hrncir@agents.fel.cvut.cz.

Although considering multiple objectives in the formulation of the routing problem, the existing approaches to bicycle routing do not use multi-criteria search methods to properly solve the resulting multi-criteria shortest path problem. This contrasts with other categories of route planning problems where the application of multi-criteria shortest path search techniques [11], [13] have been widely studied. See e.g. [4] and [3] for a multi-criteria algorithm for car routing and public transport journey planning, respectively.

Instead of employing a multi-criteria shortest path algorithm, existing approaches to bicycle routing transform multi-criteria search to single-criterion search either by optimizing each criteria function separately [12], [9] or by using a weighted combination of all criteria [14]. Unfortunately, scalarization of the multi-criteria problem using a linear combination of all criteria functions may miss Pareto optimal journeys [2] and, consequently, reduce the quality of the routes proposed to the user.

In this paper, we address the above limitations and explore how proper multi-criteria search can be applied to urban bicycle route planning. In doing so, we provide the following contributions. First, we provide a formal definition of the multi-criteria bicycle routing problem that incorporates realistic route choice factors based on recent studies of cyclists' behaviour [1], [15]. Second, we provide a multi-criteria search algorithm based on the multi-label correcting algorithm [11] which is able to generate all Pareto routes. Third, understanding that the number of Pareto routes can be very high, we propose a novel route selection method, based on hierarchical clustering in the route space, to extract few representative Pareto route suggestions. Fourth, we instantiate our approach with real-word data and study its properties, in particular the structure and size of the set of Pareto routes, in a realistic and challenging urban scenario.

II. Problem Formulation

A. Multi-criteria Bicycle Routing Problem

The cycleway network can be represented as a directed weighted *cycleway graph* $G = (V, E, g, h, l, f, \overrightarrow{c}, \overrightarrow{r})$, where V is the set of nodes representing start and end points (i.e., cycleway junctions) of cycleway segments, and $E = \{(u, v)|(u, v \in V) \wedge (u \neq v)\}$ is the set of edges representing cycleway segments. The cycleway graph is directed due to the fact that some cycleway segments in the map are one-way only. The function $g : V \rightarrow \mathbb{R}^2$ assigns a latitude and

a longitude values to each node $v \in V$. An altitude value is assigned to each node by the function $h : V \rightarrow \mathbb{R}$. The horizontal length of each edge $(u, v) \in E$ is given by the function $l : E \rightarrow \mathbb{R}_0^+$. For each edge $(u, v) \in E$, the function $f : E \rightarrow \wp(F)$ returns the *features* associated with the edge, which capture relevant properties of the edge as obtained from the input map data (e.g., the surface of the cycleway segment, or the road type). The set of all edge features is denoted by F. Note that an edge can have multiple features assigned to it, thus $f((u, v)) \subseteq \wp(F)$ with the number of elements $|f((u, v))| \geq 1$.

In the multi-criteria case, the cost on each edge is represented as a k-dimensional vector of criteria $\overrightarrow{c} = (c_1, c_2, \ldots, c_k)$. The value of any criterion $i \in (1, 2, \ldots, k)$ for the given edge $(u, v) \in E$ is computed by the cost function $c_i : E \rightarrow \mathbb{R}_0^+$. The vector $\overrightarrow{r} = (r_1, r_2, \ldots, r_k)$ returns the criteria coefficient for the k criteria of each edge; criteria coefficients reflect the aggregate influence of edge features on different criteria, e.g., the features indicating that the edge is a dedicated cyclelane with good-quality surface will have positive influence on the travel time criterion. Thus the function $r_i : \wp(F) \rightarrow \mathbb{R}_0^+$ computes the criteria coefficient of any criterion $i \in (1, 2, \ldots, k)$ for a given edge $(u, v) \in E$ with a set of features $f((u, v)) \subseteq \wp(F)$.

The *multi-criteria bicycle routing problem* is then defined as a pair $C = (G, r)$, where:

- $G = (V, E, g, h, l, f, \overrightarrow{c}, \overrightarrow{r})$ is the *cycleway graph*
- $r = (o, d, s)$ is a *journey request*; where $o \in V$ is an origin, $d \in V$ a destination, and $s \in \mathbb{R}^+$ an average cruising speed.

The solution of the multi-criteria bicycle routing problem is a full Pareto set of routes $\{\pi | \pi = (u_0\ (o), u_1, u_2, \ldots, u_w\ (d))\}$, each of which forms a finite path from the origin o to the destination d in G, with a cost value $c(\pi) = (\sum_{t=0}^{w-1} c_1(u_t, u_{t+1}), \ldots, \sum_{t=0}^{w-1} c_k(u_t, u_{t+1}))$ non-dominated by any other solution (a solution π_p dominates another solution π_q iff $c_i(\pi_p) \leq c_i(\pi_q)$, for all $1 \leq i \leq k$, and $c_j(\pi_p) < c_j(\pi_q)$, for at least one j, $1 \leq j \leq k$).

B. Criteria Function Definition

Taking into account technical constraints as well as studies of real-word cycle route choice behaviour [1], [15], we further consider a tri-criterion bicycle routing problem with three specific criteria. Specifically, we consider the travel time criterion c_1, the comfort criterion c_2 and the flatness criterion c_3 defined as follows.

1) Travel Time Criterion: The travel time criterion captures the preference towards routes that can be travelled in a short time. Travel time is a sensitive factor in cyclists' route planning especially for commuting purposes. To model the slow down caused by obstacle features such as stairs or crossings, we define the slowdown function $q : \wp(F) \rightarrow \mathbb{R}_0^+$ which returns the slowdown in seconds on the given edge $(u, v) \in E$ with a set of features $f((u, v))$.

Besides, changes in elevation may affect the cyclist's velocity and hence affect travel times. For the case of uphill rides, we define the positive vertical ascend $a : E \rightarrow \mathbb{R}_0^+$ and the positive ascend grade $a' : E \rightarrow \mathbb{R}_0^+$ for a given edge $(u, v) \in E$ as follows:

$$a((u, v)) := \begin{cases} h(v) - h(u) & \text{if } h(v) > h(u) \\ 0 & \text{otherwise} \end{cases}$$

$$a'((u, v)) := \frac{a((u, v))}{l((u, v))}$$

Analogously, for the case of downhill rides, we define the positive vertical descend $d : E \rightarrow \mathbb{R}_0^+$ and the positive descend grade $d' : E \rightarrow \mathbb{R}_0^+$ for a given edge $(u, v) \in E$ as follows:

$$d((u, v)) := \begin{cases} h(u) - h(v) & \text{if } h(u) > h(v) \\ 0 & \text{otherwise} \end{cases}$$

$$d'((u, v)) := \frac{d((u, v))}{l((u, v))}$$

To model the speed acceleration caused by vertical descend for a given edge $(u, v) \in E$, we define the downhill speed multiplier $s_d : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ as:

$$s_d((u, v), s_{\text{dmax}}) :=$$

$$:= \begin{cases} s_{\text{dmax}} & \text{if } d'((u, v)) > {d'}_c, \\ \frac{(s_{\text{dmax}} - 1) d'((u,v))}{{d'}_c} + 1 & \text{otherwise} \end{cases}$$

where $s_{\text{dmax}} \in \mathbb{R}^+$ is the maximum downhill speed multiplier, and ${d'}_c \in \mathbb{R}^+$ is the critical d' value over which a downhill ride would use the multiplier of s_{dmax}. This reflects the fact that the speed acceleration is remarkable for the ride on a steep downhill (compared to a mild one), however, it is limited due to safety concerns, bicycle physical limits and air drag.

Considering the integrated effect of edge length, the change in elevation and its associated features, the travel time criterion is defined as:

$$c_1((u, v)) = \frac{l((u, v)) + a_l a((u, v))}{s \cdot s_d((u, v), s_{\text{dmax}}) \cdot r_1((u, v))} + q((u, v)),$$

where s is the average cruising speed of a cyclist, and a_l is the penalty coefficient for uphill rides. The criteria coefficient $r_1((u, v))$ expresses how many times faster a cyclist can travel on a given edge $(u, v) \in E$ with a certain set of features $f((u, v))$. Intuitively, $c_1((u, v))$ can model the travel time of flat rides, uphill rides, and downhill rides with $s_d((u, v), s_{\text{dmax}}) = 1$ for uphill and flat scenarios, and $a((u, v)) = 0$ for downhill and flat scenarios.

2) Comfort Criterion: The comfort criterion captures the preference towards comfortable routes with good-quality surfaces and low traffic. The comfort criteria coefficient $r_2((u, v))$ is employed here to express the comfort property of an edge (u, v), which summarizes the effect of road surfaces and traffic volumes. The surface coefficient $r_{\text{s}}((u, v))$ penalises bad road surfaces, obstacles such as steps, and places where the cyclist needs to dismount his/her bicycle,

with small values indicating cycling-friendly surfaces; while the traffic coefficient $r_{\mathrm{t}}((u,v))$ measures traffic volumes by considering the infrastructure for cyclists (e.g., dedicated cycleways), the type of roads, and the junctions, where low-traffic cycleways are assigned small coefficient values. The maximum coefficient of the two is used here to avoid the cycleway segments that negatively affect the comfort the most, and the criteria function for comfort is defined as:

$$c_2((u,v)) = r_2((u,v))$$

$$r_2((u,v)) = \max\{r_{\mathrm{s}}((u,v)), r_{\mathrm{t}}((u,v))\}$$

3) Flatness Criterion: The flatness criterion captures the preference towards flat routes with minimum uphill segments. The criteria function for flatness takes into account the positive vertical ascend a, and it penalises uphill rides by the equivalent flat distance $a_l a((u,v))$ of the segment with a vertical ascend of $a((u,v))$. Criteria coefficient $r_3((u,v))$ is used to express the flatness property of an edge (u,v). The criteria function for flatness is defined as:

$$c_3((u,v)) = \frac{a_l a((u,v))}{s} \cdot r_3((u,v))$$

Next we introduce how to compute Pareto routes as well as how to identify significant ones for cyclists.

III. Multi-Criteria Route Generation Method

In this section, we describe how the multi-criteria bicycle routing problem is solved using the multi-label correcting algorithm and how representative Pareto routes are selected through a postprocessing clustering analysis.

A. Multi-label Correcting Algorithm

We employ the multi-label correcting algorithm [11] for computing the full set of Pareto routes. The multi-label correcting algorithm is an extension of Dijkstra's algorithm which operates on labels that have multiple values, one per optimization criterion.

First, we define variable structures required for algorithm execution: for each node $u \in V$, $L(u) := (u, (l_1(u), l_2(u), \ldots, l_k(u)), L^P(u))$ represents the *label* at u, which is composed of the node, the cost values with respect to each criterion, and the predecessor label $L^P(u)$. Each label has a record number, following the order it is generated (starting from 0). A priority queue Q is defined to maintain all labels created during the search. Since each node may be scanned multiple times, we define a bag structure $Bag(u)$ for each node u to maintain the non-dominated labels at u.

Then the multi-label correcting algorithm is composed of the following steps:

Step 1: Initialization. For a three-criterion optimization problem, we

- initialize the label at the origin $L(o) := (o, (0,0,0), null)$;
- insert the initial label $L(o)$ into Q and $Bag(o)$.

Algorithm 1: Multi-label correcting algorithm

Input: cycleway graph $G = (V, E, g, h, l, f, \overrightarrow{c}, \overrightarrow{r})$, origin node o, destination node d
Output: full Pareto set $Bag(d)$

$L(o) \leftarrow (o, (0,0,0), null)$
$Q.insert(L(o))$
$Bag(o).insert(L(o))$
while *!Q.isEmpty()* **do**
  $current \leftarrow Q.poll()$
  $u \leftarrow current.getNode()$
  $(l_1(u), l_2(u), l_3(u)) \leftarrow current.getCost()$
  $L^P(u) \leftarrow current.getPredecessorLabel()$
  foreach *edge* $u \rightarrow v$ **do**
    if $L^P(u).getNode() == v$ **then**
      continue
    end
    $C_i(v) \leftarrow l_i(u) + c_i(u,v)$ *for* $i = 1,2,3$
    $insert \leftarrow$ true
    foreach *label L(v)∈ Bag(v)* **do**
      if $l_i(v) \leq C_i(v)$ *for* $i = 1,2,3$ **then**
        $insert \leftarrow$ false
        break
      end
      if $C_i(v) \leq l_i(v)$ *for* $i = 1,2,3$ **then**
        $Bag(v).remove(L(v))$
        $Q.remove(L(v))$
      end
    end
    if *insert* **then**
      $next \leftarrow (v, (C_1(v), C_2(v), C_3(v)), current)$
      $Bag(v).insert(next)$
      $Q.insert(next)$
    end
  end
end
return *Bag(d)*

Step 2: Label extension. Extract from the priority queue Q the current minimum (in lexicographic order) label $current := (u, (l_1(u), l_2(u), l_3(u)), L^P(u))$. For each edge (u,v) out of node u, proceed as follows:

1) Compute new cost values $(C_1(v), C_2(v), C_3(v))$ to node v by adding the costs of edge (u,v) to $(l_1(u), l_2(u), l_3(u))$. If the new cost values are not dominated by any of existing labels $L(v) \in Bag(v)$, we:

- create a new label $(v, (C_1(v), C_2(v), C_3(v)), current)$ for node v;
- insert the new label into Q and $Bag(v)$.

2) Check if any existing label $L(v) \in Bag(v)$ is dominated by the new cost values $(C_1(v), C_2(v), C_3(v))$: If so, remove the label from the priority queue Q and $Bag(v)$.

Step 3: Pruning condition. Exit if the priority queue Q becomes empty; otherwise, go to *Step 2* and continue.

Algorithm 2: Extract routes

Input: full Pareto set Set
Output: Pareto routes π_p

```
1  p ← 0
2  foreach label L ∈ Set do
3      π_p ← ∅
4      while L != null do
5          u ← L.getNode()
6          π_p.insertAtBeginning(u)
7          L ← L.getPredecessorLabel()
8      end
9      p ← p + 1
10 end
```

A drawback of the multi-label correcting algorithm is that it can be quite slow since each node may be scanned multiple times, and domination checks are costly. To accelerate the algorithm, we record the predecessor label in the $label$ data structure; we then use the knowledge of the predecessor label to avoid extending the labels (in *Step 2*) to the nodes associated with the predecessor label (see lines 10–12 in pseudocode of Algorithm 1). Also, our $label$ data structure facilitates the retrieval of routes π_p, as illustrated in Algorithm 2.

B. Route Selection Algorithm

The multi-label correcting algorithm may produce a very large Pareto set of routes, many of which may be very similar and thus uninteresting to the user. We therefore propose a novel method for selecting a small representative subset of Pareto routes. Our method is based on clustering routes $\{\pi\}$ in the physical space, leveraging the observation that routes that share most of their segments usually have similar cost values. By first clustering all Pareto routes by their physical distance and then selecting only one solution in each of the clusters, we extract routes that differ both spatially and in terms of their cost values. The route selection algorithm is composed of the following steps:

Step 1: Evaluation of route distance. We use Jaccard distance [10] to measure the dissimilarity between Pareto routes. For routes π_p and π_q, the route distance is computed by dividing the difference of the sizes of the union and the intersection of two route sets by the size of the union:

$$d_J(\pi_p, \pi_q) := \frac{\mid (\pi_p \cup \pi_q) \mid - \mid (\pi_p \cap \pi_q) \mid}{\mid (\pi_p \cup \pi_q) \backslash \{o, d\} \mid}$$

We exclude the origin o and the destination d since all routes share the same origin and destination. Reasonably, the route distance definition obeys the triangle inequality.

Step 2: Route clustering. Given the distance metric defined in *Step 1*, we employ a hierarchical clustering method−single linkage clustering [6] to group routes into nested sets of clusters.

Step 3: Route selection. Route selection proceeds in two steps. In the first step, we select the so-called *single-cost optimum* Pareto routes, i.e., routes that have the lowest cost value for one of the three criteria. In the second step, we only consider the clusters that did not contain any of the single-cost optimum routes. From each of the remaining clusters, we select one route that has the highest average physical-space distance to all routes in the other clusters. Thus, in the end, we have at least one route from each of the clusters.

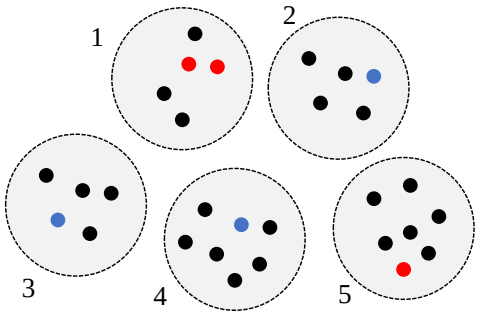


Fig. 1. Illustration of the route selection process. Red dots correspond to single-cost optimum routes; blue dots correspond to routes chosen based on their highest average distance to routes in the other clusters.

Route selection process is illustrated in Figure 1. Clusters 1 and 5 contain the three single-cost optimum routes (denoted by red dots) selected. From each of the remaining three clusters 2, 3 and 4, we selected one route (denoted by a blue dot) based on its distance to the routes in the other clusters.

IV. Implementation

We now describe important implementation details, in particular related to creating instances of the bicycle routing problem (see the definition in Section II-A) from real-world map data. The instantiation of the routing problem comprises of two steps. In the first step, the cycleway graph is created from map data and its nodes and edges are assigned respective map features. In the second step, the values of the three criteria functions are calculated for each node and edge.

A. Data

OpenStreetMap (OSM) data is used to create the cycleway graph. OSM data is organised into three entities: nodes, ways and relations, which are associated with various tags (features). Each map feature is denoted by a key and a value in the form of `entity::key::value`, e.g., `way::highway::primary`. Latitude and longitude of each node is mapped to function g, altitude of each node is mapped to h. The following map elements relevant for cyclists are loaded according to the information from OSM tags associated with OSM nodes, ways, and relations. We divide the map features into six categories:

- *Surface:* surface quality in terms of smoothness of the surface and surface material, e.g., asphalt, gravel, or cobblestone.
- *Obstacles:* steps and elevators.
- *Dismount:* places where cyclists need to dismount the bicycle, e.g., pavement, or footway crossing.

TABLE I

CATEGORIES OF MAP FEATURES TO CRITERIA FUNCTIONS MAPPING.

Criteria function	Categories of map features
Travel Time c_1	Surface, Obstacles, Dismount
Comfort c_2	Surface, Obstacles, Dismount, For bicycles, Motor roads, Crossings
Flatness c_3	$\emptyset$

TABLE II

TRAVEL TIME AND COMFORT CRITERIA BASE VALUES FOR FEATURES FROM THE SURFACE CATEGORY.

Entity	Key	Value	r'_1	r'_2
way	smoothness	bad	0.7	3
way	smoothness	excellent	1	0.5
way	smoothness	horrible	0.5	2
way	smoothness	intermediate	0.8	1
way	smoothness	very_bad	0.6	4
way	surface	cobblestone	0.7	5
way	surface	compacted	0.9	1.5
way	surface	dirt	0.7	3
way	surface	grass	0.65	5
way	surface	gravel	0.5	5
way	surface	ground	0.6	4
way	surface	mud	0.4	5
way	surface	paving_stones	0.75	1.5
way	surface	sand	0.6	4
way	surface	setts	0.8	2
way	surface	unpaved	0.75	4
way	surface	wood	0.65	4

- *For bicycles:* description of the infrastructure for cyclists, e.g., dedicated cycleway, cycle lane, or shared busway.
- *Motor roads:* category of a road that is also used by cars, e.g., primary, secondary, residential, or living street.
- *Crossings:* crossings, crossroads, or traffic lights on the road.

Geographical locations of all nodes in the OSM data are represented as their latitude and longitude values using the World Geodetic System (version WGS 84), a *geographic* coordinate system type. In order to simplify the complex calculation of the Euclidean distance between two nodes expressed in the WGS 84 coordinates, we use a *projected* coordinate system. For locations in Prague, the spatial reference system "S-JTSK (Ferro) / Krovak" is used. The horizontal length l of each edge is calculated based on the projected coordinates. Elevation h for all nodes in the OSM data is acquired using the Shuttle Radar Topography Mission (SRTM) project.

B. OSM Tags Mapping

Based on the effect of map features to different criteria, we further map the six category of features to the three optimization criteria, as shown in Table I.

Then, we provide the criteria base values for each feature in each category. The effect of OSM features from the surface category on travel time and comfort criterion are shown in Table II, where $r'_1 : F \rightarrow \mathbb{R}^+$ represents the criteria base value on travel time and $r'_2 : F \rightarrow \mathbb{R}^+$ the criteria base value on comfort. The total effect depends on all features associated with an edge that contribute to the travel time or comfort criterion. Suppose function $f((u,v))$ returns the features associated with an edge $(u,v) \in E$, then the criteria coefficients r_1 and r_2 are given by:

$$r_1((u,v)) = \min\{r'_1(p) | p \in f((u,v))\}$$

$$r_2((u,v)) = \max\{r'_2(p) | p \in f((u,v))\}$$

For the travel time criterion, the minimum base value is used since we are interested in a feature that reduces the cyclist's speed the most. In the case of the comfort criterion, the maximum base value is used since we take into account a feature that negatively affects the comfort the most. In this paper, we set $r_3((u,v)) = 1$ for all edges to minimize uphill rides exclusively.

TABLE III

VALUES FOR THE FEATURE SLOWDOWN FUNCTION q'.

Category	Entity	Key	Value	q'
crossing	node	crossing	island	20
crossing	node	crossing	traffic_signals	30
crossing	node	crossing	uncontrolled	15
crossing	node	crossing	unmarked	20
crossing	node	crossing	yes	15
crossing	node	crossing	zebra	15
crossing	node	highway	crossing	15
crossing	node	highway	traffic_signals	30
obstacles	node	highway	elevator	75
obstacles	node	highway	steps	25

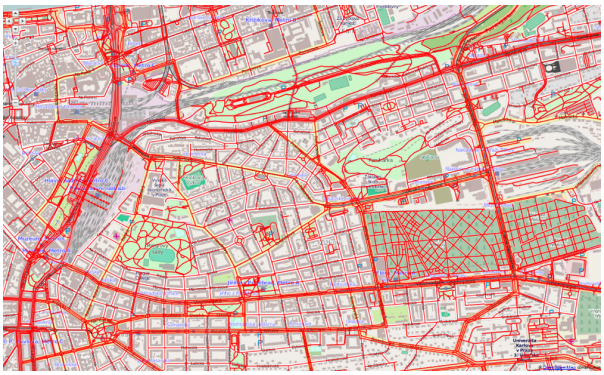


Fig. 2. A snapshot of Prague cycleway network.

Table III shows the slowdown base value $q' : F \rightarrow \mathbb{N}_0^+$ caused by each relevant feature $p \in F$. Similarly, the total effect depends on all features associated with the edge $(u,v) \in E$ that may cause a slowdown in seconds, hence the slowdown function $q : \wp(F) \rightarrow \mathbb{N}_0^+$ is given by:

$$q((u,v)) = \max\{q'(p) | p \in f((u,v))\}$$

V. EXPERIMENTAL EVALUATION

A. Setting

We evaluated our approach on the real cycleway network of Prague. Prague is a challenging experiment location due to its complex geography and fragmented cycling infrastructure,

TABLE IV

EVALUATION RESULTS.

Distance category	$\bar{d_{od}}$	N_p	$\bar{d_{J_p}}$	$\bar{d_{E_p}}$	$\bar{r_{cc}}$	N_s	$\bar{d_{J_s}}$	$\bar{d_{E_s}}$
Short	764.62	118.01	0.60	0.32	0.87	5.47	0.73	0.46
Medium	1616.66	820.18	0.65	0.28	0.83	5.68	0.83	0.49
Long	2582.14	2677.98	0.69	0.27	0.81	5.83	0.86	0.51

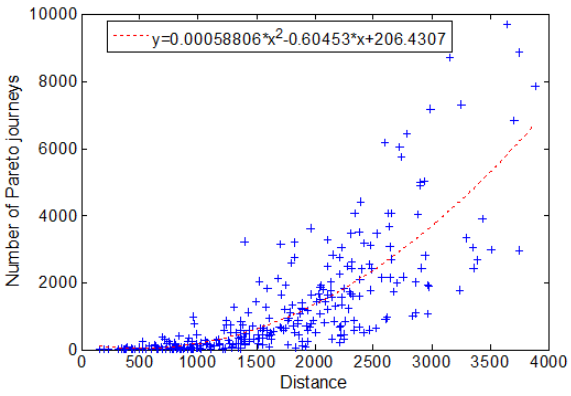


Fig. 3. Distribution of the number of Pareto routes. The dashed curve reflects the polynomial fitting of the data points.

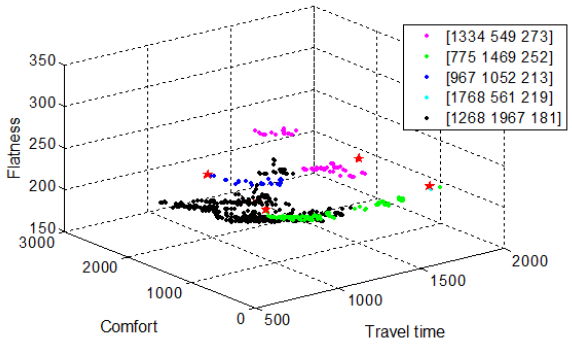


Fig. 4. Distribution of Pareto routes in the cost space. Routes that belong to different clusters are marked by different colors, the selected ones are marked by red stars with the cost values [travel time, comfort, flatness] shown in the legend.

which raises the importance of proper multi-criteria routing. For the experiments, we considered a strongly connected component of the cycleway network in Prague with 9411 nodes and 20420 edges. A snapshot showing the cycleway network of Prague is shown in Figure 2. Note that our aim was not to study computational times of the algorithm but to evaluate whether the proposed multi-criteria formalization of the bike routing problem and the route selection algorithm lead to practically useful routes.

The initialization parameters were set as follows: the average cruising speed $s = 14$ km/h; the penalty coefficient for uphill $a_l = 13$ (according to the route choice model developed in user study [1]); the maximum downhill speed multiplier $s_{\mathrm{dmax}} = 2.5$; the critical grade value $d'_{\mathrm{c}} = 0.1$; and the number of clusters $w = 5$.

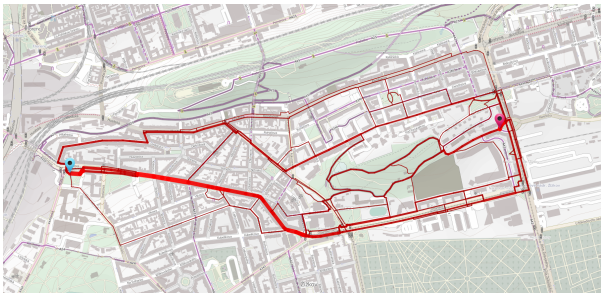
A total of 300 route requests, each specified by an origin-destination pair, generated randomly with uniform spatial distribution were used in the evaluation. Based on the Euclidean origin-destination distance, we divide the route requests into three subsets of 100 requests each, namely short distance, medium distance, and long distance.

B. Results

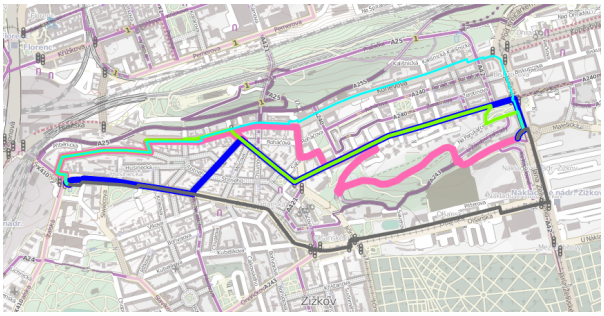
To evaluate the difference between the full set of Pareto routes and the selected ones, we count the following parameters: the average number of all Pareto routes $\bar{N}_p$ and of selected Pareto routes $\bar{N}_s$, the average Jaccard distance between routes in the full Pareto set $\bar{d_{J_p}}$ and between selected Pareto routes $\bar{d_{J_s}}$, and finally the Euclidean distance in the normalized cost space between routes in the full Pareto set $\bar{d_{E_p}}$ and between selected Pareto routes $\bar{d_{E_s}}$.

Table IV shows the evaluation results of each distance category, where $\bar{d_{od}}$ is the average Euclidean origin-destination distance; $\bar{r_{cc}}$ is a cophenetic correlation coefficient [6] reflecting the clustering quality. The closer the value of $\bar{r_{cc}}$ is to 1, the more accurately the clustering solution is. We can observe that the average number of Pareto routes increases notably with the distance. In addition, the network structure around the origin and destination also affects the number of Pareto routes; this can be seen from the distribution of the number of all Pareto routes with distance as shown in Figure 3, where the number of Pareto routes does not monotonically increase. Also we note that the average Jaccard distance and Euclidean distance in the normalized cost space between selected Pareto routes are greater than that between routes in the full Pareto set. This confirms that the selected routes are more dissimilar when dissimilarity is measured as distance in both the physical space and cost space. The average value of the coefficient $\bar{r_{cc}}$ is around 0.84, which indicates that the hierarchical clustering method and the distance metrics are effective for our problem.

To get further insight into the structure of the full set of Pareto routes and the representative subset of selected Pareto routes, we inspect one route request around a hilly area in Zizkov, Prague 3 in detail. Figure 4 and 5 illustrate the route distribution in the cost and physical space, respectively; there are 503 routes in the full Pareto set out of which 5 are selected for the representative subset. We can observe that our proposed method maintains the diversity of the Pareto set, where the routes selected are much further and more different both in the cost space and physical space. Besides, the routes are reasonable from the practical point



(a)



(b)

Fig. 5. Distribution of Pareto routes in the physical space. (a) Full set of 503 Pareto routes, where the cycleway segments passed by more Pareto routes are represented by wider and brighter lines. (b) The representative subset of selected five Pareto routes, with the color corresponding to that in Figure 4.

of view, trying to avoid the steepest areas and balancing among travel time, comfort and quietness. We can see that the route with the highest comfort (pink route in Figure 5) goes across a pleasant park area, but it takes longer and has the highest elevation gain among the five routes. The route with the shortest travel time (green one) follows main roads; it achieves time efficiency at the expense of relatively worse comfort and flatness. The blue route and the cyan route in Figure 5) go through different road areas, which have similar values of flatness and a balance between travel time and comfort. Finally, the flattest route (black one) goes through main roads all the time and it is thus the most uncomfortable among the five selected routes.

VI. Conclusions

We have investigated a multi-criteria approach to urban bicycle routing. In contrast to existing work, we have provided a well-grounded formal model of multi-criteria bicycle routing and we have applied a multi-criteria shortest path algorithm to find the full set of Pareto routes. Since the multi-criteria search can produce large Pareto sets with many similar routes, we have proposed a hierarchical clustering-based route selection method that can identify the most representative routes. We have integrated our approach with real-world OpenStreetMap data and evaluated it in challenging conditions of the city of Prague. The evaluation has confirmed the usefulness of the multi-criteria approach to bicycle routing and has shown that our specific method can generate representative sets of practically useful routes.

