TRANSPORTATION SCIENCE
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0041-1655 | EISSN 1526-5447 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Simulation and dynamic optimization of taxi services in MATSim

Michał Maciejewski
Institute of Machines and Motor Vehicles, Faculty of Machines and Transportation, Poznan University of Technology, ul. Piotrowo 3, 60-965 Poznan, Poland, michal.maciejewski@put.poznan.pl,
Transport Systems Planning (VSP), Institute for Land and Sea Transport Systems, TU Berlin, Salzufer 17-19 Sekr. SG12, D-10587 Berlin, Germany, maciejewski@vsp.tu-berlin.de,

Kai Nagel
Transport Systems Planning (VSP), Institute for Land and Sea Transport Systems, TU Berlin, Salzufer 17-19 Sekr. SG12, D-10587 Berlin, Germany, nagel@vsp.tu-berlin.de,

This paper describes how an optimizer for the dynamic vehicle routing problem ('DVRP Optimizer') is coupled to a microscopic, behavior-based traffic simulation ('MATSim'). The traffic simulation both generates the demand for taxicab trips, and provides the congested network simulation in which the taxicabs operate. The present implementation has passengers request a taxi when they depart; then they wait until they are being picked up and delivered to their destination. Afterwards, taxicabs will drive to the next request assigned to them, or remain idle until the next request arrives. Next, the paper defines the *off-line* and *on-line* taxi dispatching problems, and presents three different dispatching strategies that are then evaluated on a realistic scenario. The computational results show that a no-scheduling strategy – just assigning the nearest empty taxi to each incoming request – works well as long as the system is not under heavy load. A good distance metric – preferably taking time-dependent congestion into account – has a positive impact. Under heavy load, however, the no-scheduling strategy clearly deteriorates, and scheduling strategies show very clear advantages. The approach is constructed in a way that (1) other dispatch algorithms can be tested, and that (2) it can process real world scenarios.

Key words: dynamic taxi dispatching; dynamic vehicle routing; on-line optimization; MATSim; traffic flow simulation; simulation-based optimization

History:

1. Introduction

1.1. Optimization of taxi services

Optimization of taxi services may be considered from various perspectives and at different levels of detail. There have been numerous studies related to organizing the taxi service on the city level. Particularly, the issue of competition and regulation in the taxi market has been extensively explored for over 40 years (e.g. Douglas 1972). Whether it is fleet sizing, pricing or other means of regulating the market, the research focuses mostly on finding the demand-supply equilibrium in a regulated or deregulated market (e.g. Cairns and Liston-Heyes 1996, Arnott 1996). These studies have considered the properties of taxi services to be homogeneous in time and space, and often limit the scope only to cruising or dispatching taxis. Recently, to address the issue of spatial heterogeneity of taxi demand, Yang et al. have proposed a time-dependent network-oriented model based on multiple *origin-destination* (OD) matrices (Yang and Wong 1998, Yang et al. 2005). Regardless of the approach used, the research in this area is carried out on the macroscopic level and deals with the long-term, strategic management.

On the opposite pole, there are issues focused on various aspects of managing a taxi fleet at the operational level, and where each taxi and each request is considered separately (the microscopic level of detail). However, prior to the appearance of ICT (*Information and communications technology*) allowing for real-time coordination of a fleet, these problems had not been in the centre of interest; one of very few studies had been done by Bailey Jr and Clark Jr (1992). Over the last decade, the on-line management of taxis has been gaining popularity, as ICT offers great opportunities for managing the service even in real time. There are several theoretical studies investigating on-line taxi dispatching algorithms, for instance, Ma and Wang (2007) have formulated the *On-line Weighted k-Taxi Problem*, originating from the *On-line k-Server Problem*, and then designed several algorithms and analysed their competitiveness ratio. In most cases, however, sophisticated dynamic routing approaches are hard to analyse theoretically, and thus simulation tools are used. In some studies multi-agent simulation is used to model (partial) independence of taxi drivers (e.g. Cheng and Nguyen 2011, Seow et al. 2010, Alshamsi et al. 2009), other studies assume that taxi dispatching is arranged in a centralised form (e.g. Lee et al. 2004, Wang et al. 2009).

The second category of problems related to the operational management of taxi service, though beyond the scope of this paper, is taxi cruising. Here, the central focus is