The presented research opens a range of exciting future research directions. The most immediate topic is the expansion of the evaluation to geographically different and spatially larger areas. The second, more challenging direction, is speeding up the routing algorithm in order to support real-time route planning[1]. Finally, the underlying cycleway problem model could be extended to consider additional aspects such as detailed junction models with traffic light and turn penalty consideration, real-time traffic information or weather conditions.

Acknowledgement

Supported by the European social fund within the framework of realizing the project "Support of inter-sectoral mobility and quality enhancement of research teams at Czech Technical University in Prague", CZ.1.07/2.3.00/30.0034. Supported by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067) and by the Ministry of Education, Youth and Sports of Czech Republic (grant no. 7E12065).

References

[1] J. Broach, J. Dill, and J. Gliebe. Where do cyclists ride? A route choice model developed with revealed preference GPS data. *Transportation Research Part A: Policy and Practice*, 46(10):1730 – 1740, 2012.

[2] D. W. Corne. The good of the many outweighs the good of the one: evolutionary multi-objective optimization. *IEEE Connections Newsletter*, pages 9–13, 2003.

[3] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck. Computing multimodal journeys in practice. In *SEA*, pages 260–271, 2013.

[4] D. Delling and D. Wagner. Pareto paths with sharc. In *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA09), volume 5526 of LNCS*, pages 125–136. Springer, 2009.

[5] C. Dora and M. Phillips. *Transport, environment and health*. WHO Regional Office for Europe, Copenhagen, 2000.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[7] D. Herlihy. *Bicycle: the history*. Yale University Press, 2004.

[8] H. H. Hochmair. Towards a classification of route selection criteria for route planning tools. *Developments in Spatial Data Handling, Springer, Berlin*, pages 481–492, 2004.

[9] H. H. Hochmair and J. Fu. Web Based Bicycle Trip Planning for Broward County, Florida. In *ESRI User Conference*, 2009.

[10] M. Levandowsky and D. Winter. Distance between sets. *Nature*, (5323):3435, 1971.

[11] E. Q. V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236 – 245, 1984.

[12] J. G. Su, M. Winters, M. Nunes, and M. Brauer. Designing a route planner to facilitate and promote cycling in Metro Vancouver, Canada. *Transportation Research Part A: Policy and Practice*, 44(7):495–505, 2010.

[13] Z. Tarapata. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *Int. J. Appl. Math. Comput. Sci.*, 17(2):269–287, June 2007.

[14] R. J. Turverey, D. D. Cheng, O. N. Blair, J. T. Roth, G. M. Lamp, and R. Cogill. Charlottesville bike route planner. In *Systems and Information Engineering Design Symposium (SIEDS)*, 2010.

[15] M. Winters, G. Davidson, D. Kao, and K. Teschke. Motivators and deterrents of bicycling: comparing influences on decisions to ride. *Transportation*, 38(1):153–168, 2011.

[1]The current implementation requires hundreds of seconds to calculate the full Pareto set, which far higher than is practically acceptable.

Appendix F

Ridesharing on Timetabled Transport Services: A Multiagent Planning Approach

J. Hrnčíř, M. Rovatsos, and M. Jakob. Ridesharing on timetabled transport services: A multiagent planning approach. *Journal of Intelligent Transportation Systems*, (accepted, available on-line), 2014.

Unspecified Journal
Volume 00, Number 0, Pages 000–000
S ????-????(XX)0000-0

RIDESHARING ON TIMETABLED TRANSPORT SERVICES: A MULTIAGENT PLANNING APPROACH

JAN HRNČÍŘ, MICHAEL ROVATSOS, AND MICHAL JAKOB

Abstract. Ridesharing, i.e., the problem of finding parts of routes that can be shared by several travellers with different points of departure and destinations, is a complex, multiagent decision-making problem. The problem has been widely studied but only for the case of ridesharing using freely moving vehicles not bound to fixed routes and/or schedules – ridesharing on timetabled public transport services has not been previously considered. In this paper, we address this problem and propose a solution employing strategic multiagent planning that guarantees that for any shared journey plan found, each individual is better off taking the shared ride rather than travelling alone, thus providing a clear incentive to participate in it. We evaluate the proposed solution on real-world scenarios in terms of the algorithm's scalability and the ability to address the inherent trade-off between cost savings and the prolongation of journey duration. The results show that under a wide range of circumstances our algorithm finds attractive shared journey plans. In addition to serving as a basis for traveller-oriented ridesharing service, our system allows stakeholders to determine appropriate pricing policies to incentivise group travel and to predict the effects of potential service changes.

1. Introduction

Travelling is an important and frequent activity, yet people willing to travel have to face problems with rising fuel prices, carbon footprint and traffic jams. One way to tackle these problems is through *ridesharing*, i.e., purposeful and explicit planning to create groups of people travel together in a single vehicle for parts of the journey. Participants in such schemes can benefit from ridesharing in several ways: sharing parts of a journey may reduce cost (e.g., through group tickets), carbon footprint (e.g., when sharing a private car), and travellers can enjoy the company of others on a long journey.

In general, ridesharing is a widely studied problem – existing work, however, focuses exclusively on ridesharing using vehicles that can move *freely* on a road transport network. This overlooks the potential for innovative future transport schemes that might exploit ridesharing using *timetabled public transport*. Here, customised group discount schemes could be devised to balance the load across different times of the day, or to make more efficient use of the capacity of public modes of transport. Also, joint travel can be used to increase the comfort and safety of individuals, e.g., for female travellers using night buses, or groups of schoolchildren. In more advanced scenarios, one could imagine ridesharing on public modes of transport being combined with working together while travelling, holding

2010 *Mathematics Subject Classification.* Primary.

meetings on the road or meeting people with common interests. We would argue that, in fact, *any* future intelligent transport scheme for citizens that attempts to address the *social dimension* of travel will be incomplete if it does not take into account timetabled public transport.

Except for our own earlier work (Hrnčíř and Rovatsos, 2012), no existing work seems to attempt to compute *joint* travel plans based on public transport timetables and geographical stop locations, let alone in a way that takes into account the *strategic* nature of the problem, which comes about through the different (and potentially conflicting) preferences of individual travellers. From the point of view of (multi-agent) planning (de Weerdt and Clement, 2009), i.e., the problem of synthesising sequences of actions to reach a certain goal – in this case, arrival at a destination from a given point of departure – for several travellers in parallel, ridesharing on timetabled services presents itself as a very complex application scenario: To begin with, even if one restricted oneself to centralised planning, the domain is huge – public transport data for the UK alone currently involves 240,590 timetable connections for trains and coaches (even excluding local city buses), which would have to be translated to a quarter of a million planning actions, at least in a naive formalisation of the domain. This is the case even if we assume a *non-strategic* setting, where individuals' preferences are not taken into account, and we are simply looking for a set of itinerary that gets everybody to their destination, without any regard for how costly this might be for the individual, or how the joint plan might favour some agents while putting others at a disadvantage. Moreover, considering a *strategic* setting where we are looking for a plan for multiple self-interested agents that are willing to cooperate only if it is beneficial for them is known to be exponentially harder than planning for each agent individually (Brafman and Domshlak, 2008). Yet any automated service that proposes joint journeys would have to guarantee such strategic properties in order to be acceptable for human users (who could then even leave it to the service to negotiate trips on their behalf).

In our previous paper (Hrnčíř and Rovatsos, 2012), we discussed the possibility of using a pre-processing step to group users together in such a way that would permit applying our algorithm to thousands or even millions of users in a larger geographical area (e.g., an entire country). In this paper, we present an improved version of our algorithm which includes a pre-processing step that clusters likely co-travellers together based on the overall direction of their individual trips and the distance between the origin and destination points of these individual trips. We show that this pre-processing step enables us to reduce plan computation times from over an hour to a few minutes in the worst case, while still resulting in substantial benefits from ridesharing for those participating in shared journeys. The core of our algorithm is based on a domain-independent *best-response planning* (Jonsson and Rovatsos, 2011) approach which is the only available planner that can solve strategic multiagent planning problems of the scale required, and whose properties and assumptions combine particularly well with the ridesharing problem in hand.

The contribution of our work is threefold: Firstly, we show that current multiagent planning technology can be used in important planning domains such as ridesharing by presenting its application to a practical problem that cannot be solved with other existing techniques. In the process, we describe the engineering steps that are necessary to deal with the challenges of real-world large-scale data and propose suitable solutions. Secondly, we present an algorithm that combines

different techniques in a practically-oriented way and works with real-world public transport timetables and realistic travel demand even though it is largely based on extensible, domain-independent, off-the-shelf heuristic problem solvers. Thirdly, we evaluate the proposed algorithm on such real-world data, taking public transport services of the Yorkshire region of the UK as an example. The evaluation not only analyses the performance of the proposed approach but also provides insights into general relationships between journey duration and cost in realistic ridesharing scenarios.

We start off with an overview of the related work in Section 2. This is followed by a formal specification of the timetabled transport ridesharing problem in Section 3 based on the model used in (Jonsson and Rovatsos, 2011). Section 4 introduces our four-phase algorithm for strategic planning in ridesharing domains. An extensive experimental evaluation of the algorithm is presented in Section 5. Section 6 presents a discussion of our results and Section 7 concludes.

2. Related Work

Ridesharing is a long known and widely studied problem – existing work, however, focuses exclusively on ridesharing using vehicles that can move freely on a road transport network, without schedule or route restrictions. The work on such *non-timetabled* ridesharing covers the whole spectrum from formal problem models, through solution algorithms up to practical consumer-oriented services and applications.

On the theoretical side, the vehicle-based ridesharing problem is typically formalised as a *Dial-a-Ride Problem (DARP)*. Different variants of DARPs exist, differing, for example, in the nature of traveller's constraints, the distribution of pickup and delivery locations, the criteria optimised, or the level of dynamism supported. A comprehensive review of different variants of DARPs, along with a list of algorithmic solution approaches, is given by Cordeau et al. (Cordeau and Laporte, 2007). Most of the existing approaches rely on a centralised coordination entity responsible for collecting requests and producing vehicle assignment and schedules, though more decentralised approaches have also been presented more recently (Wu et al., 2008). Bergbelia et al. (Berbeglia et al., 2010) summarise recent advances in *real-time ridesharing*, which has been gaining prominence with the growing penetration of internet-connected smartphones and GPS-enabled vehicle localisation technologies. Existing work almost exclusively considers a single mode of transport only. One of few exceptions is the work of Horn et al. (Horn, 2002) which considers demand-responsive ridesharing in the context of flexible, multi-modal transport systems; the actual ridesharing is, however, only supported for demand-responsive non-timetabled journey legs. On the practical side, there exist various online services for car (e.g., liftshare.com or citycarclub.co.uk), bike, and walk sharing as well as services which assist users in negotiating shared journeys (e.g., companions2travel.co.uk, travbuddy.com).

Journey planning for timetabled public transport services has been extensively studied in the single-agent case. The problem is typically formalised as the *earliest arrival problem* with two major ways to represent public transport timetables for the planning algorithm as a search graph. A *time-expanded approach* (Pyrga et al., 2008) where each event at a stop, e.g., the departure of a train, is modelled as a node in the graph; and a *time-dependent approach* (Brodal and Jacob, 2004)

where the graph contains only one node for each station. To speed up the search process, many speed-up techniques for a basic shortest-path algorithm, e.g., Dijkstra's algorithm, have been proposed, including the *multi-level graph* approach (Schulz, 2005), *access-node routing* (Delling et al., 2009), and *core-ALT* (Pajor, 2009). These algorithms are the basis of public travel planning services (e.g., in the UK, nationalrail.co.uk for trains, traveline.info and maps.google.com for multimodal transport) that automate *individual* travel planning for one or several modes of transport.

So although both ridesharing using freely moving vehicle and single-agent journey planning for timetabled services have been extensively studied, the combination of both, i.e., ridesharing on timetabled services, has not been – to the best of our knowledge – studied before (with the exception of our previous paper).

Automated planning technology (Ghallab et al., 2004) has developed a variety of scalable heuristic algorithms for tackling hard planning problems, where plans, i.e., sequences of actions that achieve a given goal from a given initial state, are calculated by domain-independent problem solvers. Unlike other approaches to route planning and ridesharing, automated planning techniques permit a fairly straightforward formalisation of travel domains, and allow us to capture the joint action space and complex cost landscape resulting from travellers' concurrent activities. In terms of algorithmic complexity, the kind of multiagent planning needed to compute ridesharing plans for several agents is significantly harder than single-agent planning for two reasons: Firstly, the ability of each agent to execute actions concurrently (Boutilier and Brafman, 2001) may result in exponentially large sets of actions available in each step in the worst case. Secondly, whenever individual agents have different (and potentially conflicting) goals (Brafman et al., 2009a), a joint solution must satisfy additional requirements, e.g., being compatible with everyone's individual preferences, or not providing any incentive for any individual to deviate from the joint plan. Solving the *general* multiagent planning for problem sizes of the scale we are interested in real-world ridesharing is therefore not currently possible using existing techniques.

Because of the desire to integrate different travellers' individual plans, ridesharing is quite similar to plan merging (e.g., (Foulser et al., 1992; Tsamardinos et al., 2000), where individual agents' plans are incrementally integrated into a joint solution. Compared to these approaches, however, in our domain every agent can always achieve their plan regardless of what others do, and agents do not require others' "help" to achieve their goals. This makes the problem simpler than those of plan merging though, in return, we place *much* higher scalability demands on the respective solution algorithms.

This explains also why, as will be shown below, we are able to achieve much higher scalability than state-of-the-art multiagent plan synthesis algorithms, e.g., (Nissim et al., 2010; Dimopoulos et al., 2012; Torreno et al., 2012). These algorithms exploit "locality" in different ways in order to be able to plan for parts of a multiagent planning problem while temporarily ignoring others, e.g., by considering non-interacting subplans in isolation from each other. In a sense, our problem involves even more loosely coupled sub-tasks, as these can be essentially solved in a completely independent way, except in terms of cost optimisation.

The relationship between our work and approaches that focus more on decentralised planning, plan co-ordination, and conflict resolution among independent

planning agents (e.g., (Cox and Durfee, 2005, 2009)) is similar – as no hard conflicts can arise among individual plans in ridesharing, it is not essential to co-ordinate individual plans with each other, other than for cost optimisation purposes.

Finally, as far as the strategic aspect is concerned, this is obviously also relevant to ridesharing as ultimately each co-traveller wants to achieve an optimal solution for themselves. Various approaches have studied this problem in the past (e.g., (Ephrati et al., 1995; van der Krogt et al., 2008; Brafman et al., 2009a), yet none of them has been shown to scale to the type of domain we are interested in, with the exception of (Jonsson and Rovatsos, 2011), which makes certain simplifying assumptions to achieve scalability: it does not consider *joint* deviation from equilibrium solutions (i.e., it only safeguards against individual agents opting out of a joint plan, not whole sub-groups of agents), and it assumes that agents will honour their promises when they have agreed on a joint plan. We believe that both these assumptions are reasonable in ridesharing, as we are envisioning a platform on which users would be automatically grouped together whenever a rideshare would be beneficial to each one of them. On such a platform, it is reasonable to assume that agreements could be enforced through a trusted third party, and that collusion among travellers could be avoided by not disclosing their identities to each other until the purchase of all tickets has been completed. Below, we describe how this algorithm serves as the basic planning method used in our ridesharing system.

3. Problem Formulation

Informally, the problem we are trying to solve is the following: Assume a (potentially very large) set of agents who represent individual travellers, with their individual trips specified in terms of origin and target location. Assume also that the agents want to optimise the individual utility accrued from a trip, and this utility may depend on the travel cost and number of people travelling along each leg of the journey (generally, we will assume that group travel has a positive effect on utility, as we want to study the impact of this very aspect on travel behaviour). Based on this information, we are looking for an algorithm that can identify appropriate groups of travellers who could share parts of their journeys using the full timetabling information of public transport systems, and determine a precise joint travel plan for each group. Also, we want to be sure that if we propose a plan to a group, none of the individual agents will have an incentive to improve on the proposed solution by deviating from it, i.e., we only want to suggest rideshares from which *all* travellers involved will benefit.

This section provides the formalisation of the *timetabled transport ridesharing problem*, which is then used by the ridesharing planning algorithm described in the next section. This formalisation builds on a representation of timetabled transport services captured at two different levels of granularity, which we call the *relaxed* and *full* transport services domain. From a planning perspective, problem formulation builds on the definition of a multiagent planning problem, which is essentially the combination of several individual planning problems involving an initial and goal state, as well as a set of actions that can be performed by the agent, i.e., the public transport services it can use.

We employ the multiagent paradigm because we want to account for every individual traveller's preferences, and to satisfy certain game-theoretic properties for solutions we calculate (i.e., joint travel plans, parts of which are shared among

more than one agent). To our knowledge, such modelling of strategic interaction situations cannot be done without an agent-based model.

3.1. **Timetabled Transport Services Representation.** Since the full travel planning domain with a full granularity of timetabled connections is too large for any current state-of-the-art planner to deal with, we distinguish the *full transport services domain* from what we call the *relaxed transport services domain*, which we will use to come up with an initial plan before mapping it to the full timetable information in our algorithm below. Roughly speaking, the relaxed domain contains information about all travel connections in the transport network with their respective shortest travel times, and ignores any concrete service timetables and information about which passengers are using which services, which are only included in the full domain (the relaxed domain also ignores direct connections among locations with intermediate stops, for reasons that will be explained below). Since only trips that are possible in the relaxed domain are possible in the full domain, this gives us a sound relaxation of the problem we can work with. This relaxation is of course incomplete in the general case, as many trips that are possible in theory cannot be performed in practice due to timetabling constraints, both regarding transport services and participating travellers' requirements.

The *relaxed domain* is a single-agent planning domain represented as a weighted directed graph $T = (V, E, w)$ where the set of nodes V represents the stops and the set of edges E represents the connections provided by a service. The graph must be directed because there exist stops that can only be used in one direction . There is an edge $e = (A, B) \in E$ from stop A to B in this graph if there is at least one connection from A to B in the timetable. The weight $w(e)$ of this edge is given by the weight function $w : E \to \mathbb{R}_0^+$ which returns the minimal time needed for travelling from A to B. A plan $P_i = \langle A_1 \to A_2, A_2 \to A_3, \ldots, A_{k-1} \to A_k \rangle$ found in the relaxed domain for the agent i is a sequence of $k - 1$ connections to travel from its origin A_1 to its destination A_k.

A small example of the relaxed domain is shown in Figure 1. An example plan for an agent travelling from C to F is $P_1 = \langle C \to D, D \to E, E \to F \rangle$. To give an idea of the difference between the relaxed domain and the full timetable in terms of domain complexity, there are 497 connections in the relaxed domain for trains and coaches in the Yorkshire area compared to 10,295 timetabled, actual connections.

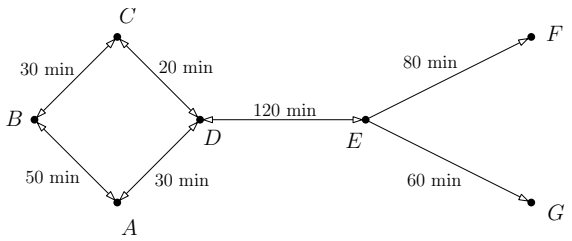


FIGURE 1. An example of the relaxed domain showing basic connection times (e.g., it takes 50 minutes to travel from A to B).

Direct trains that do not stop at every stop are filtered out from the relaxed domain for the following reason: Assume that in Figure 1, there is only one agent

travelling from C to F and that its plan in the relaxed domain is to use a direct train from C to F. In this case, it is only possible to match its plan to direct train connections from C to F, and not to trains that stop at C, D, E, and F. Therefore, the agent's plan cannot be matched against all possible trains between C and F which is problematic especially in the case where the majority of trains stop at every stop and only a few trains are direct. On the other hand, it is possible to match a plan with a train stopping in every stop to a direct train, as explained later in Section 4.4.

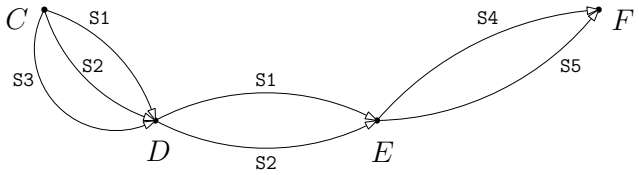


FIGURE 2. An example of the full domain with stops C, D, E and F for the merged plan of two single-agent plans $P = \{C \xrightarrow{\{1\}} D \xrightarrow{\{1,2\}} E \xrightarrow{\{1\}} F\}$.

Assume a set $N = \{1, \ldots, n\}$ of agents in the full domain, where each agent i has plan P_i from the relaxed domain. Then the *full domain* is a multiagent planning domain constructed using a *merged plan* P of single-agent plans $P_1, \ldots, P_n$ defined by formula

$$P = \bigcup_{i=1}^{n} P_i = (V', E', l')$$

where we interpret $\bigcup$ as the union of graphs that would result from interpreting each plan as a set of edges connecting stops. More specifically, given a set of single-agent plans, the plan merging operator $\bigcup$ computes its result in three steps: First, it transforms every single-agent plan P_i to a directed graph $T_i = (V_i, E_i)$ where the nodes V_i are the stops from the single-agent plan P_i and the edges E_i represent the atomic travel actions of P_i (for instance, a plan $P_1 = \langle C \to D, D \to E, E \to F \rangle$ is transformed to a directed graph $T_1 = \{C \to D \to E \to F\}$). Second, the merging operator performs a graph union operation $\bigcup_{i=1}^{n} T_i = (V', E', l')$ over the directed graphs and sets $V' = \bigcup_{i=1}^{n} V_i$, $E' = \bigcup_{i=1}^{n} E_i$, and labels every edge $e = (A, B) \in E'$ with the numbers of agents that are using the edge by a labelling function $l' : V' \times V' \to 2^N$. As an example , following Figure 1, the merged plan of plans of agent 1 travelling from C to F and sharing a journey from D to E with agent 2 would be computed as

$$\langle C \to D, D \to E, E \to F \rangle \cup \langle D \to E \rangle = \{C \xrightarrow{\{1\}} D \xrightarrow{\{1,2\}} E \xrightarrow{\{1\}} F\}$$

With this, the *full domain* is represented as a labelled directed multigraph $T' = (V', E_t, l, l')$ where the set of nodes V' represents the stops that are present in the merged plan P of plans from the relaxed domain. A set of edges E_t represents the journey services from the timetable. The labelling function $l : E_t \to \langle s, t_A, \tau \rangle$ returns a triple of a unique service name s, a departure time t_A from stop A, and a duration τ of the service journey between stops A and B for each edge

$e = (A, B) \in E_t$. The labelling function $l' : V' \times V' \to 2^N$ labels every edge $e \in E'$ with the number of agents using it.

A joint plan π with a timetable is a sequence $\pi = \langle a^1 \dots a^k \rangle$ of joint actions. Each joint action a^j from π represents a subset $N_j \subseteq N$ of agents travelling together using a specific service s_j.

In the example of the full domain in Figure 2, the agents can travel using some subset of five different services `S1` to `S5`. The full domain example is based on the group of agent 1 (travelling from C to F) and agent 2 (travelling from D to E) where initial single-agent plans have been found in the relaxed domain shown in Figure 1. In order to travel from C to D using service `S1`, an agent must be present at stop C before the departure of service `S1` to D.

3.2. **Multiagent Planning Problem.** To model the ridesharing problem, we use a multiagent planning formalism which is based on MA-STRIPS (Brafman and Domshlak, 2008) and coalition-planning games (Brafman et al., 2009b). States are represented by sets of ground fluents, actions are tuples $a = \langle pre(a), \mathit{eff}(a) \rangle$. These fluents are logical propositions describing aspects of the current state that may change over time, e.g., $at(1, l_1)$ to express that agent 1 is at location l_1. After the execution of action a, positive fluents p from $\mathit{eff}(a)$ are added to the state and negative fluents $\neg p$ are deleted from the state. For example, an action $travel(A, X, Y)$, when applied to the case of $A = 1$ travelling from $X = l_1$ to $Y = l_2$ would make $at(1, l_1)$ false and $at(1, l_2)$ true. Each agent has individual goals and actions with associated costs. There is no extra reward for achieving the goal, the total utility received by an agent is simply the inverse of the cost incurred by the plan executed to achieve the goal. In the ridesharing domain, the agents are the travellers, located in their origin locations in the initial state, and attempting to achieve goal states where they are at their destination locations. Agents specify the initial state and the goal state but their journey plans are computed for them centrally.

More formally, following the notation of (Jonsson and Rovatsos, 2011), a *multiagent planning problem* is a tuple

$$\Pi = \langle N, F, I, \{G_i\}_{i=1}^n, \{A_i\}_{i=1}^n, \Psi, \{c_i\}_{i=1}^n \rangle$$

where

- $N = \{1, \dots, n\}$ is the set of agents,
- F is the set of fluents,
- $I \subseteq F$ is the initial state,
- $G_i \subseteq F$ is agent i's goal,
- A_i is agent i's action set,
- $\Psi : A \to \{0, 1\}$ is an admissibility function,
- $c_i : \times_{i=1}^n A_i \to \mathbb{R}$ is the cost function of agent i.

$A = A_1 \times \dots \times A_n$ is the joint action set assuming a concurrent, synchronous execution model, and $G = \wedge_i G_i$ is the conjunction of all agents' individual goals. The assumption of synchronous action among agents here is an important simplification to make the problem more tractable. We will see below how it is possible to determine specific synchronisation points for jointly travelling agents when mapping the problem to the full timetabling information. A multiagent planning problem typically imposes concurrency constraints regarding actions that cannot or have to

be performed concurrently by different agents to succeed which the authors of (Jonsson and Rovatsos, 2011) encode using an admissibility function Ψ, with $\Psi(a) = 1$ if the joint action a is executable, and $\Psi(a) = 0$ otherwise.

A *plan* $\pi = \langle a^1, \ldots, a^k \rangle$ is a sequence of joint actions $a^j \in A$ such that a^1 is applicable in the initial state I (i.e., $pre(a^1) \subseteq I$), and a^j is applicable following the application of $a^1, \ldots, a^{j-1}$. We say that π *solves* the multiagent planning problem Π if the goal state G is satisfied following the application of all actions in π in sequence. The cost of a plan π to agent i is given by $C_i(\pi) = \sum_{j=1}^{k} c_i(a^j)$. Each agent's contribution to a plan π is denoted by π_i (a sequence of $a_i \in A_i$).

3.3. **Timetabled Transport Ridesharing Problem.** The real-world ridesharing domain used in this paper is based on the large and complex public transport network in the UK. An agent representing a passenger is able to use different modes of transport during its journey: walking, trains, and coaches. The aim of each agent is to get from its starting location to its final destination at the lowest possible cost. The cost of an agent's journey can be based on the weighted sum of several criteria such as journey duration, ticket price, mode of transport, and number of agents travelling together.

For the purposes of this paper, we will make the assumption that sharing a part of a journey with other agents is cheaper than travelling alone. While this may not currently hold in many public transport systems, defining hypothetical cost functions that reflect this would help assess the potential benefit of introducing such pricing schemes. This means that our cost functions reflect *synergies* occurring from the joint use of a resource, and this can be easily accommodated within the framework of best-response planning, where these positive effects on cost are simply treated as "negative contention", i.e., the cost to each agent when sharing a resource simply decreases instead of increasing. Note that this does not imply that every time an agent decreases her local cost this will benefit everybody else. For example, agent A might abandon the plan to share with B in order to reduce her overall cost, and join agent C instead, thus increasing B's cost, who will now travel alone. Thereupon B will try to improve on this result (and so on), the important property of BRP being that this process is guaranteed to terminate, and will result in a joint plan in which no individual agent can improve further on. Also, it is worth pointing out that a joint plan will not necessarily be globally optimal – its quality will depend on the initial plan computed before the best-response process.

The ridesharing problem is then, for a given travel demand expressed as a set of origin-destination pairs, one for each agent, finding groups of agents and corresponding shared journey plans. We define the ridesharing problem more formally by presenting definitions for problem instances and our formal solution concept: A *timetabled transport ridesharing problem* is a triple $P = \langle T, T', G \rangle$, where

- $T = (V, E, w)$ is the *relaxed domain* containing a set V of *public transport stops*,
- $T' = (V', E_t, l, l')$ is the *full domain* over the subset $V' \subseteq V$ of public transport stops, and
- $G = \{(o_1, d_1), \ldots, (o_c, d_c)\}$ is a *set of agent trips* (an agent's goal is to travel from an origin to a destination), where each agent's trip $g \in G$ is represented by a tuple $g = (o, d)$ denoting the agent's origin $o \in V$ and destination $d \in V$.

A solution to this problem is a *joint plan* $\pi = \langle a^1, \ldots, a^k \rangle$ specifying fully the shared journeys of agents in terms of connections from the timetable and fulfilling all agent trips $g \in G$. From the many joint plans possible, we are looking for such a joint plan that correspond to a Nash equilibrium, i.e., where no agent/traveller can unilaterally improve its individual journey cost.

4. Ridesharing Planning Algorithm

Once we have formalised the problem, we can proceed to a detailed description of the ridesharing planning algorithm. The algorithm takes as an input the timetabled transport ridesharing problem $P = \langle T, T', G \rangle$, a maximum travel group size $n_{\max}$, and a maximum bearing difference $\Delta\varphi$. A bearing $\varphi(t)$ for a trip $t = (o, d)$ is defined as an angle in degrees, measured in the clockwise direction, between the north reference ray and the origin-destination ray. Bearing of a trip is used to identify trips with a similar direction as these are more suitable for ridesharing than trips with opposite bearing. The output of the algorithm is a joint plan $\pi = \langle a^1, \ldots, a^k \rangle$ that fulfils all agent trips $g \in G$.

The main problem when planning for an identified group of agents with a centralised multiagent planner is the exponential blowup in the action space which is caused by using concurrent, independent actions (Jonsson and Rovatsos, 2011). Using a naive PDDL translation has proven that a direct application of a centralised multiagent planner to this problem does not scale well. As mentioned above, we tackle the complexity of the domain by breaking the planning process down into different phases that avoid dealing with the full fine-grained timetable data from the outset. The overall algorithm, which is shown in Figure 3, is designed to work in four phases, which we will now describe in detail.

4.1. **The Trip Grouping Phase.** The algorithm starts with the *trip grouping phase* where the trips $G = \{(o_1, d_1), \ldots, (o_c, d_c)\}$ are grouped into groups of at most $n_{\max}$ agents. Groups are created incrementally from G, until G becomes empty, in the following way: First, pick a trip $g' \in G$ at random. Then, create a set of candidate trips $G' = \{g \in G | \mathrm{bd}(g, g') \leq \Delta\varphi\}$ that have a similar bearing as g' (function $\mathrm{bd}(g, g')$ calculates the bearing difference between trips g and g'). Next, create a group $G_j \subseteq G'$ by selecting at most $n_{\max}$ trips with minimum spatial difference $\mathrm{sd}(\cdot, g')$ to g'. Here, the spatial difference $\mathrm{sd}(g, g')$ of two trips g and g' is defined as

$$\mathrm{sd}(g, g') = |o, o'| + |d, d'|,$$

where $|o, o'|$ denotes the direct distance between the origins of the two trips, and $|d, d'|$ the direct distance between their destinations. Once a group G_j is created, the trips $g \in G_j$ are deleted from the set of all trips G.

For each group G_j, a joint journey plan π with a timetable is found by applying the next three phases of the algorithm.

4.2. **The Trip Planning Phase.** In the *trip planning phase*, an initial journey is found for each agent i from the set of agents G_j using the relaxed domain $T = (V, E, w)$ where the action set is identical for every agent and contains all transport services available in the transport network. A journey for each agent is calculated independently of other agents in the scenario using a single-agent planner. As a result, each agent is assigned a single-agent plan P_i which will be further optimised in the next phase. This approach makes sense in our domain because the agents do

Input

- Timetabled transport ridesharing problem $P = \langle T, T', G \rangle$
- Maximum travel group size $n_{\max}$
- Maximum bearing difference $\Delta\varphi$

1. The trip grouping phase

Set $j = 0$

While $G \neq \emptyset$ **do**

(1) Pick a trip $g' \in G$ at random
(2) Create a set of candidate trips $G' = \{g \in G | \mathrm{bd}(g, g') \leq \Delta\varphi\}$
(3) Create a group $G_j \subseteq G'$ by selecting at most $n_{\max}$ trips with minimum spatial difference $\mathrm{sd}(\cdot, g')$ to g'
(4) Delete trips $t \in G_j$ from G
(5) Set $j = j + 1$

For each created group $G_j = \{1, \ldots, n\}$ **do** the next three phases

2. The trip planning phase

For $i = 1, \ldots, n$ **do**

Find an initial journey for agent i using a single-agent planner

3. The best-response phase

Do until no change in the cost of the joint plan

For $i = 1, \ldots, n$ **do**

(1) Create a simpler best-response planning problem from the point of view of agent i
(2) Minimise the cost of i's plan without changing the plans of others

4. The timetabling phase

Identify independent groups of agents $I = \{u_1, \ldots, u_m\}$, where $u_i \in 2^N$

For $i = 1, \ldots, m$ **do**

(1) Find the relevant timetable for group u_i
(2) Match the joint plan of u_i to timetable using a temporal single-agent planner in the full domain with the relevant timetable

Output

- Joint plan $\pi = \langle a^1, \ldots, a^k \rangle$ that fulfils all agent trips $g \in G$

FIGURE 3. Four-phase algorithm for finding shared journeys for agents.

not need each other to achieve their goals and they cannot invalidate each other's plans. A PDDL specification for the relaxed domain is shown in Section 4.5.2.

4.3. **The Best-response Phase.** The *best-response phase* is based on the relaxed domain. Again, the action set is identical for every agent and contains all transport services available in the transport network. The algorithm uses the best-response

planning algorithm as described below. It iteratively creates and solves simpler best-response planning problems from the point of view of each individual agent. In the case of the relaxed domain, the best-response planning problem looks almost the same as a problem of finding a single-agent journey. The difference is that, as we have explained in Section 3.3, we make the assumption that the cost of travelling is smaller when an agent uses a connection which is used by one or more other agents. A specific cost function used for the evaluation of the algorithm is defined in Section 5.2.

Iterations over agents continue until there is no change in the cost of the joint plan between two successive iterations. This means that the joint plan cannot be further improved using the best-response approach. The purpose of this is not only to exploit local, "greedy" optimisations for single agents in an overall schedule of plans. It also ensures that the proposed joint solution is compatible with the incentives of individual agents, i.e., they could do no better on their own by deviating from it. The fact that the first iteration of the best-response optimisation starts from initial plans that agents can perform on their own ensures this (any subsequent plan generated will be cheaper to them). The output of the best-response phase is a merged plan P of the single-agent plans in the relaxed domain (defined in Section 3.1) that specifies which connections the agents use for their journeys and which segments of their journeys are shared. The merged plan P will be matched to the timetable in the final phase of the algorithm.

4.3.1. *Best-response Planning.* The *best-response planning* algorithm proposed in (Jonsson and Rovatsos, 2011) is an algorithm which, given a solution π^k to a multiagent planning problem Π, finds a solution π^{k+1} to a *transformed planning problem* Π_i with minimum cost $C_i(\pi^{k+1})$ for agent i among all possible solutions, while considering all other agents' plans to be fixed:

$$\pi^{k+1} = \arg\min\{C_i(\pi)|\pi \text{ identical to } \pi^k \text{ for all } j \neq i\}$$

The transformed planning problem Π_i is obtained by rewriting the original problem Π so that all other agents' actions are fixed, and agent i can only choose its own actions in such a way that all other agents still can perform their original actions. Since Π_i is a single-agent planning problem, any cost-optimal planner can be used as a best-response planner.

In (Jonsson and Rovatsos, 2011), the authors show how for a class of congestion planning problems, where all fluents are *private*, the transformation they propose allows the algorithm to converge to a Nash equilibrium if agents iteratively perform best-response steps using an optimal planner. This requires that every agent can perform its actions without requiring another agent, and hence can achieve its goal in principle on its own, and conversely, that no agent can invalidate other agents' plans. Assuming infinite capacity of vehicles (or, more realistically, large enough capacities to accommodate at least the number of agents for whom we are trying to find a plan), the relaxed domain is an instance of a congestion planning problem: following the definition of a congestion planning problem in (Jonsson and Rovatsos, 2011), all actions are private, as every agent can use modes of transport on their own and the other agents' concurrently taken actions only affect action cost. The convergence of the best-response phase derives from the theorem presented in (Jonsson and Rovatsos, 2011) which states that for any congestion planning problem, best-response planning converges to a pure-strategy Nash equilibrium.

The best-response planner works in two phases: In the first phase, an initial plan for each agent is computed (e.g., each agent plans independently or a centralised multiagent planner is used). In the second phase, the planner solves simpler best-response planning problems from the point of view of each individual agent. The goal of the planner in a best-response planning problem is to minimise the cost of an agent's plan without changing the plans of others (though the cost of their plans might change as explained in Section 3.3). Consequently, it optimises a plan of each agent with respect to the current joint plan.

This approach has several advantages. It supports full concurrency of actions and the best-response phase avoids the exponential blowup in the action space resulting in much improved scalability. For the class of potential games (Monderer and Shapley, 1996), it guarantees convergence to a Nash equilibrium. On the other hand, it does not guarantee the optimality of a solution, i.e., the quality of the equilibrium in terms of overall efficiency is not guaranteed (it depends on which initial plan the agents start off with). However, experiments have proven that it can be successfully used for improving general multiagent plans (Jonsson and Rovatsos, 2011).

4.4. **The Timetabling Phase.** In the final *timetabling phase*, the optimised shared journeys are matched against timetables using a temporal single-agent planner which assumes the full domain. For this, in a first step, independent groups of agents with respect to journey sharing are identified. An independent group of agents is defined as an edge disjoint subgraph of the merged plan P. This means that actions of independent groups do not affect each other so it is possible to find a timetable for each independent group separately.

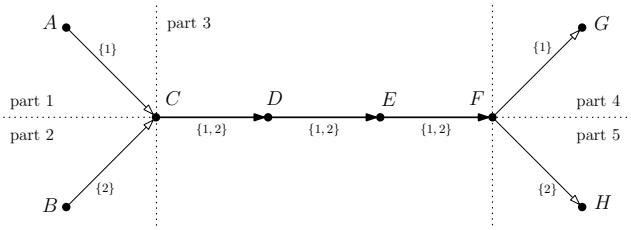


FIGURE 4. Parts of the group journey of two agents.

Then, for every independent group, *parts* of the group journey are identified. A *part* of the group journey is defined as a maximal continuous segment of the group journey which is performed by the same set of agents. As an example, there is a group of two agents that share a segment of their journeys in Figure 4: Agent 1 travels from A to G while agent 2 travels from B to H. Their group journey has five parts, with the shared part (part 3) of their journey occurring between stops C and F.

In order to use both direct and stopping trains when the group journey is matched to the timetable, the *relevant timetable* for a group journey is composed in the following way: for every part of the group journey, return all timetable services in the direction of agents' journeys which connect the stops in that part. An example of the relevant timetable for a group of agents from the previous example is shown

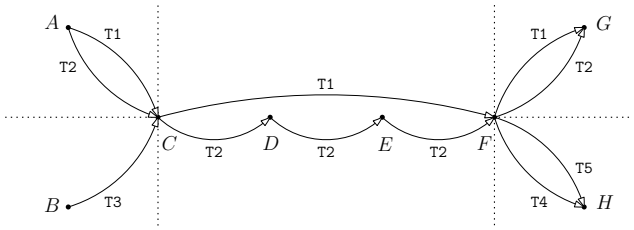


FIGURE 5. The full domain with services from the relevant timetable. There are five different trains `T1` to `T5`, and train `T1` is a direct train.

in Figure 5. Now, the agents can travel using the direct train `T1` or using train `T2` with intermediate stops.

The relevant timetable for the group journey is used with the aim to cut down the amount of data that will be given to a temporal single-agent planner. For instance, there are 9,881 timetabled connections for trains in the Yorkshire area. For an example journey of 4 agents, there are only 634 services in the relevant timetable which is approximately 6% of the data. As a result, the temporal single-agent planner gets only the necessary amount of data as input, to prevent the time-consuming exploration of irrelevant regions of the state space.

In the timetabling phase, every agent in a group of agents tries to spend the shortest possible time on its journey. When matching the plan to the timetable, the temporal planner tries to minimise the sum of durations of agents' journeys including waiting times between services. A PDDL specification for the full domain is shown in Section 4.5.2.

Once the timetabling phase finishes, the algorithm adds a joint plan $\pi^j = \langle a^1, \ldots, a^k \rangle$ for the identified group of agents $G_j \in G$ to the final joint plan $\pi = \pi \bigcup \pi^j$. The algorithm then proceeds to the next group $G_j \in G$.

4.5. **Implementation.** This section describes two important aspects of the algorithm implementation. It deals with the conversion of public transport timetables data to the Planning Domain Definition Language and with the choice of planners for implementing the individual phases of the algorithm.

4.5.1. *Importing Timetables.* To be able to use timetables data of public transport services (cf. Section 5.1) with modern AI planning systems, it has to be converted to the Planning Domain Definition Language (PDDL). We transformed the data in three subsequent stages. First, we transformed the NPTDR and NaPTAN XML data to a spatially-enabled PostgreSQL database. Second, we automatically processed and optimised the data in the database. The data processing by SQL functions in the procedural PL/pgSQL language included the following steps: merging bus bays at bus stations and parts of train stations, introducing walking connections to enable multi-modal journeys, and eliminating duplicates from the timetable. Finally, we created a script for generating PDDL specifications based on the data in the database. More details about the data processing and PDDL specifications can be found in (Hrnčíř, 2011).

```
(define (domain travelplanner)
    (:requirements :typing :action-costs)
    (:types location)
    (:predicates
        (connection ?origin - location ?destination - location)
        (at ?loc - location)
    )
    (:functions
        (time ?origin - location ?destination - location)
        (total-cost)
    )
    (:action go
        :parameters (?o ?d - location)
        :precondition (and (at ?o) (connection ?o ?d) )
        :effect (and
            (at ?d) (not (at ?o))
            (increase (total-cost) (time ?o ?d)) )
    )
)
```

FIGURE 6. The domain file for the relaxed domain.

```
(:durative-action go-agent-1-2_C-D
 :parameters (?s - service)
 :duration (= ?duration (+
     (- (+ (departure C D ?s) (runtime C D ?s)) (agent-time agent1))
     (- (+ (departure C D ?s) (runtime C D ?s)) (agent-time agent2))
 ))
 :condition (and
     (at start (connection C D ?s))
     (at start (at agent1 C))
     (at start (<= (agent-time agent1) (departure C D ?s)))
     (at start (at agent2 C))
     (at start (<= (agent-time agent2) (departure C D ?s)))
 )
 :effect (and
     (at end (at agent1 D))
     (at start (not (at agent1 C)))
     (at end (assign (agent-time agent1)
                     (+ (departure C D ?s) (runtime C D ?s))))
     (at end (at agent2 D))
     (at start (not (at agent2 C)))
     (at end (assign (agent-time agent2)
                     (+ (departure C D ?s) (runtime C D ?s))))
))
```

FIGURE 7. A durative action *go-agent-1-2_C-D* in the domain file for the full domain.

4.5.2. *PDDL definitions.* In the relaxed domain used in the trip planning and best-response phase, a single agent aims to travel from its origin to its destination. The domain file contains two predicates, two functions and only one action, cf. Figure 6. The predicate `connection` is true when there is an edge from `?origin` to `?destination` (there are separate edges for walking, travel by bus or train), the predicate `at` denotes the current location of the agent. The function `time` returns the cost of travelling from the location `?origin` to `?destination`. The action `go`

moves the agent from the location `?o` to `?d` and it increases the total cost of the plan which is stored by the `total-cost` function. The problem file then contains the origin and destination of the agent, the list of stops, and the list of connections between the stops and their costs.

In the full domain used in the timetabling phase, multiple agents aim to travel from their origins to their destinations. The full domain is a multiagent planning domain constructed using a merged plan P of single-agent plans $P_1, \ldots, P_n$. Therefore, it contains only the stops that are present in the union of these plans, with the shared parts of the journeys and "who shares which part of the journey" already specified. In the process of finding a plan for the full domain, a joint plan of the group of agents is instantiated with concrete timetabled services.

The domain file contains a list of partially instantiated durative actions for travelling from one stop to another, where origin, destination, and agents using this action are instantiated, and the only free variable is the name of the service the agents are going to use. The function `(agent-time ?a - agent)` is used to store the current time of the agent `?a`. An example of a durative action is shown in Figure 7. The durative action *go-agent-1-2_C-D* enables agent 1 and 2 to travel together from stop C to stop D. If the travel from C to D is shared by three agents, the domain file would contain an action *go-agent-1-2-3_C-D*.

Let N be the number of agents travelling together, at_i the current time of agent i, $d_{CD}(s)$ the departure time of service s from the stop C to D and $r_{CD}(s)$ its duration. Then, the duration D_{CD} of the action to travel from the stop C to D is computed as $D_{CD} = \sum_{i=1}^{N} (d_{CD}(s) + r_{CD}(s) - at_i)$. The temporal planner tries to minimise the sum of the durations of agents' journeys. In other words, it tries to find a journey with minimal waiting times between services.

The conditions of the action are the following: there must be a connection by the service s between the stops C, D and the agents must be present at the stop C before the departure of service s. Once the action is executed, the agents are located at stop D and their current time is set to the arrival of the service s at stop D. The problem file contains origins and destinations of the agents and the list of services and their departures and durations.

4.5.3. *Planners.* All three single-agent planners used for the implementation were taken from recent International Planning Competitions from 2008 and 2011. We use LAMA (Richter and Westphal, 2008) in the trip planning phase and for each of the individual single-agent best-response iterations in the best-response phase. LAMA is a sequential *satisficing* (as opposed to cost-optimal) planner which searches for any plan that solves a given problem and does not guarantee optimality of the plans computed. LAMA is a propositional planning system based on heuristic state-space search. Its core feature is the usage of landmarks, i.e., propositions that must be true in every solution of a planning problem.

SGPlan$_6$ (Hsu and Wah, 2008) and POPF2 (Coles et al., 2011) are the two temporal satisficing planners used in the timetabling phase. Such temporal planners take the duration of actions into account and try to minimise makespan (i.e., total duration) of a plan but do not guarantee optimality. The two planners use different search strategies and usually produce different results. This allows us to run them in sequence on every problem and to pick the plan with the shortest duration. It is not strictly necessary to run both planners, one could save computation effort by trusting one of them.

In many of the experiments, the $SGPlan_6$ and POPF2 used in the timetabling phase returned some plans in the first minute but then they continued exploration of the search space without returning any better plan. To account for this, we imposed a time limit for each planner in the temporal planning stage to 2 minutes for a group of up to 4 agents and 4 minutes otherwise.

5. Evaluation

We have evaluated the proposed ridesharing algorithm on realistic scenarios based on real-world public transport timetables and travel demand data for the Yorkshire area of the United Kingdom. The size of the area was dictated solely by our need to evaluate the algorithm on the whole travel demand (approximately 100,000 train trips per day). The ridesharing planning algorithm itself scales up well up to the area of the whole UK, as was shown in our previous work (Hrnčíř and Rovatsos, 2012). However, there the algorithm was evaluated on travel demand that was very sparsely and randomly sampled, not necessarily showing any correlation to actual travel demand profiles.

5.1. **Domain Data.** The timetables of public transport services were taken from the National Public Transport Data Repository (NPTDR, data.gov.uk/dataset/nptdr) which is publicly available from the Department for Transport of the British Government. For the evaluation of the algorithm, we used data from 2010, which is provided in TransXChange XML, in an XML-based UK standard for interchange of route and timetable data.

National Public Transport Access Nodes (NaPTAN, data.gov.uk/dataset/naptan) is a UK national system for uniquely identifying all the points of access to public transport. Every point of access (bus stop, railway station, etc.) is identified by an ATCO code (a unique identifier for all points of access to public transport in the UK), e.g., *9100YORK* for York Rail Station. Each stop in the NaPTAN XML data is also supplemented by common name, latitude, longitude, address and other pieces of information. This data also contains information about how the stops are grouped together (e.g., several bus bays that are located at the same bus station).

Table 1. Numbers of trips per day in the Yorkshire area (Eurostat, 2012; Office for National Statistics, 2001).

Transport mode	**Modal split**	**100% trips**	**50% trips**	**5% trips**
Trains	5.3%	106,035	53,017	5,302
Coaches	0.3%	6,002	3,001	300
Local buses	6.0%	120,039	60,020	6,002
Passenger cars	88.3%	1,766,576	883,288	88,329
Total	100.0%	1,998,651	999,326	99,933

The experiments are situated in the Yorkshire area (East and West Yorkshire, East Riding of Yorkshire, York, and Selby administrative areas) which covers an area of approximately 130 by 70 km, i.e., around 9,100 km^2. According to the UK origin-destination census data from 2001 (Office for National Statistics, 2001), there are 2 million passenger trips a day in the Yorkshire area. In order to focus on the timetabled public transport trips, the modal split in the UK across different

TABLE 2. Experiment scenarios parameters overview.

Scenario parameter	**Parameter values**
Travel demand generation	{realistic, random}
Ridesharing demand proportion	{100%, 50%, 5%}
Modes of transport	{trains only, trains and coaches}
Maximum travel group size	$\{2, 4, 6, 8\}$

modes of transport in 2001 (Eurostat, 2012) was used to estimate the number of trips for each mode, cf. Table 1.

Since we assume that all agents are travelling on the same day and that all journeys must be completed within 24 hours, in what follows below we consider only public transport timetables data for Tuesdays (this is an arbitrary choice that could be changed without any problem).

5.2. **Cost Model.** The timetable data used in this paper (cf. previous section) contains neither information about ticket prices nor distances between adjacent stops, only durations of journeys from one stop to another. This significantly restricts the design of a cost functions used for the planning problems. Therefore, the cost functions used in this paper are based solely on the duration of journeys. The cost $c_{i,n}$ for agent i travelling from A to B in a group of n agents is then defined by equation (5.1):

$$c_{i,n} = \left(\tfrac{1}{n}\, 0.8 + 0.2\right) c_i \tag{5.1}$$

where c_i is the individual cost of the single action to i when travelling alone. In this paper, we take this to be equal to the duration of the journey from A to B.

This is designed to approximately model the discount for the passengers if they buy a group ticket: The more agents travel together, the cheaper the shared (leg of a) journey becomes for each agent. Also, an agent cannot travel any cheaper than 20% of the single-agent cost. In reality, pricing for group tickets could vary, and while our experimental results assume this specific setup, the actual price calculation could be easily replaced by any alternative model.

5.3. **Experiment Scenarios.** We used the following parameters as factors in experiment scenarios: (1) travel demand generation; (2) ridesharing demand proportion; (3) modes of transport considered; (4) maximum travel group size. The values of the parameters are summarised in Table 2. We set the maximum bearing difference parameter of the algorithm to $\Delta\varphi = 25$ degrees for all scenarios.

Travel Demand Generation. We use two types of travel demand. The *realistic* travel demand generation is based on the UK census 2001 origin-destination data (Office for National Statistics, 2001) that contains numbers of trips carried out from every origin district to every other destination district. District-to-district trip counts are mapped to stop-to-stop trip counts in the following way: For each origin-destination district pair, the desired number of trips is generated randomly from the Cartesian product of stops in the origin and destination district. Since the UK census origin-destination data is not provided at the level of granularity required to select concrete stops in the travel network, we have sampled these within each district with probability proportional to the density of services passing through a stop. This is based on the assumption that service density roughly follows numbers of

travellers using a stop. In addition to the realistic demand, we also experimented using *random* travel demand generated randomly from the Cartesian product of stops in the Yorkshire area, assuming a uniform distribution over stops.

From the travel demand distribution generated, only trips with a straight-line distance between the origin and the destination in the interval 25–100 km are used for the evaluation (when using roads or rail tracks, this interval stretches approximately to a real distance of 40–160 km). This interval was chosen to filter out trips that are too short to be planned in advance and therefore not very suitable for sharing. This led to the removal of 86% of all trips in the realistic travel demand generation process and to the removal of 30% of all trips in the random travel demand generation process.

Ridesharing Demand Proportion. In order to observe the behaviour of the system with different densities of trips we set the portion of travel demand to 100%, 50%, and 5% of the total number of trips.

Modes of Transport. In order to evaluate the behaviour of the algorithm on both a unimodal and a multi-modal public transport network, *trains only* and a combination of *trains and coaches* were used in the experiments. In the Yorkshire area, there are 150 (201) stops, 330 (495) connections in the relaxed domain, and 9,881 (10,289) connections in the timetable for trains (and coaches).

Maximum Travel Group Size. The maximum travel group size $n_{\max}$ is one of the algorithm's inputs that restricts the size of groups created in the trip grouping phase. We set this parameter to 2, 4, 6, and 8 as after initial testing, it became clear that groups of a larger size are almost never practicable.

5.4. **Metrics.** We evaluate the performance of the algorithm in terms of three different metrics: improvement in the cost of agents' journeys, their prolongation, and the computation time of the algorithm.

Cost Improvement. To evaluate the net benefit of using our method for ridesharing, we calculate the cost improvement for the agents' journeys. To calculate this, recalling that $C_i(\pi) = \sum_j c_i(a^j)$ for a plan is the cost of a plan $\pi = \langle a^1, \ldots, a^k \rangle$ to agent i, assume $n(a^j)$ returns the number of agents with whom the jth step of the plan is shared. We can define the cost of a shared travel plan $C_i'(\pi) = \sum_j c_{i,n(a^j)}(a^j)$ using equation (5.1). With this, we can calculate the cost improvement ΔC as follows:

$$\Delta C = \frac{\sum_{i \in N} C_i(\pi_i) - \sum_{i \in N} C_i'(\pi_N)}{\sum_{i \in N} C_i(\pi_i)} \tag{5.2}$$

where N is the set of all agents, π_i is the single-agent plan initially computed for agent i, and π_N is the final joint plan of all agents after completion of the algorithm (which, though it is in reality a set of several plans for different subgroups of N, is interpreted as a single plan for the "grand coalition" N and reflects how subgroups within N share parts of their individual journeys).

Prolongation. On the one hand, ridesharing is beneficial in terms of cost. On the other hand, a shared journey has a longer duration than a single-agent journey in most cases, because agents have to take later services than they could use on their own if they are waiting for co-travellers to arrive. In order to evaluate this trade-off, we measure journey prolongation. Assume that $T_i(\pi)$ is the total duration of a plan to agent i in plan π, and, as above, π_i/π_N denote the initial single-agent plans and

the shared joint plan at the end of the timetabling phase, respectively. Then, the prolongation ΔT of a journey is defined as follows:

$$\Delta T = \frac{\sum_{i \in N} T_i(\pi_N) - \sum_{i \in N} T_i(\pi_i)}{\sum_{i \in N} T_i(\pi_i)} \tag{5.3}$$

Computation Time. To assess the scalability of the algorithm, we measure the amount of time needed to create groups of agents in the first phase of the algorithm and then to plan shared journeys for all agents in each group.

5.5. **Results.** In this section, we present the results of the evaluation in terms of journey cost improvement, journey prolongation, and computation time of the algorithm. Exhaustive experiment design was used; all metrics were evaluated for all combinations of all values of all scenario parameters. Specifically, for each type of travel demand generation, we tested all combinations of modes of transport, and for each ridesharing demand proportion, we generated the travel demand. Then for each maximum group size, the whole travel demand is an input for the algorithm which in its trip grouping phase creates the groups of agents for ridesharing. From the set of all groups created, a detailed journey plan with a timetable is found in the last three phases of the algorithm for a sample of 80 randomly chosen groups (sampling was performed to reduce experiment computational time while maintaining significance of the results). Each possible experiment configuration is averaged over 8 stochastic travel demand generation instances. This leads to an overall number of 30,720 groups of agents over which the algorithm was evaluated. The results obtained are based on running the algorithm on one core of a 3.2 GHz Intel Core i7 processor of a Linux desktop computer with a PostgreSQL 9.1 database (spatially enabled with PostGIS 2.0.1).

Cost Improvement. The average cost improvement obtained in our experiments is shown in Figure 8. It shows that the more agents are grouped together in the trip grouping phase of the algorithm, the higher the improvement. These results were obtained based on the specific cost function (5.1) we have introduced to favour ridesharing, and which would have to be adapted to the specific cost structure that is present in a given transport system. Also, the extent to which longer journey times are acceptable for the traveller depends on their preferences, but these could be easily adapted by using different cost functions.

Prolongation. The average prolongation of journeys is shown in Figure 8 where 8% of groups with prolongation greater than 100% is filtered out from the average calculation (these are the journeys which, though feasible, are unlikely to be accepted by travellers). The graph shows that the more agents are grouped together in the trip grouping phase of the algorithm, the higher the prolongation. Furthermore, the prolongation with the 5% ridesharing proportion is much higher then when considering 50% or 100% ridesharing proportion. As the density of trips drops, the agents in groups are more spatially dispersed, which causes higher relative prolongation ratios.

Figure 9 shows a scatter plot of cost improvement versus prolongation for individual trips for 5% and 50% ridesharing proportion. It can be observed that with a higher ridesharing proportion, the majority of the groups has either prolongation very close to 0% (identical trips are shared) or has a very high cost improvement (between 50% and 60%). With a lower ridesharing proportion, there are many more groups with lower cost improvement or higher prolongation. What is encouraging

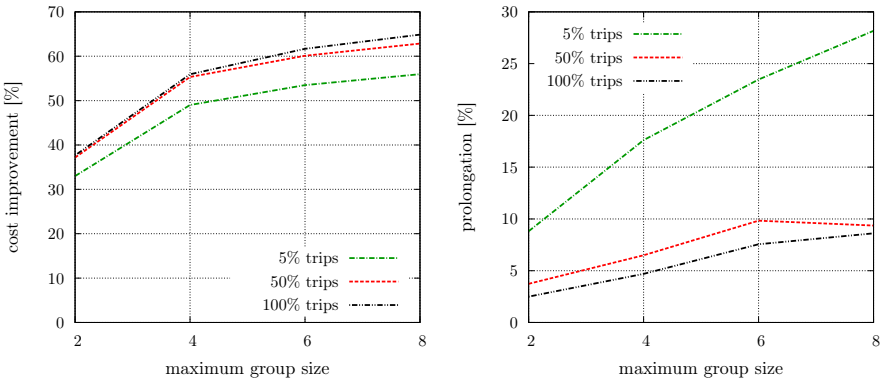
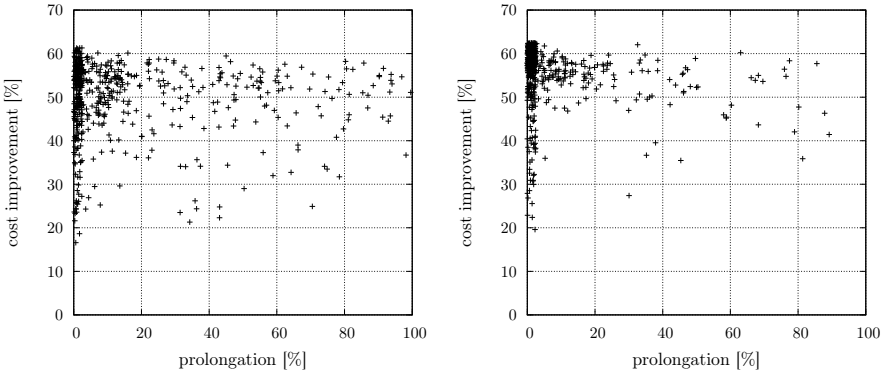


FIGURE 8. Average cost improvement and prolongation against maximum group size (realistic travel demand, trains and coaches).

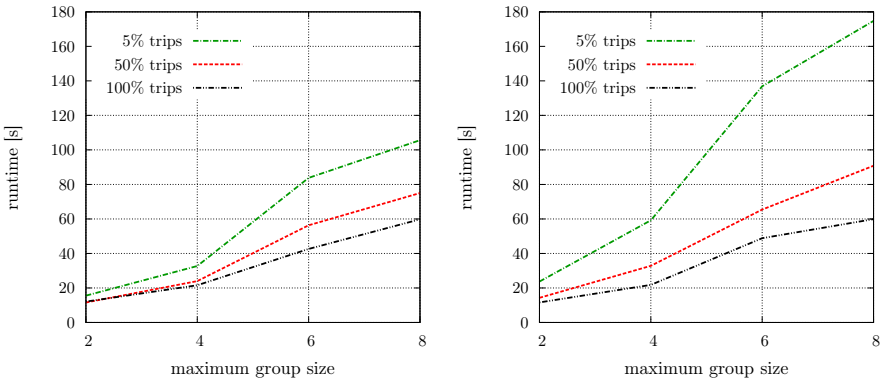


(A) 5% trips (B) 50% trips

FIGURE 9. Cost improvement against prolongation (realistic travel demand, trains and coaches, maximum group size $n_{\max} = 4$).

is that even for small populations of potential ridesharers, there are many shared journeys with a good cost improvement and a reasonable prolongation. In our algorithm, the balance between the two criteria could be calibrated by changing the weights in the cost function.

Computation Time. The first graph in Figure 10 shows the overall computation times of the algorithm for one created group of agents from the realistic demand and a combination of trains and coaches. The trip grouping phase of the algorithm is very fast (200 ms per group on average). The algorithm spends the majority of the computation time solving the problem of finding a joint plan for the group of agents. The graph indicates that the overall computation time grows roughly linearly with increasing numbers of agents in a group, which confirms that the algorithm avoids the exponential blowup in the action space characteristic for centralised multiagent



(A) Realistic travel demand (B) Random travel demand

FIGURE 10. Computation time against maximum group size (trains and coaches).

planning. This is mainly a consequence of the best-response planning algorithm, and an expected result.

The second graph in Figure 10 shows the overall computation times of the algorithm for one created group of agents from the random demand and for a combination of trains and coaches. It can be observed that the algorithm is faster at the realistic 5% ridesharing proportion than for random trips. At the 50% and 100% ridesharing proportion, there is not a very big difference between the computation times. This suggests that the trips from the realistic 5% ridesharing proportion reflects the public transport network making the journey planning easier whereas it is harder to plan for trips distributed randomly.

Regarding the modes of transport in the scenario, it is harder to find joint plans when a combination of trains and coaches is considered (on average, runtimes are 25% higher for scenarios with trains and coaches). Considering a combination of trains and coaches does not significantly affect neither the cost improvement, nor the prolongation.

While the overall computation times are considerable (up to 2 minutes for a group of 8 agents from the realistic 5% ridesharing proportion), we should emphasise that the algorithm is effectively computing equilibrium solutions in multi-player games with hundreds of thousands of states. Considering this, the linear growth hints at having achieved a level of scalability based on the structure of the domain that is far above naive approaches to plan jointly in such state spaces.

Finally, we have evaluated the overall computation time for all trips from the travel demand. We were able to compute shared journey plans for approximately 13,500 trips from realistic 100% ridesharing proportion when considering a combination of trains and coaches. It took less than 75 minutes for each setting of maximum group size while using 8 cores of 3.2 GHz Intel Core i7 processor on three computers in parallel.

The ridesharing algorithm can be further parallelised down to a level of individual groups, bringing the computation time to few minutes for the whole demand. This

follows from the structure of the algorithm: In the trip planning phase, the identification of initial single-agent plans for the travellers consists of a set of completely independent problems. In the best-response and timetabling phase, the planning problem of each group of agents is also completely independent from those of other groups or individual travellers.

6. Discussion

Our proposed algorithm clearly improves the cost of agents' journeys by sharing parts of the journeys, even though there is an inherent trade-off between cost improvement and the prolongation of journeys. On the one hand, the bigger the group, the better the improvement. On the other hand, the more agents share a journey, the higher the prolongation is likely to be. This will most likely lead to results that are not acceptable for users in larger groups. Whether prolongation or cost savings are more important in a given scenario will depend on the real preferences of travellers, and our system would allow them to customise these settings per individual on, for example, a Web-based ridesharing planner that would use our algorithm. It is also important to point out that our framework can be used without any significant modifications for any other cost function as appropriate for the transport system in question, and, subject to availability of the required real-world timetabled transport data, for any other geographical region.

Next, note that trip planning and best-response phases of the algorithm are completely domain-independent and can therefore easily be used for other types of transport problems, e.g., to plan routes that avoid traffic jams or to schedule parcel deliveries. What is more, additional constraints such as staying at a location for some time or travelling together with a specific person can be easily accommodated within standard planning languages, and the use of standard planning technology also implies that our method will directly benefit from future improvements in planning algorithms. On the other hand, the trip grouping and the timetabling phase of the algorithm are domain-specific, providing an example of the specific design choices that have to be made from an engineering point of view when applying standard AI methods to problems of decentralised decision-making in transport.

From an algorithm and systems engineering perspective, using off-the-shelf problem solvers such as AI planning systems for a complex real-world domain like ridesharing brings an additional benefit, which is that we do not need to engineer novel optimisation algorithms for the combinatorial problems arising in this family of problems *from scratch*. While it is certainly possible that faster algorithms that produce better solutions may exist for specific problems, our approach enables us to formalise different types of similar problems with comparatively little effort and to make use of the best available search heuristics in a lightweight fashion. We believe that this is an effective way of developing resource allocation and process optimisation systems in domains like transport, where it can be expensive to develop a custom solution for every different class of problems although many of them share many common characteristics.

The presented experiments work with a demand for a *whole day* and therefore the generated joint plans are not restricted to any particular part of a day. In reality, however, travellers may not be so flexible in terms of timing their journeys. This problem can be easily solved by considering time constraints in the clustering performed in the trip grouping phase of the algorithm. Trips might for instance

be put into one group only if their preferred departure and/or arrival times do not differ by more than a given time difference. The performance achieved for lower trip densities corresponding to 5% ridesharing proportion suggests that attractive shared journeys would be found even for the maximum time difference of one hour (one hour constitutes approximately 4% of a day), or even less during peek hours when demand is more concentrated.

There are many potential uses of the proposed approach to real-world ridesharing: From a traveller's perspective, it can be used to exploit current ticket discounts for group travel while enjoying the company of friends, fellow workers, and other co-travellers. A web- or smartphone-based application can be built which would collect user preferences and constraints and propose shared journey plans. Further, in future applications our approach could be combined with the use of private cars to mix public and private modes of transport. This can be achieved with fairly small modifications. It would work in a similar way as walking is combined with public transport in our evaluation, except that car travel enables non-timetabled transport for more than one individual. This is an important extension, as in most realistic settings, successful ridesharing would certainly include private cars. In fact, without this, we are only considering a very hard problem, were travellers have very limited flexibility. Naturally, if at least one person in each group has a car, this opens up (orders of magnitude) more options for joint trips. Also, for car sharing the cost benefit is arguably much higher, and can be much more objectively calculated than what we have assumed in our hypothetical cost function (e.g., cost/km divided by number of car passengers).

From a public policy and transport planning perspective, stakeholders in the public transport domain could use our method to predict customer behaviour when considering modifications to timetables, the introduction of new services, and modifications to pricing schemes to optimise usage, environmental footprint, and business revenue. Such scenario analysis could easily accommodate taking further factors into account, such as waiting times, travel interruptions for business and leisure activities, preferences of individuals to share trips with particular co-travellers, etc. In particular, it could give rise to new incentive schemes for ridesharing, such as discounts for group travel that depend on the cumulative amount of sharing or occupancy ratios along different legs of joint journeys involving various modes of transport and changing groups of jointly travelling individuals.

7. Conclusion

We have presented a multiagent planning algorithm which is able to plan meaningful shared journeys using timetabled public transport services. The algorithm has been implemented and evaluated on realistic scenarios based on real-world UK transport data. Experiments with realistic travel demand show that, for a wide range of scenarios, the algorithm is capable of finding shared journeys with very attractive trade-offs between cost saving and journey time prolongation.

The algorithm exhibits very good scalability, scaling linearly with the number of trips processed, regardless of the size of travel groups considered. The algorithm is also amenable to massive parallelisation which can bring the time required for planning shared journeys for real-world travel demand down to minutes.

Finally, the cost of travel and flexibility of ridesharing can be significantly improved by sharing private cars. In the future, we plan to extend the algorithm

towards multi-modal ridesharing in which groups of travellers can seamlessly transfer between timetabled and non-timetabled transport modes.

8. Acknowledgments

This work was supported by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067), by the Ministry of Education, Youth and Sports of Czech Republic (grant no. 7E12065) and by the Czech Technical University (grant no. SGS13/210/OHK3/3T/13).

References

Berbeglia, G., Cordeau, J. F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

Boutilier, C. and Brafman, R. (2001). Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136.

Brafman, R. and Domshlak, C. (2008). From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *Procs. ICAPS 2008*, pages 28–35. AAAI Press.

Brafman, R., Domshlak, C., Engel, Y., and Tennenholtz, M. (2009a). Planning Games. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 21, pages 73–78.

Brafman, R., Domshlak, C., Engel, Y., and Tennenholtz, M. (2009b). Planning Games. In *Procs. IJCAI 2009*, pages 73–78.

Brodal, G. S. and Jacob, R. (2004). Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. *Electronic Notes in Theoretical Computer Science*, 92(0):3–15.

Coles, A. J., Coles, A. I., Fox, M., and Long, D. (2011). POPF2: a Forward-Chaining Partial Order Planner. In *Procs. IPC-7*.

Cordeau, J. F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals OR*, 153(1):29–46.

Cox, J. and Durfee, E. (2005). An efficient algorithm for multiagent plan coordination. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 828–835, Utrecht, The Netherlands.

Cox, J. and Durfee, E. (2009). Efficient and distributable methods for solving the multiagent plan coordination problem. *Multiagent and Grid Systems*, 5(4):373–408.

de Weerdt, M. and Clement, B. (2009). Introduction to planning in multiagent systems. *Multiagent and Grid Systems*, 5(4):345–355.

Delling, D., Pajor, T., and Wagner, D. (2009). Accelerating Multi-modal Route Planning by Access-Nodes. In Fiat, A. and Sanders, P., editors, *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 587–598. Springer.

Dimopoulos, Y., Hashmi, M. A., and Moraitis, P. (2012). μ-satplan: Multi-agent planning as satisfiability. *Knowledge-Based Systems*, 29(0):54 – 62.

Ephrati, E., Pollack, M. E., and Rosenschein, J. S. (1995). A Tractable Heuristic that Maximizes Global Utility through Local Plan Combination. In *In Proceedings of the First International Conference on MultiAgent Systems (ICMAS-95)*, pages 94–101.

Eurostat ([Online], available at tinyurl.com/eurostat-modal-split, [Accessed: Oct 26, 2012]). Modal split of passenger transport.

Foulser, D. E., Li, M., and Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence*, 57(2-3):143–181.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Horn, M. (2002). Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188.

Hrnčíř, J. (2011). Improving a Collaborative Travel Planning Application. Master's thesis, The University of Edinburgh.

Hrnčíř, J. and Rovatsos, M. (2012). Applying Strategic Multiagent Planning to Real-World Travel Sharing Problems. In *7th Workshop on Agents in Traffic and Transportation, AAMAS*.

Hsu, C. and Wah, B. W. (2008). The SGPlan Planning System in IPC-6. In *Procs. IPC-6*.

Jonsson, A. and Rovatsos, M. (2011). Scaling Up Multiagent Planning: A Best-Response Approach. In *Procs. ICAPS 2011*, pages 114–121. AAAI Press.

Monderer, D. and Shapley, L. S. (1996). Potential Games. *Games and Economic Behavior*, 14(1):124–143.

Nissim, R., Brafman, R., and Domshlak, C. (2010). A general, fully distributed multi-agent planning algorithm. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 9, pages 1323–1330.

Office for National Statistics (2001). Special Travel Statistics (Level 1). [computer file], ESRC/JISC Census Programme, Census Interaction Data Service, University of Leeds and University of St. Andrews.

Pajor, T. (2009). Multi-Modal Route Planning. Master's thesis.

Pyrga, E., Schulz, F., Wagner, D., and Zaroliagis, C. (2008). Efficient models for timetable information in public transportation systems. *Journal of Experimental Algorithmics (JEA)*, 12.

Richter, S. and Westphal, M. (2008). The LAMA planner. Using landmark counting in heuristic search. In *Procs. IPC-6*.

Schulz, F. (2005). *Timetable information and shortest paths*. PhD thesis.

Torreno, A., Onaindia, E., and Sapena, O. (2012). An approach to multi-agent planning with incomplete information. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'12)*.

Tsamardinos, I., Pollack, M. E., and Horty, J. F. (2000). Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In Chien, S., Kambhampati, S., and Knoblock, C. A., editors, *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 264–272.

van der Krogt, R., Weerdt, M. D., and Zhang, Y. (2008). Of Mechanism Design and Multiagent Planning. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 423–427. IOS Press.

Wu, Y., Guan, L., and Winter, S. (2008). Peer-to-peer shared ride systems. *GeoSensor Networks*, pages 252–270.

Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University, 121 35 Prague, Czech Republic
E-mail address: hrncir@agents.fel.cvut.cz

School of Informatics, The University of Edinburgh, Edinburgh EH8 9AB, United Kingdom
E-mail address: mrovatso@inf.ed.ac.uk

Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University, 121 35 Prague, Czech Republic
E-mail address: jakob@agents.fel.cvut.cz

Appendix G

Extending Security Games to Defenders with Constrained Mobility

O. Vaněk, B. Bošanský, M. Jakob, V. Lisy, and M. Pěchouček. Extending security games to defenders with constrained mobility. In *AAAI Spring Symposium: Game Theory for Security, Sustainability, and Health*, 2012.

AAAI Technical Report SS-12-03
Game Theory for Security, Sustainability and Health

Extending Security Games to Defenders with Constrained Mobility

Ondřej Vaněk, Branislav Bošanský, Michal Jakob, Viliam Lisý and Michal Pěchouček
Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University
Technická 2, 16627 Praha 6, Czech Republic
{vanek, bosansky, jakob, lisy, pechoucek}@agents.fel.cvut.cz

Abstract

A number of real-world security scenarios can be cast as a problem of transiting an area guarded by a mobile patroller, where the transiting agent aims to choose its route so as to minimize the probability of encountering the patrolling agent, and vice versa. We model this problem as a two-player zero-sum game on a graph, termed the *transit game*. In contrast to the existing models of area transit, where one of the players is stationary, we assume both players are mobile. We also explicitly model the limited endurance of the patroller and the notion of a base to which the patroller has to repeatedly return. Noting the prohibitive size of the strategy spaces of both players, we develop single- and double-oracle based algorithms including a novel acceleration scheme, to obtain optimum route selection strategies for both players. We evaluate the developed approach on a range of transit game instances inspired by real-world security problems in the urban and naval security domains.

Introduction

Hostile area transit and patrolling is an important problem relevant to many real-world security scenarios, such as illegal border crossing, smuggling interdiction or transport logistics in insecure regions (Gilpin 2009). For the transiting agent, the problem is to choose such a route to get across the hostile transit area that minimizes the risk that it will be encountered and intercepted by the patrolling agent that moves within the area; the objective of the patrolling agent is the opposite.

Choosing the optimum routes becomes non-trivial if we assume that the agents are aware of and capable of reasoning about each other's objectives and/or whenever the situation happens repeatedly and the agents are able to learn from experience. In both such cases, predictability is disadvantageous as it can be exploited by the opponent agent. Randomized route selection can be used in these cases to make exploitation more difficult by increasing the uncertainty in opponent agent's behavior. Game theory provides a principled way of achieving optimum randomization, taking into account information, preferences and constraints of both agents.

In this paper, we model the area transit and patrolling problem as a zero-sum game, termed *transit game*, between two players – the Evader and the Patroller. Pure strategies of both players correspond to routes in/through the transit area and the utilities are directly related to the probability that the Patroller will intercept the Evader.

Game-theoretic approach has been successfully applied to similar problems in the past (Jain et al. 2010b), resulting in a variety of games reflecting specific assumptions, domain restrictions and player capabilities. In contrast to existing models of area transit, where one of the players is always stationary, our transit game assumes *both* players are mobile; (Vaněk et al. 2010) is the sole exception but see Related work section for differentiation. Unlike existing models, we also allow to consider Patroller's home base and maximum endurance, both features frequently present and required in real-world scenarios. Our objective is to find such a strategy for the players that maximizes the minimum expected utility the players can obtain, which — thanks to the zero-sum property of the transit game — corresponds to a Nash equilibrium of the game.

Unfortunately, the mobility of both players triggers combinatorial explosion in the number of possible strategies and makes the application of standard methods for finding Nash equilibria ineffective. For example, in a rectangular grid graph with 15 nodes, the number of possible strategies exceeds 10^{10}, rendering standard approaches for computing Nash equilibria in normal-form games inapplicable. We therefore employ iterative solution techniques known as *oracle-based algorithms* (Barnhart et al. 1994; McMahan, Gordon, and Blum 2003) which do not require explicit enumeration of strategies for the players. Unfortunately, although the oracle-based approach alleviates the problem to some extent, it requires repeated best response calculation, which is hard in our case. We therefore propose a novel variant of the oracle algorithms, termed *accelerated oracle*, which reduces the need for best response calculation, and thus speeds up the calculation of Nash equilibria.

We evaluate our approach on two classes of transit games, directly inspired by real-world security applications — regular grid graphs suitable for modeling transit through open areas and irregular planar graphs suitable for modeling transit through structured environments. For the former — by employing agent-based simulation of maritime piracy

AgentC (Jakob, Vaněk, and Pěchouček 2011) — we conduct experiments that transcend the game-theoretic framework and allows us to validate the transit game model from a broader perspective and compare its effectiveness to currently deployed solutions in the maritime domain.

Related Work

Game-theoretic framework has been applied to a wide range of strategic problems where one of the players — termed *evader* — wants to minimize the probability of being detected and/or intercepted, while the other player — termed *patroller* — wants to maximize the probability of detecting the first player and/or thwarting its plans.

We can distinguish several classes of such games: *ambush games* (Ruckle et al. 1976) and *interdiction games* (Washburn and Wood 1995) model an immobile intercepting player selecting static ambush points in an area represented by a graph that the evading player tries to transit, recently extended e.g. in (Dickerson et al. 2010; Jain et al. 2011). The mobility of the players is reversed in *search games* (Gal 1980) where the evader is stationary and the searching player is mobile. In hider-seeker games (Flood 1972), both players are mobile; however the players have no explicit restrictions on their trajectories and do not have additional goals.

Close to our model are *infiltration games* (Alpern 1992) in which both the players are mobile and the evader tries to cross a given area between defined entry and exit, however the patrolling player can move freely and does not have a base to return to. We further enrich this model by associating interception probability with each node and edge in the graph (similar concept was used e.g. in (Brooks, Schwier, and Griffin 2009)).

The techniques used to find solution of the transit game are inspired by single- and double-oracle algorithms; the former usually termed *column/constraint generation* or *branch and price* (Barnhart et al. 1994), used among others for solving large-scale games (Jain et al. 2010a), extended to double-oracle approach (McMahan, Gordon, and Blum 2003), used e.g. in (Halvorson, Conitzer, and Parr 2009).

The model of the transit game was first introduced in (Vaněk et al. 2010). We extend the model by (1) allowing the Evader to move freely on an arbitrary graph and by (2) allowing the positioning of the Patroller's base anywhere in the environment. Moreover, we redefine the utility of the game so that the relative direction of player's movement is taken into account, which results into a well-defined, probabilistic interpretation of the game value.

Problem Definition

We formalize the problem of hostile area transit and patrolling as follows: let us have a connected *transit area* with defined *entry* and *exit* zones and a *base* location. There are two players that move in the area: the Evader and the Patroller. The Evader's objective is to get from any location in the entry zones to any location in the exit zones *without* encountering the Patroller. The Patroller's objective is to intercept the Evader's transit by strategically moving through the transit area. In addition, because of its limited endurance, the Patroller has to repeatedly return to the base.

Modeling Assumptions

We make the following assumptions regarding the area transit problem: (1) both players have full knowledge about the topology of the transit area, including the location of Patroller's base and the location of entry and exit zones. (2) The Evader knows the Patroller's capability to detect and intercept the Evader at any position in the transit area (this capability can vary due to environmental reasons). (3) The Evader knows Patroller's maximum endurance (and consequently the limits on the maximum length of Patroller's walks). (4) The players have no information about the location of the other player, unless they meet (at which point the game ends). (5) The Patroller has no information of whether the Evader has already entered the area. The Evader does not know when the Patroller visits the base.

Transit Area Representation

We use discrete representation of space. We represent the transit area as a simple directed graph with loops termed *transit graph* $G(N, E)$, $N = \{1, 2, \ldots, n\}$ denotes a set of nodes represented directly by natural numbers, and $E = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$ where $i_k \in N \wedge j_k \in N$ is the set of edges defining legal movement of players through the transit area. Every edge has a unit length, i.e., each step (see below) the player can move from one node to any adjacent node. Three special types of nodes are defined on the transit graph:(1) *entry nodes* N_{in} – the Evader can start its path in any node of this type; (2) *exit nodes* N_{out} – the Evader aims to reach any node of this type; (3) *base node* n_b – the node in which all Patroller's walks start and end.

Player Movement

Analogously to space, time is also discretized. The movement of both players happens simultaneously in synchronized *steps*. During each step, a player can move to an adjacent node or stay in the same node. Both players have the same movement speed and all edges take a single step to traverse. Player's movement through the transit graph can be unambiguously represented by a node *walk*, i.e., a sequence of nodes $w = [n_0, n_1, \ldots, n_k]$[1]; we then denote $|w|$ the length of walk w, and $w[j]$ the j-th node on the walk (for $0 \leq j \leq |w| - 1$).

The following will be required for the definition of utilities. For any finite walk w, we define *infinite walk repetition* $w^{\infty}[i] = w[i \bmod |w|]$ and *shifted infinite walk repetition* $w^{\infty \triangleright m} = w[(i - m) \bmod |w|]$. E.g. for $w = [1, 4, 7]$, we have

index	…	[−2]	[−1]	[0]	[1]	[2]	[3]	[4]	…
w	…	-	-	1	4	7	-	-	…
w^{∞}	…	4	7	1	4	7	1	4	…
$w^{\infty \triangleright 1}$	…	1	4	7	1	4	7	1	…

[1] In a slight abuse of common mathematical notation and to emphasize similarity with the array data structure, we use brackets to denote sequences and to index sequence items.

Encounters We say two walks w_1 and w_2 have:

- a *node encounter* at node $i \in N$ at step t if $i = w_1[t] = w_2[t]$ (being at the same node at the same time step);
- an *edge encounter* at edge $(i, j) \in E$ at step t if $i = w_1[t] = w_2[t] \wedge j = w_1[t+1] = w_2[t+1]$ (traveling the same edge simultaneously in the same direction) or $i = w_1[t] = w_2[t+1] \wedge j = w_1[t+1] = w_2[t]$ (traveling the same edge simultaneously in the opposite directions)
- an *encounter* at *location* $l \in N \cup E$ at step t if w_1 and w_2 either have a node encounter or an edge encounter at l at step t.

The *encounter sequence* of two walks w_1 and w_2 is a sequence $[(l_0, t_0), (l_1, t_1), \ldots, (l_n, t_n)]$ where $\forall i \in \{0 \ldots n\}$ w_1 and w_2 have an encounter at l_i at step t_i and $t_i \leq t_{i+1}$, i.e. the order in which the encounter locations appear on walks w_1 and w_2 is preserved; the *encounter location sequence*, denoted as $w_1 \cap w_2$, is then the encounter sequence without timestep indices but with the ordering preserved – $[l_0, l_1, \ldots, l_n]$.

As an example, let us consider $w_1 = [1, 4, 7, 6, 9, 2, 5, 3]$ and $w_2 = [1, 4, 8, 6, 2, 9]$.

index	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
w_1	1	4	7	6	9	2	5	3
w_2	1	4	8	6	2	9	-	-
$w_1 \cap w_2$	1	(1,4)	4		6	(2,9)		

The resulting encounter location sequence $w_1 \cap w_2 = [1, (1, 4), 4, 6, (2, 9)]$.

Interceptions To provide more expressiveness to the transit game model, encounters are assumed to lead to interceptions only with a defined location-specific *interception probability* $p(l)$ $\forall l \in N \cap E$. Interception probability may e.g. reflect the likelihood that the Patroller will be able to detect the Evader and/or will be able to physically apprehend the Evader in a given location. The concept is similar to the concept of sensors with probability of detection in (Brooks, Schwier, and Griffin 2009).

Transit Game

The modeling assumptions allow us to model the transit problem as a normal-form game between two players – the Evader and the Patroller. In the following, we assume the transit game takes place on a transit graph $G = (N, E)$ with entry nodes N_{in}, exit nodes N_{out} and base n_b.

Strategy Spaces The set S_E of all possible pure Evader's strategies is the set of all walks starting in an entry node and ending in an exit node, with the nodes in between not being an entry or exit node, i.e.,

$$S_E = \{[n_0, \ldots, n_m] | n_0 \in N_{\text{in}} \wedge n_m \in N_{\text{out}} \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\} \forall i = \{1, \ldots, m-1\}\} \quad (1)$$

Note that because in general Evader's walks are unlimited, the above set can be infinite, however we will limit the Evader to visit every node at most once.

The set S_P of all possible pure Patroller's strategies is the set of all closed walks starting and ending in the base with length not exceeding L_P, i.e.,

$$S_P = \{[n_0, \ldots, n_m] | m \leq L_P \wedge n_i \in N \wedge n_0 = n_b \wedge (n_m, n_b) \in E\} \quad (2)$$

Note that if L_P is so small that it does not allow the Patroller to cross one of the Evader's walks, then the Evader will have a deterministic strategy that guarantees safe transit of the graph. The threshold for L_P under which this is the case depends on the position of Patroller's base and topology of the transit graph. In contrast to the Evader, which performs its walk only once, the Patroller executes its walk repeatedly[2].

We further denote Σ_E and Σ_P the set of all mixed strategies of the Evader and the Patroller, respectively.

Utility Functions The utility function in the game is directly related to the probability of interception. First, given an encounter location sequence I, we can express the probability $\pi(I)$ that the Evader will be intercepted by the Patroller as

$$\pi(I) = \sum_{i=0}^{|I|-1} p(I[i]) \prod_{j=0}^{i-1} (1 - p(I[j])) \quad (3)$$

where

$$p(I[i]) \prod_{j=0}^{i-1} (1 - p(I[j])) \quad (4)$$

is the probability that the Patroller will not intercept the Evader at locations $I[0], \ldots, I[i-1]$ and will intercept it at location $I[i]$.

To calculate the interception probability $\pi(s_E, s_P)$ of a pair of pure strategies $(s_E, s_P) \in S_E \times S_P$, we need to determine all possible encounter location sequences that can result from executing these strategies. Recalling that the Patroller has no knowledge on when the Evader enters the transit area, we have to consider all possible mutual shifts of Evader's and Patroller's walks; however, because Patroller's walk s_P is perpetually repeated, we only need to consider $|s_P|$ shift. The interception probability can therefore be calculated as

$$\pi(s_E, s_P) = \frac{1}{|s_P|} \sum_{i=0}^{|s_P|-1} \pi(I^{\triangleright i}) \quad (5)$$

where

$$I^{\triangleright i} = s_E \cap s_P^{\infty \triangleright i} \quad (6)$$

For a given pure strategy pair $(s_E, s_P) \in S_E \times S_P$, we now define the Patroller's utility $u_P(s_E, s_P)$ as equal to the interception probability, i.e.,

$$u_P(s_E, s_P) = \pi(s_E, s_P) \quad (7)$$

We define the Evader's utility as the opposite value of Patroller's utility, i.e.,

$$u_E(s_E, s_P) = -\pi(s_E, s_P) \quad (8)$$

[2]Patroller does not repeats the walk indefinitely, the game ends once the Evader reaches one of the exit nodes.

Finally, we define the *transit game with deterministic encounters* as a transit game with unit interception probabilities at all nodes and edges; i.e., $p(l) = 1 \; \forall l \in N \cup E$.

Solution

We employ mixed-strategy Nash equilibrium (NE) as a solution concept for the transit game. However, because of the enormous size of the strategy spaces of both players, standard techniques for computing a NE of normal form games, requiring the construction of the full game matrix, are not applicable. We thus employ iterative techniques known as *oracle-based algorithms* (McMahan, Gordon, and Blum 2003). In the following, we first describe the iterative-oracle-based search in general, independent of any particular normal-form game to which it is applied. The specifics of the transit game only need to be considered when formulating specific oracles, used in applying the oracle algorithms to the transit game.

Iterative Oracle-based NE Computation

Instead of searching for the NE of the full normal-form game, oracle-based algorithms iteratively construct and solve a growing succession of (significantly) smaller subgames until they reach a subgame whose NE is also a NE of the full game. Depending on the structure of player's strategy spaces, a NE of the full game maybe found (long) before the full game needs to be constructed and solved. Assuming the computation of NE is significantly faster for the much smaller subgames than for the full game, this may lead to overall significantly faster computation.

Single-Oracle Algorithm Let us consider a two-player normal-form zero-sum game Γ with pure strategy sets S_1 and S_2 for Player 1 and Player 2, respectively, and the corresponding mixed strategy sets Σ_1 and Σ_2. The single-oracle algorithm iteratively constructs a sequence of subgames $[\Gamma^{(0)}, \Gamma^{(1)}, \ldots]$ where each game $\Gamma^{(k)}$ consists of the complete pure strategy set S_1 for Player 1 but only a subset $\hat{S}_2^{(k)} \subseteq S_2$ of the full pure strategy set S_2 for Player 2. In each iteration of the single-oracle algorithm, a Nash equilibrium $(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$ of the current subgame $\Gamma^{(k)}$ is first sought using standard linear program approach. Next, a Player's 2 *best-response oracle* $\omega^* : \Sigma_1 \mapsto S_2$ is consulted to obtain a pure best-response strategy $s_2 \in S_2$ of Player 2 against the Player's 1 strategy $\sigma_1 \in \Sigma_1$; if the resulting pure strategy $s_2 \in S_2$ is already in $\hat{S}_2$, the algorithm terminates and the NE (σ_1, σ_2) is the NE of the full game Γ (McMahan, Gordon, and Blum 2003); otherwise, the strategy s_2, now termed *subgame expanding strategy*, is added to $\hat{S}_2$ and the algorithm continues.

In the ideal case, the iterative oracle-based algorithm would only add such pure strategies $s \in S_2$ to the pure strategy subset $\hat{S}_2$ that are in the *support*[3] of the Player's 2 resulting mixed NE strategy of the full game Γ. Best response calculation can be viewed as a heuristic for selecting such subgame expanding pure strategies $s_2 \in S_2$ that the algorithm terminates after as few iterations as possible.

[3]The *support* of a mixed strategy σ_i is a set of pure strategies $\{s_i | \sigma_i(s_i) > 0\}$.

Algorithm 1 Accelerated Single-Oracle Algorithm. Heuristic and best-response oracle denoted as ω and ω^*, respectively.

```
equilibrium_found ← false
Ŝ_2 ← ∅
σ_1 ← uniform distribution over all s_1 ∈ S_1
repeat
  s ← ω(σ_1)
  if s ∉ Ŝ_2 then
    Ŝ_2 ← Ŝ_2 ∪ {s}
  else
    s* ← ω*(σ_1)
    if (s* ∈ Ŝ_2) then
      equilibrium_found ← true
    else
      Ŝ_2 ← Ŝ_2 ∪ {s*}
    end if
  end if
  (σ_1, σ_2) ← compute NE using LP for S_1 and Ŝ_2
until equilibrium_found
```

Accelerated Single-Oracle Algorithm The above reasoning paves the way for using alternative methods of selecting subgame expanding strategies. In principal, there can be two reasons for doing so: (1) best response calculation is too expensive to be invoked in each cycle and (2) the alternative selection method may navigate the space of subgames more effectively, resulting in a lower number of iterations. In either case, we term the oracle that returns a subgame expanding strategy the *subgame expansion oracle* (denoted as ω) and the resulting method, which uses two distinct oracles, the *accelerated oracle* algorithm.

Pseudocode of the accelerated oracle algorithm is given in Algorithm 1. Note that we use the heuristic oracle ω to obtain game expanding strategies; the termination condition still uses the best response as only this ensures that the NE obtained on the current subgame $\Gamma^{(k)}$ is an NE of the full game Γ too.

Theorem 1 *If pure strategy sets of both players are finite, the Algorithm 1 with accelerated oracle finds the Nash equilibrium of the full transit game.*

Proof 1 *At the worst case, the heuristic oracle of Player 2 adds all pure strategies to the Player's 2 strategy subset and the final subgame equals to the full game. If the algorithm terminated* before *all Player's 2 pure strategies were enumerated, then the best response for the current mixed strategy σ_1 of Player 1 had to be already in Player's 2 strategy subset $\hat{S}_2$. However, this is the termination condition of the original oracle algorithm for which (see (McMahan, Gordon, and Blum 2003)) show that it is satisfied only if the NE of the subgame corresponds to the NE of the full game. Hence, this is also true for the accelerated oracle algorithm.*

Double-Oracle Algorithm The double-oracle algorithm (Algorithm 2) uses incrementally expanded strategy subsets

Algorithm 2 Accelerated Double-Oracle Algorithm. Best-response oracles for Player 1 and Player 2 denoted as ω_1^* and ω_2^*, respectively. Heuristic oracle for Player 1 denoted as ω_1.

```
equilibrium_found ← false
Ŝ1 ← { arbitrary strategy s1 ∈ S1 }
Ŝ2 ← { arbitrary strategy s2 ∈ S2 }
repeat
    (σ1, σ2) ← compute NE using LP for Ŝ1 and Ŝ2
    s1 ← ω1(σ2)
    s2 ← ω2*(σ1)
    if (s1 ∈ Ŝ1) ∧ (s2 ∈ Ŝ2) then
        s1* ← ω1*(σ2)
        if s1* ∈ S1 then
            equilibrium_found ← true
        else
            Ŝ1 ← Ŝ1 ∪ {s1*}
        end if
    else
        Ŝ1 ← Ŝ1 ∪ {s1}
        Ŝ2 ← Ŝ2 ∪ {s2}
    end if
until equilibrium_found
```

$\hat{S}_1$ and $\hat{S}_2$ for both players. Termination condition requires that best responses for both players, computed by best response oracula ω_1^* and ω_2^*, are already present in the respective strategy subsets. Analogously to the single-oracle algorithm, distinct subgame expansion oracles can be used for subgame expansion as long as the best response oracula ω_1^* and ω_2^* are used in the termination condition. For the sake of simplicity and because of the way the double-oracle algorithm is employed for solving the transit game, Algorithm 2 describes a variant where the subgame expansion oracle is only employed for Player 1.

Transit Game Oracles

We now describe the implementation of the specific oracles used in solving the transit game. The Evader and the Patroller correspond to Player 1 and Player 2, respectively. Correspondingly, we further use E and P instead of 1 and 2 to index players. The definition of utility (see Equations 5 and 7) — taking into account the relative directions of the paths — prohibits the direct formulation of a best-response as an (mixed-integer) linear program. The oracles thus employ standard search techniques to provide the best response for each player.

Evader's Best-Response Oracle In a given transit game, the Evader's best response oracle ω_E^* provides a pure strategy $s \in S_E$ which is the Evader's pure-strategy best response to Patroller's mixed strategy $\sigma_P \in \Sigma_P$; in other words, such a walk $s_E^* \in S_E$ from any entry node to any exit node which minimizes the Patroller's expected utility

$$s_E^* = \arg\min_{s_E \in S_E} \sum_{s_P \in S_P} u(s_E, s_P) \cdot \sigma_P(s_P) \qquad (9)$$

where $\sigma_P(s_P)$ is the probability that $s_P \in S_P$ is played in σ_P.

The Evader's best-response oracle ω_E^* is implemented as a *best-first-search* algorithm, starting from all entry nodes and expanding the best walk so far. To avoid infinite-length walks, we require that the same node is not visited multiple times. Partial-walk utility for each incomplete walk is computed at each step and the best walk (i.e. the one with the highest partial-walk utility) is chosen for further expansion. If this best walk is complete (i.e. going from an entry node all the way to an exit node), the algorithm terminates and returns the walk. Because of the utility formulation (Equation 8), the Evader's utility of an Evader's walk cannot increase with the walk's expansion[4], thus the first complete walk is guaranteed to be the best response for the Evader.

Evader's Subgame Expansion Oracle Due to the combinatorial explosion in the number of possible Evader's walks, best response calculation for the Evader is expensive. For subgame expansion, we therefore employ a distinct subgame expansion oracle ω_E which determines the subgame expanding strategy by searching for the best response $s \in S_E$ only in the set of Evader's walks of a limited maximum length. The optimal maximum length depends on the structure of the transit graph; as a reasonable estimate, the length of the shortest path between the two most distant entry and exit nodes can be used.

Patroller's Best-Response Oracle Patroller's best-response oracle ω_P^* provides a pure strategy $s^* \in S_P$ which is the Patroller's pure strategy best response to an Evader's mixed strategy $\sigma_E \in \Sigma_E$., i.e., a bounded-length closed walk from Patroller's base maximizing the expected Patroller's utility

$$s_P^* = \arg\max_{s_p \in S_P} \sum_{s_E \in \mathcal{S}_E} u(s_E, s_P) \cdot \sigma_E(s_E) \qquad (10)$$

Computation-wise, the Patroller's best-response oracle ω_P^* is implemented as a branch-and-bound depth-first search.

Evaluation

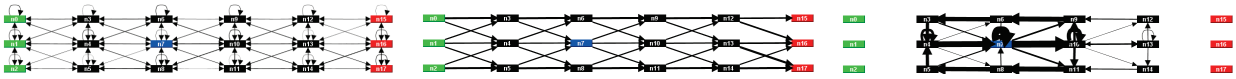
We present the evaluation of our approach on two characteristic and application-relevant classes of transit games. We begin with example solutions and then compare the properties of individual solution algorithms in terms of performance, scalability and convergence. Finally, we validate the game-theoretic strategies on an agent-based simulator of maritime piracy.

Test Problems

We study the properties of the transit game and its solution on two types of graphs, motivated by real-world domains: (1) rectangular grid graphs and (2) planar city graphs. For each graph, we examine the properties of the proposed algorithms, both for a transit game with deterministic and non-deterministic encounters.

A grid graph with cycles and diagonal edges (depicted on the Figure 1a) is a suitable representation of a open transit

[4]Note that the complete-walk utility will be always lower than or equal to the partial-walk utility so we can use the partial-walk utility for the *best-first* estimate.



(a) Regular 6x3 grid graph. (b) Evader's optimal strategy. (c) Patroller's optimal strategy.

Figure 1: Exemplar transit game on a rectangular grid graph with deterministic encounters; three entry nodes placed to the left, three exit nodes placed to the right and Patroller's base positioned in the middle. The final game value is 0.327, i.e. giving the Patroller a chance of 32.7% to intercept the Evader.

areas (such as desert, forest or sea.). We denote the longer and shorter side as *length* and *width*, respectively. In this evaluation, we place the entry and exit locations on the opposite shorter sides of the grid and the base in the middle.

As a representative of planar city graphs, we use a road network graph extracted from GIS data of part of Prague (see Figure 2a).

Example Solutions

The solution of the transit game with deterministic encounters on the rectangular grid graph (of width $w = 3$, length $l = 6$ and base $b = 7$) for the Evader and Patroller is presented in the form of a probability distribution over edges on Figures 1b and 1c, respectively. The weight of an edge is the probability of the respective player traversing that edge if it plays its optimal mixed strategy. Note that, in the transit games with deterministic encounters, the resulting Evader strategy contains only forward edges (i.e. edges $\{(i,j)|\ d(j,o) < d(i,o),\ o \in N_{out}\}$[5]). The solution of the transit game on the city graph is depicted on Figures 2b and 2c.

Note that all described variants of oracle algorithms provide the same solutions on the above test problems. In contrast, standard solution approach working with full strategy sets for both players was not applicable on a standard desktop computer due to the size of the resulting linear program.

Computation Time

Although all described oracle algorithms provide the same solution, the time required to compute the solution differs. We evaluate the following three variants of oracle-based algorithms:

1. *Evader's Single Oracle* (*ESO*) is the standard single-oracle algorithm utilizing only the Evader's best-response oracle for the Evader.
2. *Evader's Accelerated Single Oracle* (*ESO-A*) is the accelerated single-oracle algorithm utilizing the Evader's subgame expansion oracle.
3. *Accelerated Double Oracle* (*DO*) is the accelerated double oracle algorithm utilizing Evader's subgame expansion oracle, and Patroller's best-response oracle.

Note that the single-oracle algorithm utilizing Patroller's best-response oracle could not be used since the size of the set of all possible Evader paths (that are needed to be enumerated completely) quickly exceeds the memory limits.

[5] $d(j,o)$ denotes Chebyshev/L_∞ distance and N_{out} is the set of exit nodes.

Tables 1a and 1b summarize the computation times for the algorithms on both types of graphs and with both deterministic and non-deterministic encounters. The maximum length of Patroller's walk was $L_P = 7$. The Evader's accelerated single oracle *ESO-A* outperforms both the Evader's standard single oracle *ESO* and the double oracle *DO* on all test instances. Interesting difference in performance can be observed when solving games on grid and city graphs. The number of all possible Patroller's walks is 24463 for the grid graph and only 185 for the city graph; this inequality is reflected in runtimes of *DO* (employing Patroller's oracle). On the grid graph, *DO* needs significantly more time to find a NE than the single-oracle-based algorithms. However, on the city graph, *DO* is not penalized by using the Patroller's oracle; in this case, the computation time is highest for *ESO*.

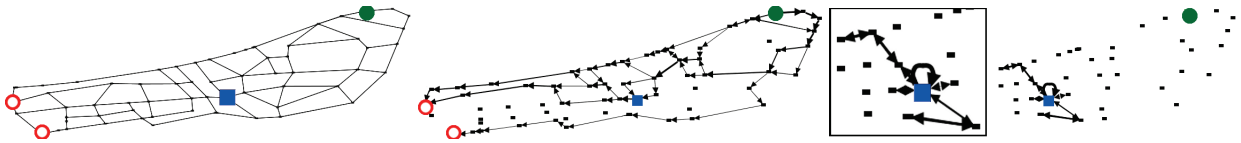
Note the higher number of iterations for *DO* compared to *ESO-A*. Although both *DO* and *ESO-A* employ Evader's heuristic oracle, due to the incomplete enumeration of the Patroller's strategies, *DO* needs more iterations to converge. For the single-oracle algorithms, the time spend on linear programs computing NE of the subgames is higher than for the double-oracle algorithm due to the fact that in the case of single-oracle algorithms, the generated subgames contain all Patroller's strategies (i.e. all closed walks from the base) and the resulting linear programs are significantly larger.

We explore the scalability of the fastest algorithm (*ESO-A*) on set of rectangular grid graphs of various width and length. The runtime of the algorithm depends both on the size of the graph and the maximum length of Patroller's path, which was varied from 6 to 8. See the results in Figure 3. In general, the limits of *ESO-A* were hit because of the Evader's subgame expansion oracle which turned up unable to find a subgame expanding strategy on a grid graph 12x4 with $L_P = 8$.

Convergence and Approximation

We have studied the dynamics of the proposed oracle algorithms with the ambition of utilizing the knowledge of their convergence for trading time for optimality and producing approximate solutions.

The speed of convergence of *ESO-A* and the time needed to obtain a solution within required bounds is depicted on the Figure 4. We denote the difference between the value of the response provided by the oracle and the current subgame value as δ and the final game value as $\mathcal{V}^*$. The y-axis shows the ratio $\epsilon = \delta/\mathcal{V}^*$ which serves as the error upper bound, if we terminate the algorithm at given time (x-axis).



(a) Exemplar planar city graph. Loops and interception probabilities omitted.

(b) Evader's optimal strategy.

(c) Patroller's optimal strategy.

Figure 2: Exemplar transit game on a planar city graph with non-deterministic encounters. The graph contains 66 nodes and 100 edges and has one entry node in the eastern part of the graph (depicted as the full green circle), two exit nodes in the western part of the graph (depicted as empty red circles) and the base (depicted as the blue square) in the middle. The final game value is 0.146, i.e. giving the Patroller a chance of 14.6% to intercept the Evader.

Graph	Alg	Iters	Time	EO	PO	CLP
Grid 4x6	ESO	36	7.2	3.5	1.1	2.5
	ESO-A	33	6.2	2.7	1.1	2.4
	DO	60	91.1	2.7	88.3	0.01
City	ESO	13	76.7	76.6	0.1	0.04
	ESO-A	14	18.7	18.6	0.07	0.04
	DO	16	29.8	28.8	0.5	0.06

(a) Transit game with deterministic encounters.

Graph	Alg	Iters	Time	EO	PO	CLP
Grid 4x6	ESO	29.0	37.5	34.1	1.1	2.3
	ESO-A	30.9	6.9	3.5	1.2	3.2
	DO	51.9	178.6	5.7	172.56	0.09
City	ESO	18.8	177.2	177.0	0.04	0.06
	ESO-A	11.7	29.3	29.3	0.01	0.03
	DO	14.3	37.5	37.0	0.3	0.02

(b) Transit game with non-deterministic encounters. Averaged over 20 samples of random distribution of interception probabilities.

Table 1: Runtime results in seconds for the transit game. EO — Evader's oracle time, PO — Patroller's oracle time, CLP — time to solve all subgames.

Simulation-based Evaluation

To validate the presented approach outside the very framework of game theory and to provide a bridge towards more applied work on maritime security, we have tested the game-theoretic route selection strategies on an agent-based simulation of maritime traffic (Vaněk et al. 2011). In contrast with the highly abstracted transit game, the simulation represents the domain with a much higher level of detail with near-continuous time and continuous space. We applied the transit game solution from the perspective of a vessel (corresponding to the Evader) that needs to repeatedly transit through the Gulf of Aden area. The pirate (corresponding to the Patroller) was not employing game-theoretic model; instead, in line with the domain knowledge, it was represented as an adaptive agent capable of learning from its past successes and failures in trying to intercept the transiting vessel. The learning capability was implemented using a softmax multi-armed bandit model (Sundaram 2005). We represented the transit area as a 12x4 grid graph with Patroller's base placed on the node closest to the Bosaso harbor, known as a major piracy hub. We computed the optimal Evader's strategy and deployed it in the simulation (Figure 5). We

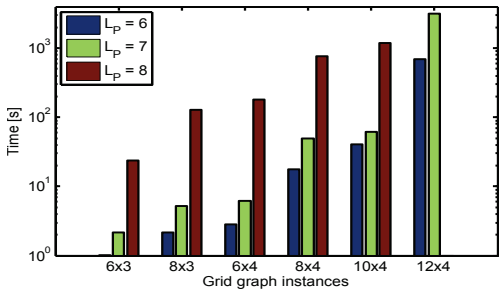


Figure 3: Computation time of *ESO-A* for a transit game with deterministic encounters on grid graphs of various size and varying maximum length of Patroller's walk L_P.

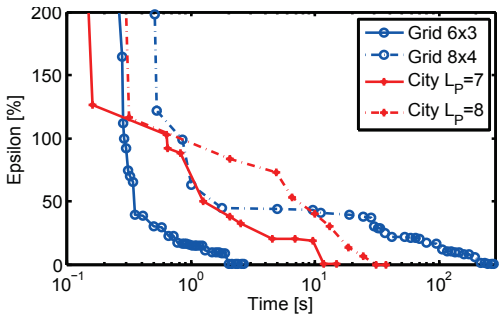


Figure 4: Trade-off between solution accuracy and convergence speed for various test problems for *ESO-A*.

then simulated 10000 transit runs and measured the pirate success ratio after its performance converged.

We compared the game-theoretic solution (denoted as *GT*) to two other transit strategies: the *IRTC* method representing the current transit scheme in which the transport vessels follow a fixed International Recommended Transport Corridor[6] and the *UNIFORM* method choosing randomly with uniform distribution from all possible shortest paths. Results in the Table 2 show that both the randomized strategies significantly outperform the current transiting scheme, moreover, the game-theoretic randomization out-

[6] http://www.cusnc.navy.mil

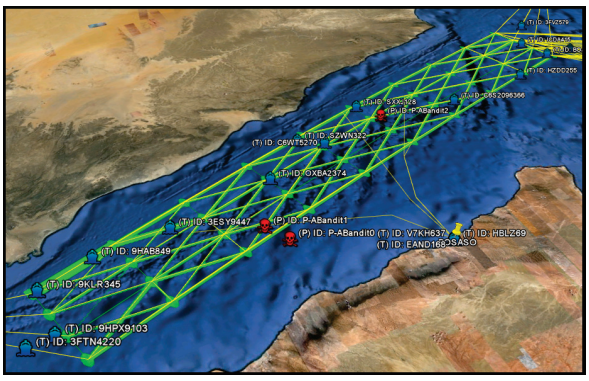


Figure 5: Gulf of Aden transit simulation with a deployed game-theoretic solution. Best viewed in color.

performs the uniform randomization. The relatively small difference between the two randomized strategies is given by the fact that the symmetric and homogeneous nature of the transit game leads to a solution which is close to uniform. Higher differences can be expected if more detail and/or structure is introduced (e.g. the real area in not strictly rectangular) however this is left for future work.

Strategy	IRTC	UNIFORM	GT
Rate [%]	22	9.19	8.53

Table 2: Comparison different strategies in simulation of transiting the Gulf of Aden.

Conclusions

We addressed the problem of a strategic confrontation between two mobile agents — an Evader trying to pass through an area where a Patroller is trying to intercept its transit. In order to find optimum route selection strategies, we have formalized the problem as a transit game, a novel security game model which assumes both players to be mobile.

The huge size of the strategy spaces called for the application of iterative oracle-based algorithms for NE computation. In order to limit the need for best response calculation, which is hard for games on graphs, we have introduced a novel accelerated variant of oracle algorithm which speeds up NE computation without the loss of the optimality guarantee. We have evaluated the approach on two classes of games inspired by real-world security applications and, in one case, were even able to compare the effectiveness of game-theoretic solution with currently deployed solutions.

Acknowledgments

This research was supported by the U.S. Office of Naval Research (grant no. N000140910537) and by the Czech Ministry of Education, Youth and Sports (grant no. LH11051).

References

Alpern, S. 1992. Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications* 163:286–288.

Barnhart, C.; Johnson, E.; Nemhauser, G.; and Savelsbergh, M. 1994. Branch and Price: Column Generation for Solving Huge Integer Programs. *Operations Research* 46:316–329.

Brooks, R. R.; Schwier, J.; and Griffin, C. 2009. Markovian Search Games in Heterogeneous Spaces. *IEEE Trans. on Systems, Man and Cybernetics Part B* 39(3):626–335.

Dickerson, J.; Simari, G.; Subrahmanian, V.; and Kraus, S. 2010. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In *Proceedings of AAMAS*, 299–306.

Flood, M. M. 1972. The Hide and Seek Game of Von Neumann. *Management Science*.

Gal, S. 1980. *Search Games*. Academic Press, New York.

Gilpin, R. 2009. Counting the Costs of Somali Piracy. Technical report, US Institute of Peace.

Halvorson, E.; Conitzer, V.; and Parr, R. 2009. Multi-step Multi-sensor Hider-seeker Games. In *Proceedings of IJCAI*.

Jain, M.; Kardes, E.; Kiekintveld, C.; Ordonez, F.; and Tambe, M. 2010a. Security Games with Arbitrary Schedules: A Branch and Price Approach. In *Proceedings of AAAI*.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordonez, F. 2010b. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces* 40:267–290.

Jain, M.; Vaněk, O.; Korzhyk, D.; Conitzer, V.; Tambe, M.; and Pěchouček, M. 2011. Double oracle algorithm for zero-sum security games on graphs. In *Proceedings of AAMAS*.

Jakob, M.; Vaněk, O.; and Pěchouček, M. 2011. Using agents to improve international maritime transport security. *IEEE Intelligent Systems* 26(1):90–96.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *Proceedings of ICML*, 536–543.

Ruckle, W.; Fennell, R.; Holmes, P. T.; and Fennemore, C. 1976. Ambushing Random Walks I: Finite Models. *Operations Research* 24:314–324.

Sundaram, R. K. 2005. *Studies in Choice and Welfare*. Springer Berlin Heidelberg.

Vaněk, O.; Bošanský, B.; Jakob, M.; and Pěchouček, M. 2010. Transiting Areas Patrolled by a Mobile Adversary. In *Proceedings of IEEE CIG*.

Vaněk, O.; Jakob, M.; Hrstka, O.; and Pěchouček, M. 2011. Using multi-agent simulation to improve the security of maritime transit. In *Proceedings of MABS*.

Washburn, A., and Wood, K. 1995. Two-person Zero-sum Games for Network Interdiction. *Operations Research* 43(2):243–251.

Appendix H

A Profit-Aware Negotiation Mechanism for On-Demand Transport Services

M. Egan and M. Jakob. A profit-aware negotiation mechanism for on-demand transport services. In *European Conference on Artificial Intelligence (ECAI)*, pages 273 – 278, 2014.

ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-273

A Profit-Aware Negotiation Mechanism for On-Demand Transport Services

Malcolm Egan and **Michal Jakob** [1]

Abstract. As new markets for transportion arise, on-demand transport services are set to grow as more passengers seek affordable personalized journeys. To reduce passenger prices and increase provider revenue, these journeys will often be shared with other passengers. As such, new negotiation mechanisms between passengers and the service provider are required to plan and price journeys. In this paper, we propose a novel profit-aware negotiation mechanism: a multi-agent approach that accounts for both passenger and service provider preferences. Our negotiation mechanism prices each passenger's journey, in addition to providing vehicle routing and scheduling. We prove a stability property of our negotiation mechanism using a connection to hedonic games. This connection yields new insights into the link between vehicle routing and passenger pricing. We also show via simulations the dependence of the service provider profit and passenger prices on the number of passengers as well as passenger demographics. In particular, our key observation is that increasing the number of passengers has the effect of increasing passenger diversity, which in turn increases the service provider's profit.

1 Introduction

On-demand transportation services are initiated at the request of passengers, between flexible origins and destinations. In current transport systems, on-demand transport plays an important role in the form of taxi services, and transportation for the elderly and disabled [6]. Looking to the near future, on-demand services are set to grow dramatically with advances in online markets, increased data collection and analysis in transport systems, and near-ubiquitous mobile communication services. Even now, numerous public (e.g. SUPERHUB [4]) and private (e.g. Lyft[2] and Cabforce[3]) organizations have begun R&D projects to implement open transport markets, supported by intelligent data aggregation.

The traditional formulation of the journey planning problem for on-demand transport is the dial-a-ride problem (DARP); a constrained version of the classical traveling salesman problem [9]. In the DARP, a fleet of vehicles services passengers with pick-up and drop-off time constraints. Importantly, the fleet of vehicles is operated by a single provider. The optimal solution of the DARP is then the minimum cost vehicle routes that satisfy all passenger constraints. An extensive collection of optimal and heuristic algorithms have been proposed within the operations research literature to solve the DARP, which are comprehensively summarized in [6].

[1] Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic email: {malcolm.egan,michal.jakob}@agents.fel.cvut.cz
[2] https://www.lyft.me/
[3] https://www.cabforce.com/home/

Despite the improved efficiency of the traditional DARP over unprincipled heuristics, it remains a centralized approach–well-known to scale poorly as the size of the transport system increases. To overcome the scaling problems in the traditional DARP, distributed approaches have been proposed. In particular, multi-agent techniques have been employed, where passengers and vehicles are treated as autonomous agents. For instance, multi-agent taxi scheduling was proposed in [11, 1, 14] and multi-agent DARP (and related vehicular routing problems) in [12, 7, 3, 2, 13].

Unfortunately, the state-of-the-art multi-agent DARP approaches in [12, 7, 3, 2, 13] have largely focused on finding the minimum cost routes for each vehicle (subject to passenger pick-up and delivery time constraints). In near-future transport markets, the optimal vehicle routes will be determined by what passengers are prepared to pay–hidden from the service provider–in addition to the cost of vehicle routes. These price preferences were partially addressed in [10] via a cost sharing mechanism; however, the intimate connection between passenger pricing and journey planning was not considered.

In this paper, we propose a novel profit-aware negotiation mechanism for the DARP, to obtain vehicle routes, as well as passenger allocations and prices. In particular, we develop a four-stage negotiation between each passenger and the service provider; passenger preferences, vehicle capacities, and route costs are all accounted for. Our negotiation mechanism fundamentally differs from previous approaches to the multi-agent DARP as our focus is on the service provider *profit*–explicitly accounting for the individual preferences of both the provider and passengers–instead of costs that lump preferences together. As such, our negotiation mechanism should in fact be viewed as a market mechanism–a protocol to exchange services for monetary payment. Our approach opens the way for the outcomes of our negotiation to reflect the behaviors and motivations of service providers and passengers in the real world.

In order to efficiently route and price each passenger's journey, we cluster passengers into feasible trips. Each vehicle's route is then found by routing through a subset of the passenger clusters. To ensure that the clusters are stable–i.e., no bias against any passenger–we introduce the passenger cluster game and show that it is in fact a type of hedonic game. We then prove that the stability of passenger clusters (analogous to coalitions) is determined by the cost of travelling between clusters and the price each passenger is offered. Our result has the surprising implication that minimizing the cost for the initial clustering is not necessarily optimal when cluster stability is also required.

We then show via Monte Carlo simulations that the service provider profits and passenger prices are highly dependent on the number of potential passengers and the passenger demographic. In particular, increasing the number of potential passengers increases

diversity, which in turn increases the profit of the service provider. We also show that our negotiation mechanism has the desirable property that the service provider charges more when passengers are prepared to pay more.

2 System Model

Consider an on-demand transportation network consisting of a single service provider and N passengers. The service provider owns a fleet of K vehicles, each with a capacity of C passengers. Passenger pick-up and drop-off locations, and direct routes between locations are represented by the graph $G = (V, E)$. In particular, the pick-up and drop-off locations are represented by vertices in the set V, while the direct routes between locations in V are represented by the edges in the set E. We assume that all passengers to be serviced are known before the negotiation begins[4] and that each vehicle starts and finishes at the same depot.

Associated to each vertex $v \in V$ (corresponding to pick-up/drop-off locations) is a service time s_v, which represents the time required for the vehicle to board passengers. Moreover, associated to each edge $e \in E$ (corresponding to direct routes between locations) are:

1. a start location $u \in V$;
2. an end location $w \in V$;
3. a cost $c_e \in [0, \infty)$ to the service provider to traverse edge $e \in E$;
4. and an edge traversal time $t_e \in \{0, 1, 2, \ldots\}$.

The edge cost c_e and the edge traversal time t_e are found during pre-processing where the service provider solves the shortest path problem between u and w on the underlying road network.

2.1 Passenger Preferences

Before the passenger allocation is performed via our negotiation mechanism, each passenger provides the following information to the service provider:

1. pick-up and drop-off locations, denoted by $v_{i,p} \in V$, $v_{i,d} \in V$ for passenger i's pick-up and locations, respectively;
2. pick-up time interval, denoted by $(a_i, b_i) \in \{0, 1, 2 \ldots\} \times \{0, 1, 2, \ldots\}$ with $a_i \leq b_i$ for the i-th passenger;
3. latest drop-off time, $l_i \in \{0, 1, 2, \ldots\}$ with $l_i > b_i$ for the i-th passenger.

In addition to the travel requirements, each passenger also has preferences for the amount she is willing to pay. In particular, we assume that passenger i is prepared to pay a maximum price of $p_{i,\max} = r_{i,\max} R_i$, where R_i is the distance as the crow flies between passenger i's pick-up and drop-off locations[5] and $r_{i,\max} \in (0, \infty)$ is a price rate in €/km, which converts the distance traveled into euros.

To account for differences in price preferences between passengers, we model $r_{i,\max}$ as a random variable (independently and identically distributed for each passenger) distributed according to the generalized Beta distribution on support $[0, r_{\max}]$. In particular, the cumulative distribution function (CDF) for the price rate $r_{T,i}$ is

$$F_{r_{i,\max}}(x) = \frac{1}{B(\alpha, \beta)} \int_0^{\frac{x}{r_{\max}}} t^{\alpha-1}(1-t)^{\beta-1} dt \qquad (1)$$

for all $i \in \{1, 2, \ldots, N\}$, where $B(\alpha, \beta)$ is the Beta function. We have chosen the generalized beta distribution as it has a large number of distributions with bounded support as special cases, which means that our model can be tailored to a variety of demographics.

Importantly, the actual maximum price, $p_{i,\max}$ that passenger i is prepared to pay is known *only* to passenger i. The service provider only knows the *distribution* of $p_{i,\max}$, for each passenger. This has important consequences in the negotiation as the service provider cannot initially be sure whether or not a passenger will accept an offer.

2.2 Service Provider Preferences

The objective of the service provider is to maximize its *profit*; i.e the difference between the total revenue obtained from all the passengers it services, and the total cost of all vehicle journeys. This is fundamentally different from the traditional DARP, where the service provider minimizes the total cost, which does not reflect passenger price preferences. We note that the profit maximization problem is always subject to the route feasibility constraints, as passengers will not accept the service if pick-up or drop-off time constraints are not satisfied.

There are two notions of profit maximization in this paper. The first notion is that of the maximum expected revenue, which is applicable when the service provider does not have any side information about each passenger's maximum price. On the other hand, the second notion is that of the maximum minimum (maximin) revenue; that is, the vehicle routes and passenger pricing that maximizes the minimum possible profit under the given routing and pricing policy. We use the maximin approach when the service provider does have side information about each passenger's maximum price. As detailed in Section 3, the maximum expected revenue is employed to generate an initial offer to each passenger. Later in the negotiation, the service provider uses the maximin revenue in the final offer to ensure that passengers are guaranteed to accept.

3 Proposed Negotiation Mechanism

In this section, we propose our negotiation mechanism for the on-demand transport network detailed in Section 2. Our negotiation mechanism proceeds in four stages, which ends with a feasible journey plan for each vehicle and prices for each passenger. The stages are summarized as follows:

1: The service provider offers each passenger a journey along with a price.
2: Each passenger makes an initial decision whether to reject the offer or to conditionally accept.
3: The service provider makes final offers to the passengers that conditionally accepted.
4: Each remaining passenger makes its final decision whether to reject or unconditionally accept the offer.

3.1 Stage 1: Initial Service Provider Offers

In the first stage of the negotiation, the service provider constructs possible routes for each vehicle and prices the journey for each passenger; summarized in Algorithm 1. It is important to note that the passenger journeys are coupled, even though all passengers negotiate independently of each other. This is due to vehicle capacity and passenger pick-up/drop-off time constraints, which mean that the optimal vehicle allocation for a given passenger depends on which other

[4] This scenario is known as the static DARP and is known to be realistic for several types of on-demand services [6].

[5] Although the distance travelled may be significantly further than R_i, it is often difficult for passengers to estimate the actual distance to be travelled. As such, the distance as the crow flies is a reasonable estimate, on which passengers can make price-related decisions.

passengers are also allocated to the vehicle. As the service provider does not know a priori which passengers will ultimately accept their offer, it must first enumerate multiple feasible journey plans for each vehicle.

Algorithm 1 Summary of Stage 1

1. **Group passengers into feasible clusters** subject to constraints on vehicle capacity, passenger pick-up interval, and passenger drop-off time. This is illustrated in Fig. 1.

2. **Enumerate journeys via the cluster tree** (illustrated in Fig. 2). Each branch is a feasible journey for a single vehicle. Vehicles are allocated a single branch, each with distinct passengers.

3. **Price each journey for each passenger**. This is achieved by solving the optimization problem in (7) to maximize the expected profit for the service provider.

In practice, both the number of vehicles in the service provider's fleet and the number of potential passengers can be very large. As such, it is not computationally feasible to enumerate all possible routes for each vehicle through each subset of passengers. To overcome this problem, we instead use a principled heuristic approach based on passenger clustering.

3.1.1 Passenger Clustering and Journey Enumeration

Our journey enumeration algorithm first clusters passengers together into (minimum cost) clusters and then enumerates feasible vehicle routes between different clusters. As noted in [8, 9], minimum cost clustering is a type of set partition problem[6], which partitions the passengers into routes, with the k-th vehicle route (chosen from the set of all routes Ω) denoted y_k and passengers in route y_k (with edges $E_k \subset E$) given by $\{i|\delta_{ki} = 1\}$. The minimum cost partition is then the solution to

$$\begin{aligned} \underset{\delta, y_k}{\text{minimize}} \quad & \sum_{k\in\Omega} \sum_{a\in E_k} c_a y_k \\ \text{subject to} \quad & \sum_{k\in\Omega} \delta_{ki} y_k = 1, \quad i \in \{1, 2, \ldots, N\} \\ & y_k \in \{0, 1\}, \qquad\qquad k \in \Omega, \end{aligned} \tag{2}$$

where a trip through cluster k (with passengers $\{i|\delta_{ki} = 1\}$) is feasible with respect to vehicle capacity and passenger pick-up/drop-off constraints.

For large on-demand transport networks, it is not practical to solve (2) optimally. Instead, we adopt a heuristic clustering approach based on [8], where the clustering is based on the locations of the passengers. The output of the passenger clustering is a set of clusters $C_1, C_2, \ldots$, each with an initial pick-up interval (corresponding to the first passenger in the cluster) and final drop-off time (corresponding to the last passenger in the cluster). The result of the passenger clustering algorithm is illustrated in Fig. 1.

Once the clustering has been performed, feasible journeys between clusters are enumerated. This is achieved by forming a tree (illustrated in Fig. 2), with each branch corresponding to a feasible route between clusters. As each vehicle begins and ends at the same depot, it is only necessary to enumerate the tree for a single vehicle.

[6] We note that the set partitioning problem in (2) cannot be solved using standard set partitioning approaches for coalition formation due to the constraints from vehicle capacities and passenger pick-up/drop-off times.

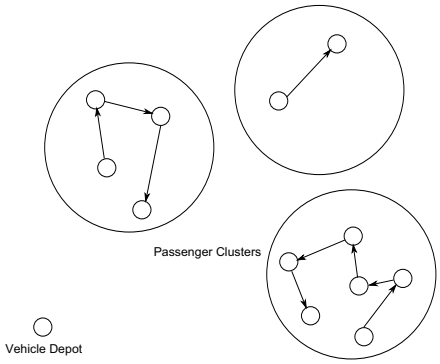


Figure 1: Illustration of passenger clustering. The small circles represent passenger pick-up and drop-off locations. If a given passenger is a member of a cluster, then the cluster contains both the pick-up and drop-off locations.

This is because the multiple vehicle routes are obtained by allocating vehicles branches with distinct passengers.

At the end of the journey enumeration step in Stage 1, the service provider obtains the potential routes for each vehicle in its fleet, with each journey enumeration corresponding to a different subset of the potential users. In addition, the cost (to the service provider) of each route is also given, which is used to compute passenger prices.

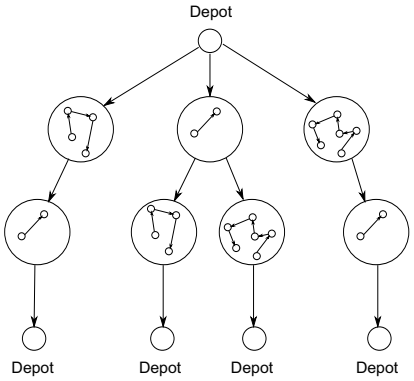


Figure 2: Illustration of the journey enumeration tree.

3.1.2 Journey Pricing

The next step in Stage 1 in the negotiation is to allocate prices to each passenger. As the service provider only knows the distribution of the passengers' maximum price, it maximizes its expected profit.

To formulate the optimization problem, we first define the profit conditioned on the acceptance of a set $S \subset \{1, 2, \ldots, N\}$ (and all other passengers rejecting the offer), denoted by $P_S(r)$ at price rate r. There are two scenarios to consider: S consists of passengers that can be served simultaneously by the K vehicles; and S consists of passengers that cannot be served simultaneously. In the first scenario, P_S is given by

$$P_{S,1}(r) = \sum_{k\in S} r R_k - c_S, \tag{3}$$

where c_S is the cost to the provider of servicing passengers S and R_k is the distance as the crow flies between passenger k's pick-up and drop-off locations, as detailed in Section 2.1.

On the other hand, if the passengers in S cannot be served simultaneously then the service provider must choose the subset $S_c \subset S$ that can be served and maximizes the operators profit over S. As the price rate is not known yet (it is the solution to the optimization problem in (7)), we obtain S_c by solving

$$S_c = \arg\max_{S_c \subset S} \sum_{k \in S_c} R_k - c_{S_c}, \tag{4}$$

where c_{S_c} is the cost to the provider to service the passengers in S_c. The conditional profit in the second scenario is then given by

$$P_{S,2}(r) = \sum_{k \in S_c} rR_k - c_{S_c}. \tag{5}$$

Finally, the conditional profit for the service provider when the set S of passengers accept their offers is given by

$$P_S(r) = \begin{cases} P_{S,1}(r), & \text{if the service provider can simultaneously} \\ & \text{serve all passengers in } S; \\ P_{S,2}(r), & \text{otherwise.} \end{cases} \tag{6}$$

The total profit P_{T1} is then obtained using the law of total probability.

We find the passenger price rate, r, via the following optimization problem.

$$\max_r \mathbb{E}[P_{T1}] = \max_r \sum_{S \in \mathcal{P}} P_S(r) F_{r_T}(r)^{|S|} \left(1 - F_{r_T}(r)\right)^{N-|S|}, \tag{7}$$

where $\mathcal{P}$ is the power set of $\{1, 2, \ldots, N\}$ and F_{r_T} is the Beta distribution CDF in (1). In general, the optimization problem (7) is non-convex. As such, the problem is numerically solved to find local maxima.

At the end of the first stage, each passenger k is offered their desired journey at a price p_k based on the price rate obtained in (7). In particular, the offer price for passenger k is $p_k = rR_k$ for $k \in \{1, 2, \ldots, N\}$[7].

3.2 Stage 2: Initial Passenger Decisions

In the second stage of the negotiation, each passenger i makes a preliminary decision based on the price p_i that it has been quoted by the service provider. The passenger can respond to the service provider's quote in one of two ways: *conditionally accept*; or *reject*.

If passenger i conditionally accepts, it means that she has formed a contract with the service provider, which ensures that the service must be paid for unless the service provider increases the price. On the other hand, if passenger i rejects the offer then she no longer is interested in a journey with the service provider.

3.3 Stage 3: Final Service Provider Offers

In the third stage of the negotiation, the service provider has additional information. In particular, the service provider knows: what the users that conditionally accept are prepared to pay; and which users have rejected the offer.

As not all passengers will usually accept their offer, the service provider must update the passenger clusters. Although the cluster sizes will change if not all passengers accept, we assume that the service provider does not change the passenger clusters. The consequences of this assumption are examined further in Section 4. We emphasize that the journey enumeration from Stage 1 is still feasible, even with changes in the cluster sizes.

To obtain final prices for each passenger, the service provider solves the maximin profit problem over the passengers that have accepted in the previous stage. Let S^* be the passengers that accepted their offers in the previous stage and P_{T2} the final profit obtained from our negotiation. The maximin profit problem is then

$$\max_r \min P_{T2}. \tag{8}$$

As we have a lower bound on the maximum price each user is prepared to pay, the maximin profit problem in (8) is equivalent to

$$S_c^* = \arg\max_{S_c \subset S^*} \sum_{k \in S_c} rR_k - c_{S_c}, \tag{9}$$

where the passengers in S_c^* are charged $rR_k,\ k \in S_c^*$ and the other passengers are priced out.

The service provider uses the pricing in Stage 3 to ensure that the passengers in desirable clusters accept, which in turn maximizes the service provider's profit. That is, the service provider will raise the price of passengers that have conditionally accepted so that they reject the final offer[8]. This means that Stage 3 is equivalent in concept to the maximum determination problem in auctions (see e.g. [15]).

3.4 Stage 4: Final Passenger Decisions

In the fourth stage of the negotiation, the remaining passengers make their final decision based on the latest offer from the service provider. Each passenger either *unconditionally accepts* the final offer, or *rejects* it; i.e, passengers that conditionally accepted in Stage 2 now either accept the offer if the maximum price did not rise above p_i or reject otherwise.

4 Passenger Cluster Stability

So far, we have focused on how the service provider can perform pricing and journey planning in order to maximize its profits. We now take the perspective of the passengers. Although the passengers are not directly part of the service provider's clustering algorithm in Stage 1 (see Section 3.1.1), it is highly desirable from the perspective of fairness–an additional means for passengers to discriminate between providers–that the clustering is not biased against any given passenger; that is, the passenger cannot find a cluster where both the passenger and the new cluster are better off. In order to avoid this bias after Stage 2, the fact that some passengers are likely to have rejected their offer must be accounted for in the original clustering and passenger pricing.

In this section, we prove a new relationship between the stability of the passenger clusters (defined precisely in Definition 2) and the price each passenger is charged after the initial passenger decisions in Stage 2. Our main result is a sufficient condition for the stability of the passenger clusters, which guarantees that each passenger cannot unilaterally find a different cluster that will improve her chance of being serviced, while also improving the chance that the cluster she seeks to join will be serviced.

[7] Although we consider a common price rate for each passenger (largely for the purposes of exposition), it is possible to extend to different price rates for each passenger and even different price rates for each possible route (for the same passenger).

[8] We note that it is possible that passengers may accept service, even after the price is raised. In this case the service provider can subcontract the journey with no financial loss.

To begin, we define the passenger cluster game, which naturally arises from the notion of passenger clusters and is closely related to a coalitional game.

Definition 1 *The passenger cluster game is the set of players $N = \{1, 2, \ldots, n\}$ and preference relation over the clusters $\succeq$ (analogous to the payoff function), which for passenger i*

$$\{i\} \cup C_j \succeq_i \{i\} \cup C_k, \tag{10}$$

if

$$\sum_{m \in \{i\} \cup C_j} p_m - c_{C_j \cup \{i\}} \geq \sum_{m \in \{i\} \cup C_k} p_m - c_{C_k \cup \{i\}}, \tag{11}$$

where p_m is the price offered to passenger m and c_{C_k} is the total cost to the service provider of servicing cluster k (excluding the cost of traveling to the cluster from another cluster or the depot).

Intuitively, a passenger i prefers cluster C_j over C_k if the profit generated by $\{i\} \cup C_j$ for the service provider is greater than the profit generated by $\{i\} \cup C_k$. As neither the passengers nor the service provider ultimately know the passengers that will accept, this means that the service provider is more likely to route through cluster $\{i\} \cup C_j$ than $\{i\} \cup C_k$ and hence passenger i is more likely to be serviced.

Importantly, the passenger cluster game is a hedonic game[9]. A practical notion of stability in hedonic games is individual stability, which we state in terms of the passenger cluster game.

Definition 2 *Suppose that the passengers have formed clusters $\Pi = \{C_1, C_2 \ldots\}$. Then, Π is individually stable if for every passenger i and for all $C_k \in \Pi$, $C_i \succeq_i C_k \cup \{i\}$ or $C_k \succeq_j C_k \cup \{i\}$ for all $j \in C_k$.*

Intuitively, the clusters are individually stable if no passenger can unilaterally find a new cluster that both the passenger and the cluster it seeks to join prefer over the original clusters.

As we show in Theorem 1, the stability of passenger clusters is intimately linked to the price each passenger is offered for its journey.

Theorem 1 *All passengers are individually stable in their allocated clusters if $p_i < c_{r,k}\ \forall k \neq i$, where $c_{r,k}$ is the minimum cost path between cluster C_k and cluster C_i (containing passenger i).*

Proof of Theorem 1: Observe that if passenger i is to pay p_i, then the profit of the new cluster, $P_{C_k \cup \{i\}}$, is bounded by

$$P_{C_k \cup \{i\}} \leq \sum_{j \in C_k} p_j + p_i - c_{C_k} - c_{r,k} \tag{12}$$

since the cost of servicing passenger i is at least the cost of travelling from a passenger in cluster k to the pick-up location of passenger i. As such, when $p_i < c_{r,k}$ it follows that

$$P_{C_k \cup \{i\}} < P_{C_k} \tag{13}$$

and the result follows. □

The main consequence of Theorem 1 is that the price the service provider charges must be bounded by the cost of traveling between clusters, to ensure cluster stability. Surprisingly, this means that initial clustering based on minimum cost is not necessarily optimal when stability is required, even if the set partitioning problem in (2) is solved optimally. In fact, both the cost of each cluster and the distance between clusters must be taken into account[10].

[9] A hedonic game is a coalitional game where the preference relation for each player over coalitions depends only on the members of the coalitions and nothing else.

[10] We leave the design of clusters to optimally tradeoff between cost and distance for future work.

5 Simulation Results

In this section, we perform a Monte Carlo simulation study of the influence of the number of passengers as well as passenger demographics on pricing and service provider profits. To the best of our knowledge, this is the first study on the effect of the on-demand transport network on service provider *profits* (as opposed to costs) and actual passenger prices.

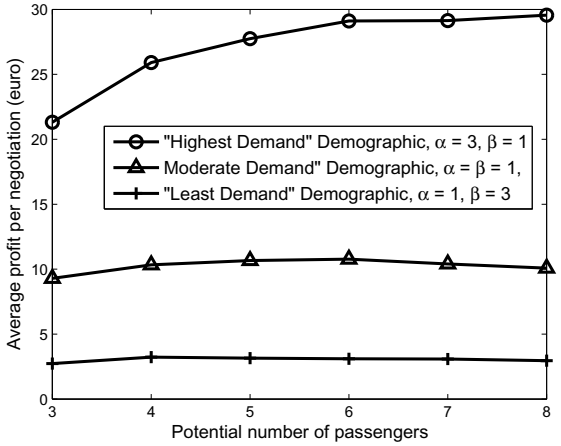


Figure 3: Plot of the average profit per negotiation for a varying number of potential passengers. Three passenger demographics are considered: "highest demand" ($\alpha = 3$, $\beta = 1$); "moderate demand" ($\alpha = \beta = 1$); and "least demand" ($\alpha = 1$, $\beta = 3$).

Consider the on-demand transport network consisting of a service provider with $K = 3$ unit capacity vehicles and N potential passengers. This is a realistic fleet size when (as we consider) the passenger pick-up and drop-off locations are placed randomly according to the uniform distribution on $[0, L] \times [0, L]$, where $L = 4$ km, which along with the direct routes between locations forms the graph G (see Section 2). We also expect that the insights we obtain approximately hold for larger scale networks with a similar vehicle density (≈ 5 vehicles/km^2). We also assume that:

1. the start of the pick-up interval is uniformly distributed on $\{1, 2, \ldots, 60\}$;
2. the duration of the pick-up interval is uniformly distributed on $\{5, 6, \ldots, 20\}$;
3. the average vehicle velocity is $v = 20$ km/hour;
4. the maximum journey time is uniformly distributed on $\{\lfloor 3R_i/v \rceil, \lfloor 6R_i/v \rceil, \ldots, \lfloor 7R_i/v \rceil\}$;
5. and the standard cost of vehicle journeys is $c_s = €0.3$/km.

As detailed in Section 2, the maximum price rate for passenger i is distributed according to the generalized Beta distribution on $[0, 10]$ with parameters α, β. To model different passenger demographics, we vary α and β, which in turn changes the shape of the corresponding distribution function. In particular, we define a passenger demographic in terms of demand for services; for example, passengers at peak hour typically have higher demand, and as such are prepared to pay more. More precisely, we consider the following demand demographics:

1. $\alpha = 3$ and $\beta = 1$ corresponds to the Beta distribution with the largest proportion of the probability mass above €5 (half of the

maximum possible price rate), which means that the demographic has a high proportion of "high-demand" passengers.

2. $\alpha = \beta = 1$ corresponds to the Beta distribution with half the probability mass above €5, which means that the demographic has a moderate proportion of "high-demand" passengers.
3. $\alpha = 1$ and $\beta = 3$ corresponds to the Beta distribution with the largest proportion of the probability mass below €5, which means that the demographic has a low proportion of "high-demand" passengers.

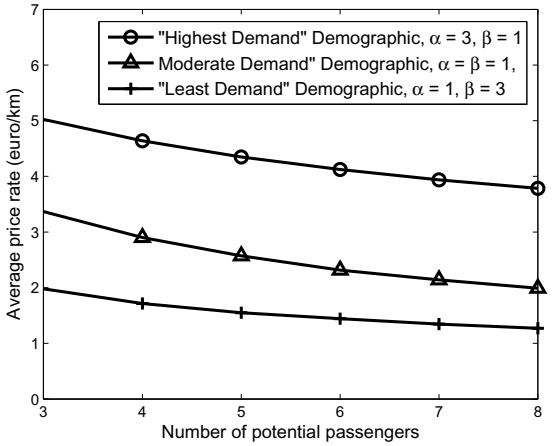


Figure 4: Plot of the average price rate for a varying number of potential passengers. Three passenger demographics are considered: "highest demand" ($\alpha = 3$, $\beta = 1$); "moderate demand" ($\alpha = \beta = 1$); and "least demand" ($\alpha = 1$, $\beta = 3$).

Fig. 3 shows the relationship between the average profit the service provider makes from a single negotiation as the number of potential passengers increases. We observe that for all passenger demographics (corresponding to different α, β) the average profit increases and the number of potential passengers increases. This is due to the fact that increasing the pool of potential passengers also increases the diversity, until the profit saturates. As such, it is possible to find cheaper routes with passengers that pay more. We also observe that the demographic with the lowest proportion of the high demand passengers ($\alpha = 1$, $\beta = 3$) also yields the lowest average profit. As the proportion of wealthier passengers increases further ($\alpha = \beta = 1$ and $\alpha = 3$, $\beta = 1$), the average profit also increases, for all sizes of the potential passenger pool.

Fig. 4 shows the relationship between the average price rate paid by each passenger as the number of potential passengers increases. Observe that the average price rate reduces for a larger number of passengers. This can be explained by noting that it is not always possible to find low cost passengers with only a small number to choose from, which means that the price must be higher for the service to be profitable. Also observe that as the proportion of the passengers that have high demand increases, so does the average price rate. As such, when the passengers are prepared to pay more for a journey, the service provider will charge more.

6 Conclusions and Future Work

On-demand services are set to play an important role in transport markets. In light of this, we have developed a negotiation mechanism between passengers and the service provider to obtain vehicle routes, as well as passenger allocation and pricing. In contrast with previous work, we focus on the profit of the service provider instead of the cost, which allows us to account for both service provider and passenger preferences. Future extensions will account for larger scale networks (using the testbed in [5]), dynamic passenger arrivals and more complex passenger decision making processes.

ACKNOWLEDGEMENTS

Supported by the European social fund within the framework of realizing the project Support of inter-sectoral mobility and quality enhancement of research teams at Czech Technical University in Prague, CZ.1.07/2.3.00/30.0034 and by the Ministry of Education, Youth and Sports of Czech Republic grant no. LD12044.

REFERENCES

[1] R. Bai, J. Li, J.A.D. Atkin, and G. Kendell, 'A novel approach to independent taxi scheduling problem based on stable matching', *Journal of the Operational Research Society*, (2013).

[2] D. Barbucha, 'A multi-agent approach to the dynamic vehicle routing problem with time windows'. Springer Berlin Heidelberg, (September 2013).

[3] C. Bertelle, M. Nabaa, D Olivier, and P. Tranouez, 'A Decentralised Approach for the Transportation On Demand Problem', in *From System Complexity to Emergent Properties*, eds., M.A. Aziz-Alaoui and C. Bertelle, Understanding Complex Systems, 281–289, Springer Berlin Heidelberg, (2009).

[4] I. Carreras, S. Gabrielli, D. Miorandi, A. Tamilin, F. Cartolano, M. Jakob, and S. Marzorati, 'SUPERHUB: a user-centric perspective on sustainable urban mobility', in *Proc. Sense Transport '12*. ACM, (2012).

[5] M. Certicky, M. Jakob, R. Pibil, and Z. Moler, 'Agent-based simulation testbed for on-demand transport services', in *Proc. of the 13th International Conference on Autonomous Agents and Multiagent Systems*, (2014).

[6] J.-F. Cordeau, 'The dial-a-ride problem: models and algorithms', *Ann Oper Res*, **153**, 29–46, (2007).

[7] C. Cubillos, F. Guidi-Polanco, and C. Demartini, 'MADARP: multi-agent architecture for passenger transportation systems', in *Proc. of the 8th International IEEE Conference on Intelligent Transportation Systems*, (September 2005).

[8] F.H. Cullen, J.J. Jarvis, and H.D. Ratliff, 'Set partitioning based heuristics for interactive routing', *Networks*, **11**, 125–143, (1981).

[9] Y. Dumas, J. Desrosiers, and F. Soumis, 'The pickup and delivery problem with time windows', *European Journal of Operational Research*, **54**, 7–22, (1991).

[10] M Furuhata, K. Daniel, S. Koenig, F. Ordonez, M. Dessouky, M.-E. Bruent, L. Cohen, and X. Wang, 'Online cost-sharing mechanism design for demand-responsive transport systems', in *Proc. of the Thirteenth International Joint Conference on Autonomous Agents and Multiagent Systems*, (2014).

[11] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev, 'Multi-agent real time scheduling system for taxi companies', in *Proc. of 8th Internation Conference on Autonomous Agents and Multiagent Systems*, (2009).

[12] J. Koźlak, 'Multi-agent approach to dynamic pick-up and delivery problem with uncertain knowledge about future transport demands', *Fundamenta Informaticae*, **71**, 27–36, (2006).

[13] M. Mes, M. van der Heijden, and A. van Harten, 'Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems', *European Journal of Operational Research*, **181**(1), 59–75, (August 2007).

[14] K.T. Seow and D.H. Lee, 'Performance of multiagent taxi dispath on extended-runtime taxi availability: a simulation study', *IEEE Transactions on intelligent Transportation Systems*, **11**(1), 231–236, (March 2010).

[15] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, 2009.

Appendix I

Market Mechanism Design for Profitable On-Demand Transport Services

M. Egan and M. Jakob. Market Mechanism Design for Profitable On-Demand Transport Services. Submitted to *Transportation Research Part B: Methodological*. 2014.

Market Mechanism Design for Profitable On-Demand Transport Services

Malcolm Egan[1] and Michal Jakob[1]

[1] *Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic.*

Abstract

On-demand services in the form of dial-a-ride and taxi services are crucial features of all major cities. However, not all on-demand services are equal. In particular, not-for-profit dial-a-ride service with coordinated drivers drastically differ from profit-motivated taxi services with uncoordinated drivers. As such, there are two key threads of research for efficient scheduling, routing, and pricing for passengers: dial-a-ride services (first thread); and taxi services (second thread). Unfortunately, algorithms for joint optimization of scheduling, routing, and pricing have not been developed in either thread; largely due to the widespread assumption of fixed pricing. In this paper, we introduce the third thread: profit-motivated on-demand services with coordinated drivers. To maximize provider profits and efficiency of the service, we propose a new market mechanism for third thread on-demand services, where passengers negotiate with the service provider. In sharp contrast to previous work, our mechanism jointly optimizes scheduling, routing, and pricing. Ultimately, we demonstrate that our approach leads to higher profits, compared with standard fixed price approaches, while maintaining comparable efficiency.

Keywords: on-demand transport, market mechanism, pricing

I. Introduction

Call a taxi in any major city and it will often arrive within minutes. Despite the success and widespread use of taxis and other on-demand services, there is room for improvement: higher profits; reduced prices; and even lower waiting times, targeted at passengers that have the highest demand for the service. Fortunately, the ubiquity of mobile internet and secure online financial transactions offers has opened the way for highly efficient on-demand transport; able to ensure that the right driver transports the right passenger at the right time.

At present, efficiently operating a fleet of on-demand vehicles remains difficult. The cause is the coupling between three key sub-problems: which passengers should be serviced by each vehicle *(routing)*; at what time should each passenger be picked up *(scheduling)*; and how much each passenger should be charged *(pricing)*. In efficient on-demand transport, the sub-problems should be solved jointly; not be decoupled.

Unfortunately, there are few computational techniques to solve the three on-demand sub-problems jointly; despite over three decades of research on related vehicle routing problems. While this might seem surprising, there is good reason: there are in fact two distinct research threads—each addressing a different niche.

The first thread is dial-a-ride services—targeted at the niche of elderly and disabled transport. Vehicles in fleets offering this type of service are coordinated by a single provider. Each vehicle collects passengers within requested pick-up time intervals and drops each passenger off before a requested drop-off time. Dial-a-ride services are typically heavily subsidized by governments, due to the important role they play for vulnerable members of the community; for instance, by taking an elderly woman to the hospital for a check-up. These subsidies have an important repercussion: dial-a-ride service providers are often not-for-profit organizations. As such, the aim of providers is to minimize costs; as opposed to maximizing profits. This means that the pricing sub-problem is not considered, and the most popular formalization known as the dial-a-ride problem focuses on the routing and scheduling sub-problems. Both centralized (see [1] for an extensive survey) and decentralized agent-based [2], [3] approaches have been proposed.

The second thread is taxi and private hire services. Taxi services contrast with dial-a-ride

services in two key ways: they are profit-motivated; and vehicles are not heavily coordinated by a single service provider. This is because the drivers are self-interested and unable to easily determine the current locations and destinations of other nearby taxis. While taxi services are profit-motivated, the most common pricing strategy is to use a fixed price-rate; i.e., the price scales as a—usually linear—function of the distance (see [4] for pricing with a non-linear function of distance). As such, the price-rate does not factor in the number and demand[1] of passengers that have ordered a ride. Moreover, the fixed price-rate means that the pricing sub-problem is decoupled from routing and scheduling; instead, the focus is on reducing waiting times and travel distance (and hence reducing costs) [5]–[9]. We note that the approach in [10] does optimize pricing for single taxi operation via dynamic programming; however, scheduling and routing for a fleet of taxis is not considered.

In essence, algorithms for the joint solution of the three on-demand sub-problems have not been developed in either of these well-established threads of on-demand transport research. As such, in this paper, we introduce a new niche—made practical with recent technological developments—where joint solution is feasible: the third thread.

A. The Third Thread

The aim of the third thread of on-demand services is profitability and efficiency when drivers are coordinated (as in dial-a-ride services) and providers are profit-motivated (as in taxi services). It is worth noting that this niche only recently arose as more drivers and passengers have adopted internet-enabled smartphones.

Looking from the computational perspective, standard algorithms for dial-a-ride or taxi services cannot be directly applied. This is due to the fact that the first two threads decouple pricing from routing and scheduling. As such, to solve the three sub-problems jointly, new computational techniques are required.

[1]The notion of demand here can be easily misconstrued. We mean demand relative to factors such as cost of living and time of day. We do not mean in terms of need; in particular, we believe that pricing should not be manipulated in emergency situations.

In this paper, we propose a new market-based approach for the third thread of on-demand services. In fact, we are able to jointly solve the three sub-problems: routing; scheduling; and pricing. There are two fundamental aspects of our approach: a new passenger model; and a new market mechanism. The passenger model goes beyond the standard approach in the dial-a-ride problem by providing a realistic probabilistic model for each passenger's expectations—ultimately allowing the provider to tailor journey offers to passenger that order the service. Our new approach significantly extends on our initial work in [11] by enriching the passenger models, and improving the joint scheduling, routing and pricing algorithm to allow for deviations from requested journeys.

B. Modeling Passenger Expectations

In traditional approaches to the dial-a-ride problem and taxi routing and scheduling, a passenger is simply a request; that is, pick-up and drop-off times and locations. By viewing a passenger in this way it is not possible to optimize the price for each passenger. This is due to the fact that the service provider must also account for passenger preferences; in other words, how likely a passenger is to accept a journey.

To enrich the passenger model, we account for the probability that a passenger will accept an offer, on top of her request. Two key factors are considered: the price of the offer; and the journey deviation. The journey deviation corresponds to the difference between the time that passengers request pick-up and drop-off and the actual times. This passenger model forms the basis for price optimization in our new mechanism for scheduling, routing, and pricing.

C. A Market Mechanism for the Third Thread

At its heart, the third thread of on-demand services is multiple, independent, passengers ordering transportation from a service provider with multiple vehicles. A natural way forward is to use a market, as markets, by definition, exchange goods or services for money.

In this paper, we propose a new market mechanism for third thread on-demand services—profit-motivated service providers with coordinated drivers. Our market mechanism is designed

to jointly route and schedule vehicles, and price passengers. In particular, we introduce a four-stage mechanism, initiated by the service provider generating an offer for passengers, and ending with passengers making a final decision of whether to accept or reject.

It is worth pointing out the key difference between our market mechanism and other negotiation mechanisms used for on-demand services. That is, our mechanism ultimately schedules, routes, and *prices* passengers. This is not the case in other approaches. The reason is that the other negotiation mechanisms are used for, in a decentralized fashion, scheduling and routing vehicles. There is no mention of how to price passengers. On the other hand, we are able to price passengers (in addition to routing and scheduling vehicles) due to our enriched passenger model, which captures passenger expectations.

D. Key Contributions

In this paper, we introduce third thread on-demand services—profit-motivated providers and coordinated drivers—and a market mechanism to route, schedule, and price passengers. We summarize our key contributions as follows:

1) **Agent-based passenger modeling:** We introduce new models for passengers that are enriched to include expectations for price and deviations from requests, on top of the standard request model.
2) **Market mechanism:** We propose a new market-mechanism to jointly schedule, route, and price passengers. Offers are generated for passengers via an expected profit maximization algorithm. We also analyze the effect of varying the time between when the mechanism is run, which leads to closed-form expressions.
3) **Business case:** Third thread on-demand services are a new niche and as such there is a genuine need for a business case. To this end, we perform a simulation study to evaluate the potential for profits, and also efficient service of passengers. We demonstrate that incorporating passenger expectations in our new passenger models does in fact improve profitability over standard fixed price-rate approaches, while maintaining comparable efficiency.

II. Modeling Agents

Consider the network consisting of a single on-demand service provider and N passengers. The service provider owns a fleet of K *unit capacity* vehicles that all start and finish their journeys' at a common depot; each vehicle traveling with average velocity ν. Each passenger has requested pick-up and drop-off locations, which are represented by elements from the set of vertices V in a directed graph G. The directed graph G represents the underlying road network. As such, the set of edges E in G represent direct routes between locations in V.

Associated to each edge $e \in E$ (corresponding to direct routes between locations) are:

1) a start location $u \in V$;
2) an end location $w \in V$;
3) a cost $c_e \in [0, \infty)$ to the service provider to traverse edge $e \in E$;
4) and an edge traversal time $\tau_e \in \mathbb{Z}_+$.

The edge cost c_e and edge traversal time τ_e are found during pre-processing where the service provider solves the shortest path problem between u and v on the underlying road network. Note that when edge e connects the vertices u and w, the traversal time is denoted by $\tau_{u,w}$. This model is appropriate when drivers are salaried or pay a fixed commission, which can be easily incorporated into the edge cost c_e.

So far, we have simply described the basic service provider and passenger model used to model dial-a-ride services. In order to enrich the model for third thread on-demand services, we need to introduce two new features to the model:

1) passengers capable of making a decision whether or not to accept a journey offer;
2) and a service provider capable of evaluating the probability a passenger will accept an offer.

A. Passengers

The first step in modeling passengers is descriptive. In particular, we identify two stylized facts, which we believe hold for passengers using on-demand services. The concept of stylized facts has been widely used in computational macroeconomics as a means of validating descrip-

tive models of real markets [12]. Based on the stylized facts, we develop a passenger policy, which is used to model how passenger decide whether or not to accept a journey.

Before introducing the stylized facts and passenger policy, we define the parameters that determine passenger behavior. First, a request from passenger i consists of:

1) a pick-up location $v_{i,p} \in V$;
2) a drop-off location $v_{i,d} \in V$;
3) a pick-up time interval $(a_i, b_i) \in \{0, 1, 2 \ldots\} \times \{0, 1, 2, \ldots\}$, with $a_i \leq b_i$;
4) and a latest drop-off time $l_i \in \{0, 1, 2, \ldots\}$, with $l_i > b_i$.

In response to the requests, the service provider offers a journey to each passenger, which consists of two components: the deviation of the journey from the request; and the price of the journey. More precisely, the deviations are defined as follows.

Definition 1. *Let T_i be the actual pick-up time and L_i be the actual drop-off time, for passenger i. The pick-up interval deviation, denoted by $\gamma_{p,i}$, is defined as*

$$\gamma_{p,i} = \begin{cases} a_i - T_i, & \text{if } T_i < a_i \\ T_i - b_i, & \text{if } T_i > b_i \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Similarly, the drop-off time deviation, denoted by $\gamma_{d,i}$, is defined as

$$\gamma_{d,i} = \begin{cases} L_i - l_i, & \text{if } L_i > l_i \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The deviation is then given by $\delta_i = \gamma_{p,i} + \gamma_{d,i}$.

The next step towards the passenger policy is to introduce two stylized facts for passenger behavior. Stylized facts—widely used in computational economics [13]—are important as they provide a means of justifying the passenger policy as a plausible description of real passenger decision-making relevant to on-demand transport.

Our first stylized fact is as follows.

Stylized Fact 1. *The maximum price a passenger will pay for a journey and maximum deviation δ do not vary significantly for passengers that regularly use on-demand services.*

We can justify this stylized fact by observing that passengers change transportation habits with great difficulty [14]. As such, we expect that regular on-demand users will have a well-defined maximum price they are prepared to pay.

Next, we have the second stylized fact.

Stylized Fact 2. *The probability that a passenger will accept an offer decreases when either the price increases with the deviation fixed, or the deviation increases with the price fixed.*

More colloquially, this stylized fact is a formal statement of the intuitive notion that if there is a better deal, more passengers will accept.

Based on the first stylized fact, we propose the following descriptive model for passenger policies for whether or not to accept an offer:

1) If $r < r_{\max}$ and $\delta < \delta_{\max}$, then the passenger will accept;
2) Otherwise, then the passenger will reject.

Observe that this policy ensures that the first stylized fact holds. Moreover, our descriptive model does not violate the second stylized fact, as the policy is concerned with individual passenger decisions and the stylized fact is concerned with the aggregate.

B. Service Provider

The aim of the service provider is to schedule, route and price passengers to maximize its expected profit. Previously, the price-rate was assumed to be fixed, and only the scheduling and routing were optimized. This means that there is no uncertainty and the expected profit can be maximized by minimizing the cost. The key new feature that we introduce for the service provider is a probabilistic model of each passenger. As we show in Section III, this extra information improves the expected profit over standard approaches.

We now detail the passenger model that the service provider uses. Importantly, we also show that it satisfies both the stylized facts, which means that it is a plausible probabilistic descriptive model of passengers.

In order for the service provider to infer demand at a given price, it requires the probability each passenger will accept her offer. It is necessary to consider the *probability* an offer is accepted

as the service provider does not perfectly know the maximum deviations and price that any given passenger will accept. In particular, the probability passenger i accepts her offer is given by

$$\Pr(i \text{ accept}) = \Pr(\delta_i \leq \delta_{i,\max}, r_i \leq r_{i,\max}). \tag{3}$$

Remark 1. *We emphasize that the realizations of the maximum deviation $\delta_{i,\max}$ and the maximum price rate $r_{i,\max}$ are not known to the service provider, only to passenger i.*

To obtain the probability that any given passenger accepts, we assume that the service provider knows the joint probability density function $f(\delta_{i,\max}, r_{i,\max})$. Our assumption that the service provider has statistical knowledge of $(\delta_{i,\max}, r_{i,\max})$ ensures that each passenger is not always charged at the maximum possible price she is prepared to pay–realistic in competitive profit-motivated on-demand services. On the other hand, statistical knowledge is enough to enable the service provider to optimize the expected profit, as we detail in Section III.

The density function $f(r_{i,\max}, \delta_{i,\max})$ will typically depend on factors such as time of day, or the location of the service region. We focus on scenarios where the maximum deviations and price-rate are independent, which occur when the factors determining the maximum deviations and the financial factors affecting the price-rate are unrelated. An example is the taxi spot market, where only a small deviation is acceptable. In these scenarios, the density function is separable; i.e.,

$$f(r_{i,\max}, \delta_{i,\max}) = f_r(r_{i,\max}) f_\delta(\delta_{i,\max}) \tag{4}$$

We model $f_r(r_{i,\max})$, $f_\delta(\delta_{i,\max})$ via the scaled Beta distribution with parameters (α_r, β_r), $(\alpha_\delta, \beta_\delta)$ respectively. The reason for this is that the Beta distribution is a flexible distribution, which generalizes a wide variety of distributions with bounded support. The density functions are given by

$$\begin{aligned} f_r(r_{i,\max}) &= \frac{1}{r_u B(\alpha_r, \beta_r)} \left(\frac{r_{i,\max}}{r_u}\right)^{\alpha_r - 1} \left(1 - \frac{r_{i,\max}}{r_u}\right)^{\beta_r - 1}, \\ f_\delta(\delta_{i,\max}) &= \frac{1}{\delta_u B(\alpha_\delta, \beta_\delta)} \left(\frac{\delta_{i,\max}}{\delta_u}\right)^{\alpha_\delta - 1} \left(1 - \frac{\delta_{i,\max}}{\delta_u}\right)^{\beta_\delta - 1}, \end{aligned} \tag{5}$$

where $B(\alpha, \beta)$ is the Beta function, and the densities have support $[0, r_u]$ for $r_{i,\max}$ and $[0, \delta_u]$ for $\delta_{i,\max}$.

It is easy to see that both stylized facts are features of the passenger model that the service provider uses. In particular, as $r_{i,\max}$ and $\delta_{i,\max}$ increase, the probability that a passenger will accept is reduced. This means that both the descriptive passenger models we have detailed will exhibit the features described by the stylistic facts.

C. On Prediction

So far, we have argued that our descriptive passenger models satisfy the stylized facts exhibited by real passengers. However, at this point these stylized facts have only been shown to hold under current conditions; that is, the present fixed price-rate approach. The remainder of this paper is concerned with a new routing, scheduling, and pricing approach. Clearly this is a structural change and as such, it is necessary to justify that the stylized facts still hold. This is due to the fact that we are predicting the performance of a socio-technical system under structural changes; known to be notoriously hard to do in many economic settings [13].

Fortunately, the stylized facts (Stylized Facts 1 and 2) are independent of how the service provider performs scheduling, routing, and pricing. This suggests that the stylized facts should hold even when the service provider changes the underlying algorithms and indeed the market. Despite this, it is possible that the parameters of the distributions for the maximum price-rate and deviation may vary when the pricing algorithm is changed. These variations can potentially be overcome by updating the parameters $\alpha_r, \beta_r, \alpha_\delta, \beta_\delta$ to appropriately model the passengers' preferences.

The invariance of the stylized facts to service provider scheduling, routing, and pricing, suggests that key features arising from analysis of our model will be consistent with real-world practice.

III. Our Proposed Market Mechanism

In this section, we propose a new market-based approach for scheduling, routing, and pricing in third thread on-demand services. We first detail the desiderata that our design should fulfill.

We then overview the proposed market mechanism, and detail step-by-step the interactions between the service provider and each passenger.

A. Design Desiderata

The design of personalized on-demand transport services is constrained by the physical (i.e., the vehicle fleet) and financial resources of the service provider, and the expectations of passengers. Ultimately, these constraints determine whether or not the provider is financially viable.

To ensure that the resource constraints of the provider and the expectations of passengers are satisfied, the design of our market mechanism is guided by three key desiderata:

1) The service provider should be profitable.
2) The passengers that desire the service the most should obtain it.
3) There should be a simple interface between passengers and the service provider.

Our first two desiderata are to ensure that the service provider is profitable and that vehicles are allocated to the passengers that most value the service, while the third desiderata brings our design in line with current trends that simplify access to transportation services.

We point out that our second desiderata is closely related to the standard notion of efficiency in mechanism design [15]; that is, an efficient mechanism allocates service to the passengers that are prepared to pay the most for it. In particular, we can formalize the value of a journey to a given passenger as follows.

Definition 2. *Denote the the maximum price for passenger i as $p_{i,\max}$. If the set of serviced passengers is S, then the efficiency of our market mechanism is then defined as*

$$\mathcal{E} = \sum_{i \in S} p_{i,\max}. \tag{6}$$

Our third desiderata is to bring our mechanism in line with current trends towards simplifying access to transpotation. The key impact of this desiderata on our approach is that passengers are not required to price their own journey; instead, the service provider always generates the first offer. This means that passengers do not need to be aware of the behavior

of other passengers or how the service provider allocates vehicles. The only decision each passenger needs to make is whether the journey offer is acceptable or not, which in our model is determined by the deviation and price of the offered journey.

B. Overview

The three key design desiderata motivate a market mechanism, where passengers are not required to price their own journeys and the service provider generates offers that maximize its profits. In our approach, we propose the following mechanism structure:

1) The service provider makes each passenger an initial offer. The offer to each passenger is based on vehicle allocations and passenger pricing done by the service provider to maximize its average profit.
2) Each passenger responds to the initial offer by either rejecting or conditionally accepting the offer. Conditional acceptance is a contract between the passenger and the service provider, which requires the passenger to pay the amount offered unless either the price is raised or the deviation from the requested journey increases.
3) The service provider computes final vehicle journey plans and passenger pricing.
4) The passenger either rejects the offer (due to the service provider breaking the contract) or unconditionally accepts, where the passenger is required to pay the provider for the service.

A key aspect of the mechanism is that the service provider generates initial offers to each passenger via optimization of the average profit. There are two sets of variables associated to the optimization problem. First is the sets of passengers allocated to the same vehicle, which leads to the allocation $C = \{C_1, \ldots, C_K\}$ with C_i corresponding to the passengers allocated to the i-th vehicle. Second is the price-rate, r that each passenger will pay. Formally, the optimization problem is

$$\begin{aligned} &\underset{C_1,\ldots,C_K,r}{\text{maximize}} \quad \sum_{S\subset N}\left(\sum_{i\in S} rR_i - c_i\right)\prod_{i\in S}\Pr(i \text{ accept})\prod_{j\in S^c}(1-\Pr(j \text{ accept})) \\ &\text{subject to} \quad 0 \le r \le r_u, \end{aligned} \tag{7}$$

where $\Pr(i \text{ accept})$ is given by (3), r is the price-rate offered to each passenger, and c_i is the cost of servicing passenger i.

It is important to note that this optimization problem generalizes the standard formulation for dial-a-ride services. In particular, we optimize over the pricing (encoded in r) and the scheduling and routing (encoded in the sets $C_1, \ldots, C_K$). In contrast, minimum cost scheduling and routing for dial-a-ride services does not consider pricing; i.e., the price-rate r is assumed to be fixed. As such, there is potential for higher expected profit than simply using a minimum cost approach.

We also point out that irrespective of how the scheduling and routing is performed, using our approach in (7) will always lead to a higher expected profit. This is particularly important in the case that it is not tractable to optimally solve the clustering problem and heuristics are required.

While optimally scheduling, routing, and pricing passengers will yield a higher expected profit, it is not straightforward to solve. In particular, the problem in (7) is difficult for two reasons:

1) The objective is generally nonlinear and also non-convex.
2) The number of sets $C_1, \ldots, C_K$ to be searched is even greater than in the standard approaches as there are no feasibility constraints that constrain the passengers able to serviced by a given vehicle. This occurs because we allow deviations from passenger requests.

To alleviate the two difficulties in solving (7), we adopt the "cluster-then-price" strategy; that is, we solve the problem (7) in two stages. In the first stage we cluster passengers into groups that are all served by the same vehicle. We note that the cluster formation is not a straightforward extension of standard approaches (e.g., [16]) due to the profit-based objective. We detail our clustering algorithm in Section III-C. We then show how the price-rate offered to each passenger is obtained by pricing the passengers based on the clustering. The remainder of section details the interactions between the service provider and passengers in our mechanism– including how final vehicle allocations and prices are computed.

Our proposed market mechanism is summarized in Fig. 1, where dependencies are illustrated by arrows between each stage.

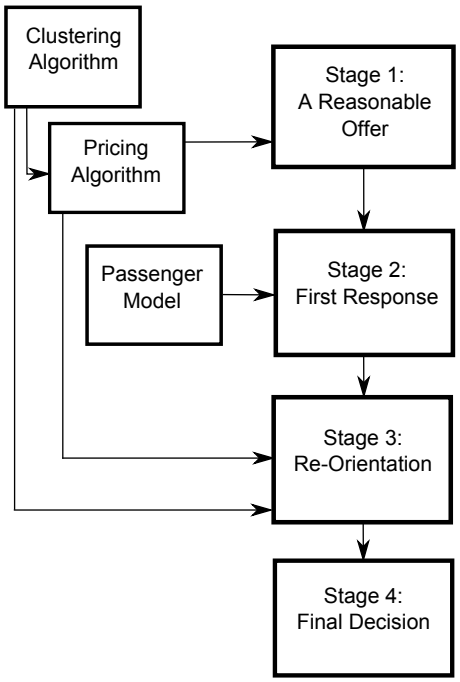


Fig. 1: Overview of the proposed market mechanism for pricing and vehicle allocation in on-demand transport networks.

C. Stage 1: A Reasonable Offer

In the first stage of the mechanism, the service provider generates an offer for each passenger i. The offer consists of:

1) maximum deviation, γ_i;
2) and the price, $p_i = rR_i$, where R_i is the distance between pick-up and drop-off locations for passenger i.

To allow deviations from passenger requests during clustering, we introduce a probabilistic feasibility constraint, which is closely related to the probability that a passenger will accept the journey. The constraint is defined as follows.

Definition 3. *A journey is ϵ-feasible for passenger i, if*

$$\Pr(\delta_{\max} \geq \delta_i) \geq 1 - \epsilon, \tag{8}$$

where $\delta_{\max}$ is a random variable representing the unknown maximum deviation for passenger i. Moreover, a cluster is said to be ϵ-feasible if each passenger in the cluster is ϵ-feasible, with the set of ϵ-feasible clusters denoted by $\mathcal{F}_\epsilon$.

Intuitively, the notion of ϵ-feasible allocations generalizes the standard notion of feasible clustering. That is, standard hard constraint clustering is 0-feasible; i.e., each passenger must have a feasible journey with probability one. Importantly, ϵ-feasible allocations are a key feature in our approach, as by relaxing the feasible constraints it is possible for the service provider to offer journeys with deviations from requests to passengers. Ultimately, this can allow the provider to service more passengers with a single vehicle, which (as we show in Section V) leads to higher profits and efficiency of the market mechanism.

Passenger clustering: We now develop a clustering algorithm that minimizes the cost of clusters subject to the probabilistic feasibility constraint detailed in Definition 3. In order to form passenger clusters, we determine whether or not the addition of a passenger to a cluster is ϵ-feasible. This is achieved by computing the times that the inserted passenger will picked up and dropped off. To reduce complexity of the insertion algorithm, a passenger can be inserted only if the new passenger does not affect the deviations of the passengers already in the cluster, which means that previously clustered passenger pick-up and drop-off times are not changed after the passenger has been allocated to a cluster.

To illustrate, consider a passenger 0 to be inserted into cluster $C_j = \{k_1, \ldots, k_{|C_j|}\}$, where passengers in C_j are in the order that they are serviced. There are three situations to consider: insert passenger 0 before passenger k_1; insert passenger 0 between passengers k_l and k_{l+1}, with $1 \leq l < |C_j|$; and insert passenger 0 after passenger $k_{|C_j|}$.

To determine whether a passenger can potentially be inserted into the cluster, there are three checks that need to be performed:

C1: Can the passenger be inserted without changing the deviation of previously clustered passengers?

C2: If C1 holds, what are the pick-up and drop-off times with the minimum deviation (i.e., δ_i)?

C3: Is the cost of traveling to the passenger's pick-up location and from the drop-off location less than any other previously checked cluster?

The checks are performed using the insertion algorithm detailed in Algorithm 2. In particular, the algorithm attempts to insert passengers such that the new passenger does not change the deviations of previously clustered passengers—corresponding to **C1**. In the case that it is possible to insert the passenger without affecting other clustered passengers, the insertion algorithm determines whether the passenger is ϵ-feasible and computes the pick-up and drop-off times to minimize the deviation (implementing **C2**). Finally, the cost is computed and compared with previously checked clusters, and the lowest cost, c_{best}, insertion is updated (based on **C3**).

The minimum deviation pick-up and drop-off times (for **C2**) are computed using simple inequality tests due to the fact that the objective is linear; i.e., it is $\delta = \gamma_p + \gamma_d$ (from Section II). In particular, there are three types of insertion tests: before the first passenger in the cluster; between passengers j and $j+1$; and after the last passenger. To illustrate, consider the potential insertion of passenger 0 between passengers j and $j+1$. First, the new passenger's journey must fit. This means that $L_j + \tau_{j,0} + \tau_{0,0} + \tau_{0,j+1} < T_{j+1}$. Next, we choose the minimum deviation insertion. This is done by checking whether $T_{j+1} - \tau_{0,j+1} \leq l_0$, which means that the deviation can only be caused by the pick-up time. The pick-up time is then chosen to minimize the deviation. The case where $T_{j+1} - \tau_{0,j+1} > l_0$ can be treated similarly.

The cost is computed as

$$c_{i,ins} = c_{l-1,i} + c_{i,l}, \tag{9}$$

where passenger i is inserted between passenger $l-1$ (if $l = 0$, then this corresponds to the depot), and passenger l (this may also be the depot).

With the insertion algorithm in hand, it is now possible to describe our cluster formation algorithm. The cluster formation algorithm searches through the current clusters for each passenger i to determine whether the passenger can be inserted. After checking each potential cluster, passenger i is inserted into the lowest cost and feasible cluster, and the next passenger

procedure INSERTION(c_{best},i,C_j)

Set passenger to be inserted as i.

Compute cost $c_{i,ins}$ for cluster $\{i\}$ using (9).

if $c_{i,ins} < c_{best}$ **then**

$c_{best} = c_{i,ins}$; $l_{best} = l$; $j_{best} = j$.

end if

Set cluster $C_j = \{k_1, \ldots, k_M\}$, where $M = |C_j|$.

for $1 \leq l \leq |C_j| + 1$ **do**

Perform check **C1**.

Compute pick-up and drop-off times t_p^* and t_d^* (see discussion).

if $\Pr(\delta_{\max} \geq t_p^*, \gamma_{\max} \geq t_d^*) \geq 1 - \epsilon$ **then**

Compute cost $c_{i,ins}$ using (9).

if $c_{i,ins} < c_{best}$ **then**

$c_{best} = c_{i,ins}$; $l_{best} = l$; $j_{best} = j$.

end if

end if

end for

return j_{best}, l_{best}, c_{best}.

end procedure

Fig. 2: Insertion algorithm.

is considered. Our cluster formation algorithm is detailed in Algorithm 3.

Passenger pricing: With the cluster formation algorithm in hand, we now introduce the joint scheduling, routing, and pricing algorithm used by the service provider to generate expected

procedure CLUSTERFORMATION

Randomly choose a unique index in $\mathcal{N} = \{1, 2, \ldots, N\}$ for each passenger.

Initialize cluster $C_1 = \{1\}$;

Initialize allocation $\pi = \{C_1\}$ and partition index $\mathcal{I} = \{1\}$.

Initialize the set of unclustered passengers $\mathcal{U} = \{2, \ldots, N\}$.

while $\mathcal{U} \neq \emptyset$ **do**

Set $\mathcal{J} \leftarrow \mathcal{I}$

while $\mathcal{J} \neq \emptyset$ **do**

Randomly choose an element $j \in J$.

Update j_{best}, l_{best}, c_{best} using Insertion(c_{best}),i, C_j) (see Algorithm 2).

$\mathcal{J} \leftarrow \mathcal{J} \setminus j$

end while

Update $C_{j_{best}}$ by inserting i before l_{best}; $\mathcal{U} \leftarrow \mathcal{U} \setminus i$; $\mathcal{J} \leftarrow \emptyset$.

end while

return Cluster allocation π

end procedure

Fig. 3: Cluster formation algorithm with probabilistic feasibility constraints.

profit maximizing offers. The algorithm approximately solves the following problem:

$$\begin{aligned} \underset{r,\epsilon}{\text{maximize}} \quad & \mathbb{E}[P(r)] \\ \text{subject to} \quad & 0 \leq r \leq r_u \\ & 0 \leq \epsilon \leq 1 \end{aligned} \tag{10}$$

,

where $\mathbb{E}[P(r)]$ is the expected profit at price rate r, which is given by

$$\mathbb{E}[P(r)] = \sum_{S \subset N} \left(\sum_{i \in S} p_i - c_i \right) \prod_{i \in S} \Pr(i \text{ accept}) \prod_{j \in S^c} \left(1 - \Pr(j \text{ accept})\right), \tag{11}$$

with $\Pr(i \text{ accept})$ given by (3). Importantly, all price rates greater than r_u are rejected by the passengers with probability one (this follows from (5)). The price offered to each passenger i is

procedure EXPECTEDPROFITMAXIMIZATION

Initialize ϵ and set $P_{opt} \leftarrow 0$.

while $\epsilon > 0$ **do**

Set $\epsilon \leftarrow \epsilon - \epsilon_{step}$.

Solve (10) to obtain the optimal expected profit P^*, with ϵ fixed.

if $P^* > P_{opt}$ **then**

Set $P_{opt} \leftarrow P^*$; $r_{opt} \leftarrow r$.

Set $C_{opt} \leftarrow \{C_1, C_2, \ldots\}$ (corresponding to clusters from Algorithm 3).

end if

end while

return $P_{opt}, r_{opt}, C_{opt}$.

end procedure

Fig. 4: Joint pricing and clustering algorithm to maximize the expected profit for Stage 1 of our market mechanism.

then given by $p_i = rR_i$, where R_i is the distance from passenger i's pick-up to drop-off.

Our solution clusters passengers based on the probabilistic feasibility constraint with parameter $\epsilon_1 \approx 1$ and then optimizing the price using a standard scalar nonlinear optimization algorithm (i.e., a descent algorithm). This is repeated for parameter $\epsilon_{k+1} = \epsilon_k - \epsilon_{step}$ until $\epsilon_{k+1} < 0$, with $0 < \epsilon_{step} \leq 1$. The pricing and clustering solution that maximizes the expected profit over probabilistic feasibility parameters $\epsilon_1, \epsilon_2, \ldots$ is then chosen. As such, at the end of Stage 1, the price rate r and passenger clusters $C_1, C_2, \ldots$ are obtained by the service provider. The procedure is summarized in Algorithm 4.

The final output of the expected profit maximization algorithm is an offer consisting of a journey deviation and price for each passenger. This offer is then communicated to the passenger, and the service provider waits for the passengers response, given in the next stage.

D. Stage 2: First Response

In the second stage of the negotiation, each passenger makes a preliminary decision to accept or reject the *conditional journey* offered by the service provider. If passenger i accepts, it means that she has accepted the journey offer as long as the service provider does not change the offer. On the other hand, if passenger i rejects the offer then she is no longer interested in a journey with the service provider.

We emphasize that if the passenger accepts, then she has agreed to a contract with the service provider; that is, she must pay for the service unless the service provider either raises the price or increases the journey deviation. We note that this type of contract is standard for other transportation services, such as pre-booked trains or buses.

E. Stage 3: Re-Orientation

In the third stage of the negotiation, the service provider has additional information. In particular, the service provider knows both what the users that conditionally accept are prepared to pay, and which users have rejected the offer.

As not all passengers will usually accept their offer, the service provider must update the passenger clusters. Although the cluster sizes will change if not all passengers accept, no passengers are allocated to different clusters. This ensures that the maximum journey deviation for each passenger does not change.

To obtain final prices for each passenger, the service provider solves the maximin profit problem over the passengers that have accepted in the previous stage. Let S^* be the passengers that accepted their offers in the previous stage and Q the final profit obtained from the market mechanism. The maximin profit problem is then

$$\max_{\{r_i\}_{i \in S^*}} \min Q. \tag{12}$$

We note that in this stage of the mechanism, different passengers can be offered journeys at different price rates.

As the previous stages of the mechanism have revealed a lower bound on the maximum price each user is prepared to pay, the maximin profit problem in (12) is equivalent to finding

the set of passengers

$$S_c^* = \arg\max_{S_c \subset S^*} \sum_{k \in S_c} rR_k - c_{S_c}, \tag{13}$$

and then pricing passengers such that the passengers in S_c^* are charged $rR_k,\ k \in S_c^*$, while the other passengers are charged at a higher price rate. This allows these passengers to find another service or for the provider to subcontract the journeys, which avoids losses.

The service provider uses the pricing strategy in Stage 3 to ensure that the passengers in desirable clusters accept, which in turn maximizes the service provider's profit. That is, the service provider will raise the price of users that have conditionally accepted so that they do not require the provider to be exposed to large losses. Although undesirable from the perspective of service provider reputation, we believe that this strategy is likely to be necessary in real-world practice. This is due to the fact that service providers have both physical (i.e, fleet size) and financial (i.e., initial capital) constraints. As such, it is not possible to service all passengers that might accept without either investing in a larger fleet size or hiring additional vehicles. To cope with these additional costs, it is necessary for passengers to be charged more when they are difficult to serve.

F. Stage 4: Final Decision

In the fourth stage of the negotiation, the remaining passengers make their final decision based on the latest offer from the service provider. Each passenger either *unconditionally accepts* the final offer, or *rejects* it; i.e, passengers that conditionally accepted in Stage 2 now either accept the offer or reject otherwise. We emphasize that a passenger can reject the offer unless the provider has increased the price; otherwise, the passenger must pay for the journey.

We point out that the service provider cannot always service all passengers that accept; either because there are not enough vehicles, or the passenger would cause a net loss for the provider. As such, it is highly desirable from the perspective of financial solvency of the provider to be able to raise the price and allow the passenger to find alternative transport. If the passenger still accepts the journey even after the price has been raised, then the service provider can use the additional revenue to hire an additional vehicle to service the passenger.

At the end of Stage 4 of the mechanism, all passengers to be serviced are known to the service provider, are priced, and have been allocated to vehicles. Moreover, each vehicle has a journey plan. The performance of our market mechanism is evaluated in Section V. In the next section, we analyze the role of the time interval between mechanism runs, which determines the mechanism rate.

IV. Mechanism Parameter Design

A key assumption in our market mechanism is that the passengers are known to the provider before the beginning of the mechanism. For dial-a-ride services this is known as the static scenario, and is often problematic as passengers can make requests after one run of the mechanism and before the next. This was solved by allowing dynamic arrivals, where passengers can be inserted while vehicles are on the road. However, the dynamic approach cannot be directly used with our mechanism without statistics for the locations and the prices dynamic passengers would be prepared to pay. While such an approach is in principle possible using historical passenger prices, and pick-up and drop-off location data, an unprecedented level of data refinement would be required.

To resolve this issue, we instead adapt the rate our market mechanism is run. Importantly, the mechanism rate is in fact a fundamental feature of any on-demand market mechanism. There are two key parameters that determine the mechanism rate: the probability that a passenger request is ignored; and the probability that a passenger cannot be serviced before the next mechanism run.

In this section, we derive simple analytical expressions for the probability a request is ignored, P_{ignore}, and the probability a passenger cannot be serviced in time, $P_{overtime}$. The key purpose of the analytical expressions are to guide design of the mechanism rate. In particular, we demonstrate the tradeoff between P_{ignore} and $P_{overtime}$, as the time between mechanism runs is increased.

A. Analysis

Our analysis of P_{ignore} and $P_{overtime}$ is based on a simplified probabilistic model of passengers and vehicles. Although the simplifications lead to a coarse approximation of real-world on-demand networks, conclusions from our analysis are supported by intuitive explanations. The key assumptions are as follows:

1) The time between mechanism runs is T minutes and the corresponding rate is $R = 1/T$.
2) Each vehicle services only a single passenger per interval between mechanism runs. This is reasonable for short intervals T.
3) The time for a vehicle to travel a distance z is given by νz, where ν is the average velocity.
4) The time each request arrives forms a homogeneous Poisson process with rate λ. Moreover, the time between request delivery and the desired pick-up time, Δ, is exponentially distributed with mean $1/\lambda$.
5) Pick-up and drop-off locations are distributed according to a Poisson point process with intensity ζ. Moreover, the drop-off location is the closest point to the pick-up locations, which means that the distribution of the distance is[2]

$$f_Z(z) = e^{-\zeta \pi z^2} 2\pi\zeta z. \tag{14}$$

We now turn to analysis of P_{ignore} and $P_{overtime}$. Our analysis is based on two new analytical expressions for the probabilities. First, the probability a request is ignored, P_{ignore}, is given in the following proposition.

Proposition 1. *The probability a request is ignored is given by*

$$P_{ignore} = 1 - \frac{1}{\lambda T}\left(1 - e^{-\lambda T}\right), \tag{15}$$

where a is the desired pick-up time, and Δ is the time between a passenger's request and desired pick-up time.

Proof. See Appendix A. □

[2]This result follows immediately by considering the probability that there is no point within radius R from the origin, which is given by $e^{-\pi R^2}$.

Observe that as $T \to \infty$, $P_{ignore} \to 1$, which means that as the interval between mechanism runs increases, the probability a request is ignored tend to one–an intuitive result. On the other hand, as $T \to 0$, $P_{ignore} \to 0$.

The probability that a passenger cannot be serviced before the next mechanism run, $P_{overtime}$, is given in the following proposition.

Proposition 2. *The probability a passenger cannot be serviced before the next mechanism run is given by*

$$P_{overtime} = \left(1 - \frac{2\pi\zeta T}{\nu}\right) e^{-\pi\zeta T^2/\nu^2} + \frac{\nu}{T\sqrt{\zeta}} \left(\frac{1}{2} - Q\left(\frac{T\sqrt{2\pi\zeta}}{\nu}\right)\right) \tag{16}$$

where a is the earliest pick-up time, z is the distance between pick-up and drop-off, and $Q(\cdot)$ is the Q-function, a standard special function defined in (22).

Proof. See Appendix B. □

Observe that as $T \to \infty$, $P_{overtime} \to 0$, and as $T \to 0$, $P_{overtime} \to 1$. This is an intuitive result as it simply states that when the interval is large the probability that a passenger cannot be serviced in time approaches zero. Importantly, this observation means that there is a tradeoff between P_{ignore} and $P_{overtime}$. In other words, it is not possible to avoid passengers being ignored and not being serviced, with probability one using the same mechanism rate.

B. Tradeoff

It is not immediately obvious from Propositions 1 and 2, which factors are key in determining the tradeoff between P_{ignore} and $P_{overtime}$. As such, we now examine the behavior of P_{ignore} and $P_{overtime}$ numerically.

Fig. 5 plots the tradeoff between P_{ignore} and $P_{overtime}$ for varying λ and ζ, based on our analysis in Section IV-A. Observe that increasing λ, also leads to an increase in P_{ignore}. On the other hand, an increase in ζ leads to increase in $P_{overtime}$. As such, the intersection between P_{ignore} and $P_{overtime}$ (the crossover rate) reduces when λ or ζ are increased.

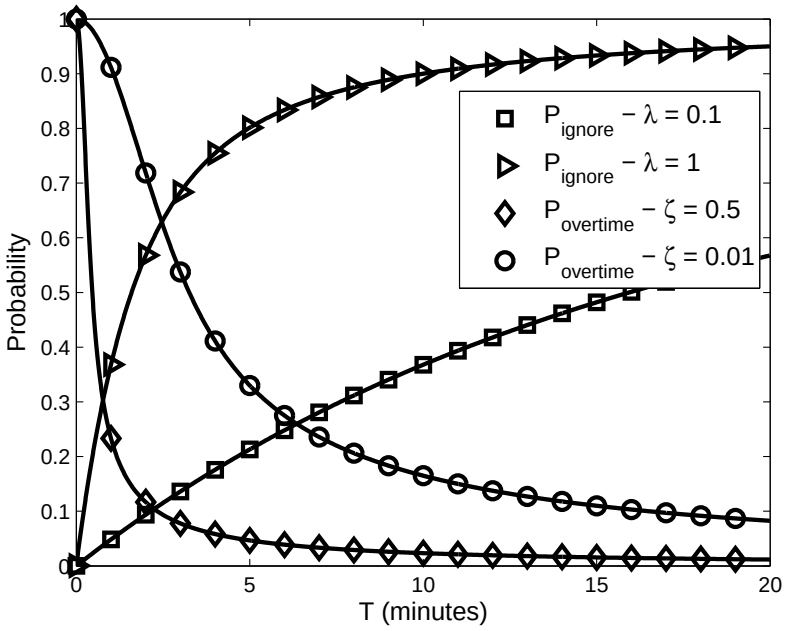


Fig. 5: *Plot of the tradeoff between P_{ignore} (from Proposition 1) and $P_{overtime}$ (from Proposition 2), for varying time between mechanism runs T. We assume vehicles travel at* 20 *km/hr.*

V. The Business Case: Simulation Results

With our mechanism for third thread on-demand transport services in hand, we are ready to present the business case. The benchmark we use for comparison is the fixed price-rate approach, where passengers are charged the same price-rate, irrespective of the number of potential passengers or the requested journeys. In particular, under the fixed price-rate policy, each passenger is charged at a rate given by the expected maximum price-rate passengers are prepared to pay, given by $\mathbb{E}[r_{i,\max}]$, which can be easily obtained from (5).

A. Key Trends

We now illustrate key trends in the expected profit and expected efficiency, based on a network setup with $K = 5$ drivers and up to $N = 13$ potential passengers. Importantly, this setting is practical—despite the small scale—when the serviced region is partitioned and separate negotiations are performed in each partition. Such an approach is desirable as suboptimal

heuristics do not need to be applied to ensure practical negotiation run-times. We demonstrate the effect of heuristics on a large scale network in Section V-B.

The pick-up and drop-off locations of passengers in the network are drawn from real locations in Prague, Czech Republic: $K = 5$ drivers; average vehicle velocity $\nu = 30$ km/hr; cost/km of 0.4 euros/km; and a maximum price-rate for each passenger of 3 euros/km. We further assume that there is an hour between each mechanism run, which means that the beginning of a passengers pick-up interval is uniformly distributed over the 60 minute interval. The maximum length of each passenger's pick-up interval is 10 minutes, with the actual interval length uniformly distributed.

We first demonstrate the performance mechanism with clustering using hard constraints (i.e., $\epsilon = 0$). We note that the mechanism with a fixed price-rate given by $\mathbb{E}[r_{i,\max}]$ and clustering with hard constraints is used as a benchmark.

Fig. 6 plots the expected profit per mechanism run for a varying number of potential passengers, N. The key observation is that our mechanism with optimized price-rate always improves the profit over the fixed price-rate approach, although the improvement depends on the passenger demand; i.e., how much passengers are willing to pay. Observe that there is a significant improvement in expected profit for both low and high demand, corresponding to $\alpha_r = 1, \beta_r = 3$ and $\alpha_r = 1, \beta_r = 1$, respectively. We also point out that rate of increase of expected profit with the number of passengers, N, is reduced as N increases. This is due to a saturation effect, where increasing the number of potential passengers in the network does not significantly improve the profit.

Next, we turn to the expected efficiency (see (2)) of our mechanism. In this case, observe that our mechanism improves the expected profit over the fixed price-rate approach for low demand with $N \geq 7$. This means that our mechanism outperforms the fixed price-rate approach in both expected profit and efficiency. On the other hand, the fixed price-rate approach has a higher efficiency for high demand. Next, we demonstrate the performance improvements that can be obtained by using our optimized clustering algorithm. In particular, this improves the efficiency in the case of high demand.

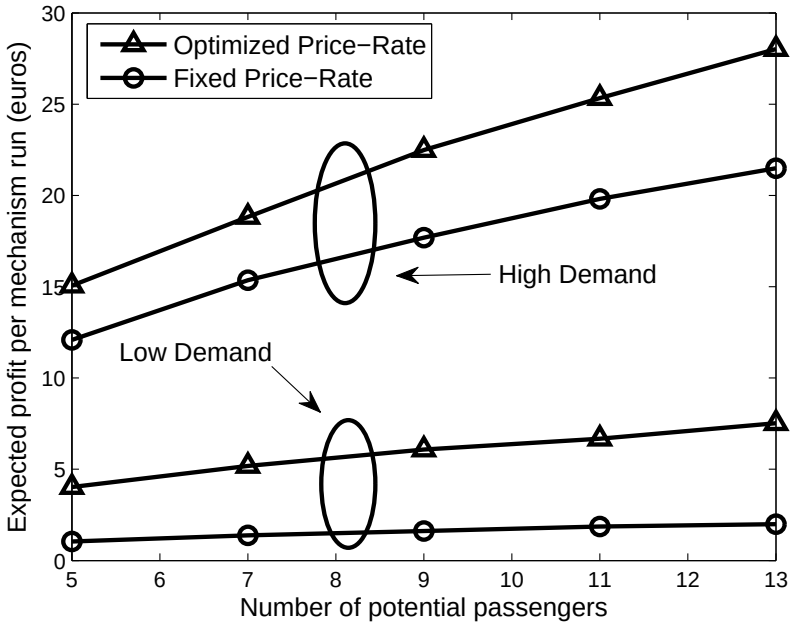


Fig. 6: *Plot of the expected profit for our mechanism with optimized price-rate and the standard fixed price-rate approach, for varying number of potential passengers,* N*. High demand corresponds to* $\alpha_r = 3$*,* $\beta_r = 1$*, medium demand corresponds to* $\alpha_r = \beta_r = 1$*, and low demand corresponds to* $\alpha_r = 1$*,* $\beta_r = 3$*. Simulation parameters:* $K = 5$*; average velocity of* 30 *km/hr; cost/km of* 0.4*; maximum price-rate of* 3 *euros/km; and hard feasibility constraints are enforced.*

In Table I, the expected profit is compared with the number of potential passengers, with and without optimized clustering. The optimized clustering solves (10) with $\epsilon_{step} = 0.2$, while the algorithm without optimized clustering enforces the hard constraints (i.e., $\epsilon = 0$). In both cases, the price-rate is optimized. Observe that our optimized clustering algorithm improves the expected profit, even with optimal pricing. In particular, gains of up to 4 euros per negotiation can be achieved. This means that the expected profit for the fixed price-rate approach can be improved significantly (up to 10 euros per negotiation with $\alpha_r = \beta_r = 1$) by using both optimized pricing and clustering.

In Table II, the expected efficiency (in euros/m) is compared with the number of potential passengers, with and without optimized clustering. Observe that the expected efficiency of opti-

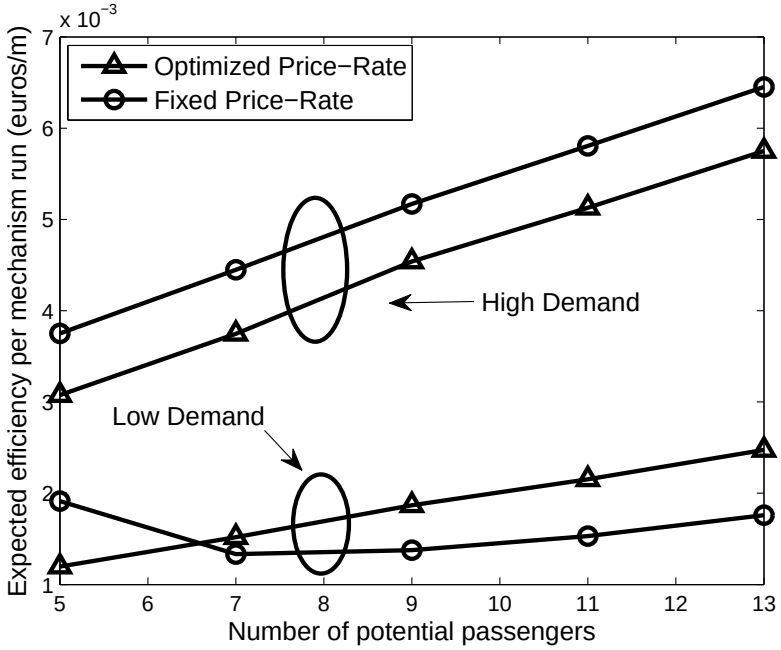


Fig. 7: *Plot of the expected efficiency for our mechanism with optimized price-rate and the standard fixed price-rate approach, for varying number of potential passengers,* N*. High demand corresponds to* $\alpha_r = 3$*,* $\beta_r = 1$*, medium demand corresponds to* $\alpha_r = \beta_r = 1$*, and low demand corresponds to* $\alpha_r = 1$*,* $\beta_r = 3$*. Simulation parameters:* $K = 5$*; average velocity of* 30 *km/hr; cost/km of* 0.4 *euros/km; maximum price-rate of* 3 *euros/km; and hard feasibility constraints are enforced.*

TABLE I: Expected profit per mechanism run (in euros) with and without optimized clustering. Parameters: $\alpha_r = \beta_r = 1$, $\alpha_\delta = 3$, $\beta_\delta = 1$.

Potential Passengers N	5	7	9	11	13
With Optimized Clustering (euros)	17.5	23.2	26.3	28.1	30.0
Without Optimized Clustering (euros)	14.5	18.8	22.0	25.5	27.8

mized clustering outperforms the expected efficiency without optimized clustering. Importantly, optimizing the clustering ensures that the expected efficiency of our mechanism is comparable with the efficiency using the fixed price-rate approach. As such, our mechanism can outperform

the fixed price-rate approach in both expected profit and expected efficiency.

TABLE II: Expected efficiency per mechanism run (in euros) with and without optimized clustering. Parameters: $\alpha_r = \beta_r = 1$, $\alpha_\delta = 3$, $\beta_\delta = 1$.

Potential Passengers N	5	7	9	11	13
Optimized Clustering (euros/m)	0.0038	0.0051	0.0063	0.0074	0.0085
Without Optimized Clustering (euros/m)	0.0030	0.0038	0.0044	0.0052	0.0057

B. The Effect of Heuristic Approximations

We now turn to simulation results for a network with 30 vehicles and up to 100 passengers. The average velocity, cost/km, maximum price-rate, and pick-up interval distributions are the same as in Section V-A.

In order to apply our mechanism to a network of this scale, it is necessary to approximate the objective of the optimization problem in (10), by truncating the sum involved in the expectation. Moreover, clustering is limited to the finding the first feasible cluster. Based on this approximation, we next compare our mechanism using these approximations with the fixed price-rate approach without clustering; i.e., each vehicle served up to one passenger per negotiation run.

Table III demonstrates the effect of the heuristic on the performance of the network. Observe that in all cases the optimized pricing in our mechanism outperformed the fixed price-rate approach, in terms of expected profit. Note that the expected profit for the fixed price-rate approach is approximately constant, irrespective of the number of passengers. This is due to the saturation effect also observed in Fig. 6. On the other hand, the expected profit for the optimized pricing approach reduces as the number of potential passengers increases, in contrast with the results in Section V-A. This is due to the approximation of the objective, which is less accurate as the number of potential passengers increases.

The results in Table III suggest that using our mechanism, the appropriate method to dealing with large-scale networks is to partition the network by simultaneously running several small-scale negotiations. This means that approximations of the objective function are not required,

TABLE III: Expected profit per mechanism run (in euros). Parameters: $\alpha_r = \beta_r = 1$, $\alpha_\delta = 3$, $\beta_\delta = 1$.

Potential Passengers N	30	60	90
Optimized Pricing (euros)	83.7	66.7	52.9
Fixed Price-Rate (euros)	52.6	51.7	52.4

which in turn ensures that the expected profit increases as the number of potential passengers increases.

VI. Conclusions

We have proposed a new market mechanism for third thread on-demand services, which enables negotiations with passengers to both increase provider profits and select passengers that value service the most. A key feature of our mechanism is that it jointly optimizes scheduling, routing and passenger pricing; in sharp contrast with standard approaches for services targeted at the elderly and disabled (first thread), and taxis (second thread).

Our mechanism is based on a new agent-based model, which emphasizes the role of provider profit in allocating resources. In particular, we have developed new models that incorporate price-based preferences for both the passengers and the provider. We also consider the effect of deviations from passenger requests in our resource allocation.

Our business case for third thread on-demand services demonstrated that our market mechanism improves the profitability of the service provider, compared with standard fixed price-rate approaches, while maintaining comparable efficiency.

Acknowledgments

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

APPENDIX A
PROOF OF PROPOSITION

Consider a time interval $[nT, (n+1)T]$ ($n \in \mathbb{N}$) between two consecutive mechanism runs, with $a \in [nT, (n+1)T)$. Then, the probability that a passenger's request is ignored is given by

$$P_{ignore} = 1 - \Pr(a + \Delta \geq (n+1)T). \tag{17}$$

Since request arrive according to homogeneous Poisson process, the following lemma gives the distribution of a.

Lemma 1. *The earliest possible pick-up time a is uniformly distributed in $[nT, (n+1)T)$, conditioned on the number of requests in the interval.*

This means that for a random request in the interval $[nT, (n+1)T)$, a is uniformly distributed.

Using Lemma 1 and conditioning on the time between the request arrival and desired pick-up time, Δ, we have

$$P_{ignore} = \int_0^\infty \Pr(a \geq (n+1)T - \delta | \Delta = \delta) \kappa e^{-\kappa\delta} d\delta. \tag{18}$$

Integrating by parts, we obtain the required result.

APPENDIX B
PROOF OF PROPOSITION

As for Proposition A, consider a time interval $[nT, (n+1)T)$ ($n \in \mathbb{N}$) between two consecutive mechanism runs, with $a \in [nT, (n+1)T)$. Then, the probability that a passenger cannot be serviced before the next mechanism run is given by

$$\begin{aligned} P_{overtime} &= \Pr(a + \nu z \geq (n+1)T) \\ &= \int_0^\infty \Pr(a \geq (n+1)T - \nu z) f_Z(z) dz, \end{aligned} \tag{19}$$

which follows by conditioning on the distance, Z. We note there is no loss in generality by assuming that the pick-up location is at the origin by Slivnyak's theorem [17].

Using Lemma 1 and the fact that $a \in [nT, (n+1)T)$, it follows that

$$\begin{aligned} P_{overtime} &= \frac{\nu}{T}\int_0^{T/\nu} z f_Z(z)dz + \int_{T/\nu}^{\infty} f_Z(z)dz \\ &= \frac{\nu}{T}\int_0^{T/\nu} 2\pi\zeta z^2 e^{-\pi\zeta z^2}dz + e^{-\pi\zeta T^2/\nu^2}. \end{aligned} \tag{20}$$

Next we integrate by parts[3] to obtain

$$P_{overtime} = \frac{\nu}{T}\left(\frac{T}{\nu}e^{-\pi\zeta T^2/\nu^2} + \int_0^{T/\nu} e^{-\pi\zeta z^2}dz\right) + e^{-\pi\zeta T^2/\nu^2}. \tag{21}$$

We then apply the change of variables $u = \sqrt{2\pi\zeta}z$ and the definition of the Q-function, which is given by

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty} e^{-u^2/2}du. \tag{22}$$

Using the fact that $Q(0) = \frac{1}{2}$, we then have

$$P_{overtime} = \left(1 - \frac{2\pi\zeta T}{\nu}\right)e^{-\pi\zeta T^2/\nu^2} + \frac{\nu}{T\sqrt{\zeta}}\left(\frac{1}{2} - Q\left(\frac{T\sqrt{2\pi\zeta}}{\nu}\right)\right), \tag{23}$$

as required.

REFERENCES

[1] J.-F. Cordeau, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, pp. 29–46, 2007.

[2] D. Barbucha, "A multi-agent approach to the dynamic vehicle routing problem with time windows," in *Proc. of the Internation Conference on Computational Collective Intelligence*, 2013.

[3] C. Cubillos, F. Guido-Polanco, and C. Demartini, "Madarp: multi-agent architecture for passenger transportation systems," in *Proc. IEEE International Conference on Intelligent Transport Systems*, 2005.

[4] H. Yang, C. Fung, K. Wong, and S. Wong, "Nonlinear pricing of taxi services," *Transportation Research Part A: Policy and Practice*, vol. 44, no. 5, pp. 337–348, 2010.

[5] K. Seow, N. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, 2010.

[6] R. Bai, J. Li, J. Atkin, and G. Kendell, "A novel approach to independent taxi scheduling problem based on stable matching," *Journal of the Operational Research Society*, vol. 65, 2014.

[7] H. Yang and T. Yang, "Equilibrium properties of taxi markets with search frictions," *Transportation Research Part B: Methodological*, vol. 45, no. 4, pp. 696–713, 2011.

[8] R. Balan, N. Khoa, and L. Jiang, "Real-time trip information service for a large taxi fleet," in *Proc. International Conference on Mobile Systems, Applications, and Services*, 2011.

[3] We use the fact that $\int 2\pi\zeta z e^{-\pi\zeta z^2}dz = -e^{-\pi\zeta z^2}$.

[9] A. Glaschenko, A. Ivaschencko, G. Rzevski, and P. Skobolev, "Multi-agent real time scheduling system for taxi companies," in *Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

[10] C. Zeng and N. Oren, "Dynamic taxi pricing," in *In Proc. European Conference on Artificial Intelligence*, 2014.

[11] M. Egan and M. Jakob, "A profit-aware negotiation mechanism for on-demand transport services," in *In Proc. European Conference on Artificial Intelligence*, 2014.

[12] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance*, vol. 1, pp. 223–236, 2001.

[13] L. Tesfatsion, "Introduction to the special issue on agent-based computational economics," *Journal of Economic Dynamics and Control*, vol. 25, no. 3-4, pp. 281–292, 2001.

[14] B. Verplanken and W. Wood, "Interventions to break and create customer habits," *Journal of Public Policy and Marketing*, vol. 25, no. 1, pp. 90–103, 2006.

[15] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[16] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," *European Journal of Operations Research*, vol. 54, pp. 7–22, 1991.

[17] F. Baccelli and B. Blaszczyszyn, *Stochastic Geometry and Wireless Networks: Volume 1: Theory*. Now Publishers, Inc., 2009.

Bibliography

[1] S. Alpern, A. Morton, and K. Papadaki. Patrolling games. *Operations research*, 59(5):1246–1257, 2011.

[2] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. W. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of MATSim-T, 2008.

[3] M. Barth and M. Todd. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7(4):237–259, 1999.

[4] H. Bast, D. Delling, A. Goldberg, M. Muller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. Werneck. Route Planning in Transportation Networks. Technical report, Microsoft Research, 2014.

[5] A. L. Bazzan, F. Klügl, and S. Ossowski. Agents in traffic and transportation: Exploring autonomy in logistics, management, simulation, and cooperative driving. *Transportation Research Part C: Emerging Technologies*, 13(4):251–254, 2005.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[7] G. Berbeglia, J. F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[8] R. Boel and L. Mihaylova. A compositional stochastic model for real time freeway traffic simulation. *Transportation Research Part B: Methodological*, 40(4):319–334, 2006.

[9] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7280–7287, 2002.

[10] B. Bošanský, V. Lisý, M. Jakob, and M. Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 989–996, 2011.

[11] S. Bourdon, Y. Gauthier, and J. Greiss. MATRICS: A maritime traffic simulation. Technical report, Defence R&D Canada, 2007.

[12] J. Broach, J. Dill, and J. Gliebe. Where do cyclists ride? A route choice model developed with revealed preference GPS data. *Transportation Research Part A: Policy and Practice*, 46(10):1730 – 1740, 2012.

[13] I. Carreras, S. Gabrielli, D. Miorandi, A. Tamilin, F. Cartolano, M. Jakob, and S. Marzorati. SUPERHUB: a user-centric perspective on sustainable urban mobility. In *6th ACM workshop on Next generation mobile computing for dynamic personalised travel planning*, pages 9–10. ACM, 2012.

[14] C.-Y. Chan. Connected vehicles in a connected world. In *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–4, 2011.

[15] B. Chen and H. H. Cheng. A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):485–497, 2010.

[16] S.-F. Cheng and T. D. Nguyen. Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 14–21, 2011.

[17] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31, 2006.

[18] F. Ciari, M. Balmer, and K. W. Axhausen. Concepts for large-scale carsharing system: Modeling and evaluation with agent-based approach. In *Transportation Research Board 88th Annual Meeting*, number 09-1888, pages 1–23, 2009.

[19] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

[20] J. J. Dabrowskia and J. P. de Villiersb. Maritime piracy situation modelling with dynamic bayesian networks. *Information Fusion*, 9:90–99, 2013.

[21] P. M. d'Orey, R. Fernandes, and M. Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *15th International IEEE Conference on Intelligent Transportation Systems*, pages 140–146. IEEE, 2012.

[22] S. Druitt. Introduction to microsimulation. *Traffic engineering & control*, 39(9), 1998.

[23] M. Egan and M. Jakob. Market mechanism design for profitable on-demand transport services. *Transportation Research Part B: Methodological*, (submitted), 2014.

[24] M. Egan and M. Jakob. A profit-aware negotiation mechanism for on-demand transport services. In *European Conference on Artificial Intelligence (ECAI)*, pages 273 – 278, 2014.

[25] S. Gal. *Search Games*. Academic Press, New York, 1980.

[26] K. Hasegawa, K. Hata, M. Shioji, K. Niwa, S. Mori, and H. Fukuda. Maritime traffic simulation in congested waterways and its applications. In *4th Conference for New Ship and Marine Technology*, pages 195–199, 2004.

[27] M. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188, 2002.

[28] J. Hrnčíř, M. Rovatsos, and M. Jakob. Ridesharing on timetabled transport services: A multiagent planning approach. *Journal of Intelligent Transportation Systems*, (accepted, available on-line), 2014.

[29] J. Hrnčíř and M. Jakob. Generalised time-dependent graphs for fully multimodal journey planning. In *16th IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2138–2145, 2013.

[30] J. Hrnčíř, Q. Song, P. Žilecký, M. Nemet, and M. Jakob. Bicycle route planning with route choice preferences. In *ECAI 21st European Conference on Artificial Intelligence - Including Prestigious Applications of Intelligent Systems (PAIS)*, pages 1149–1154, 2014.

[31] M. Jakob, J. Hrnčíř, L. Oliva, F. Ronzano, P. Žilecký, and J. Finnegan. Personalized fully multimodal journey planner. In *ECAI 21st European Conference on Artificial Intelligence - Including Prestigious Applications of Intelligent Systems (PAIS)*, pages 1225 – 1226, 2014.

[32] M. Jakob and Z. Moler. Modular framework for simulation modelling of interaction-rich transport systems. In *16th IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2152–2159, 2013.

[33] M. Jakob, Z. Moler, A. Komenda, Z. Yin, A. X. Jiang, M. P. Johnson, M. Pěchouček, and M. Tambe. AgentPolis: towards a platform for fully agent-based modeling of multi-modal transportation. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1501–1502, 2012.

[34] M. Jakob, M. Pěchouček, M. Čáp, P. Novák, and O. Vaněk. Mixed-reality testbeds for incremental development of HART applications. *IEEE Intelligent Systems*, 27(2):1541–1672, 2012.

[35] M. Jakob, O. Vaněk, B. Bošanský, O. Hrstka, and M. Pěchouček. AgentC: agent-based system for securing maritime transit. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1309–1310. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[36] M. Jakob, O. Vaněk, O. Hrstka, and M. Pěchouček. Agents vs. Pirates: multi-agent simulation and optimization to fight maritime piracy. In *11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 37–44. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[37] M. Jakob, O. Vaněk, and M. Pěchouček. Using agents to improve international maritime transport security. *Intelligent Systems, IEEE*, 26(1):90–96, 2011.

[38] M. Jakob, O. Vaněk, S. Urban, P. Benda, and M. Pěchouček. Employing agents to improve the security of international maritime transport. In *AAMAS 2010 Workshop on Agents In Traffic and Transportation*, 2010.

[39] A. Jonsson and M. Rovatsos. Scaling up multiagent planning: A best-response approach. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–121, June 2011.

[40] S. Kikuchi (ed.). Artificial intelligence applications to critical transportation issues. *Transportation Research Circular*, (E-C168), 2012.

[41] L. M. Martinez, G. Correia, and J. Viegas. An agent-based model to assess the impacts of introducing a shared-taxi system in Lisbon (Portugal). In *7th International Workshop on Agents in Traffic and Transportation*, 2012.

[42] E. Q. V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236 – 245, 1984.

[43] J. H. Miller and S. E. Page. *Complex Adaptive Systems: An Introduction to Computational Models of Social Life (Princeton Studies in Complexity)*. Princeton University Press, Princeton, NJ, USA, 2007.

[44] J. Nykl, M. Jakob, and J. Hrnčíř. Advanced public transport network analyser. In *ECAI 21st European Conference on Artificial Intelligence - Including Prestigious Applications of Intelligent Systems (PAIS)*, pages 1229–1230, 2014.

[45] W. Ruckle, R. Fennell, P. T. Holmes, and C. Fennemore. Ambushing random walks I: Finite models. *Operations Research*, 24:314–324, 1976.

[46] R. Schleiffer. Intelligent agents in traffic and transportation. *Transportation Research Part C: Emerging Technologies*, 10(5):325–329, 2002.

[47] Q. Song, P. Žilecký, M. Jakob, and J. Hrnčíř. Exploring pareto routes in multi-criteria urban bicycle routing. In *17th IEEE Intelligent Transportation Systems Conference (ITSC)*, 2014.

[48] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[49] O. Vaněk, B. Bošanský, M. Jakob, V. Lisy, and M. Pěchouček. Extending security games to defenders with constrained mobility. In *AAAI Spring Symposium: Game Theory for Security, Sustainability, and Health*, 2012.

[50] O. Vaněk, B. Bošanský, M. Jakob, and M. Pěchouček. Transiting areas patrolled by a mobile adversary. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 9–16. IEEE, 2010.

[51] O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček. Using multi-agent simulation to improve the security of maritime transit. In *12th Workshop on Multi-Agent-Based Simulation*, pages 44–58. 2012.

[52] O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček. Agent-based model of maritime traffic in piracy-affected waters. *Transportation Research Part C: Emerging Technologies*, 36:157–176, 2013.

[53] O. Vaněk, M. Jakob, V. Lisý, B. Bošanský, and M. Pěchouček. Iterative game-theoretic route selection for hostile area transit and patrolling. In *The 10th International Conference on Autonomous Agents (AAMAS)*, pages 1273–1274, 2011.

[54] O. Vaněk, M. Jakob, and M. Pěchouček. Using data-driven simulation for analysis of maritime piracy. *Prediction and Recognition of Piracy Efforts Using Collaborative Human-Centric Information Systems*, 109:109, 2013.

[55] A. E. Varol and M. M. Gunal. Simulation modeling of maritime piracy using discrete event and agent-based approaches. In *SIMULTECH*, pages 438–445, 2013.

[56] M. Čertický, M. Jakob, and R. Píbil. Simulation testbed for autonomic demand-responsive mobility systems (to appear). In F. Kluegl, A. Kotsialos, J. P. McCluskey, Lee Mueller, O. Rana, and R. Schumann, editors, *Autonomic Road Transport Support Systems*, 2015.

[57] M. Čertický, M. Jakob, R. Píbil, and Z. Moler. Agent-based simulation testbed for on-demand mobility services. In *3rd International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS)*, 2014.

[58] M. Čertický, M. Jakob, R. Píbil, and Z. Moler. Agent-based simulation testbed for on-demand transport services (demonstration). In *13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1671–1672, 2014.

[59] Z. Šitavancová and M. Hájek. *Intelligent transport systems: thematic research summary*. European Commission, 2009.

[60] G. Weiss (ed.). *Multiagent Systems (2nd edition)*. MIT press, 2013.

[61] S. Winter, M. Sester, O. Wolfson, and G. Geers. Towards a computational transportation science. *Journal of Spatial Information Science*, (2):119–126, 2014.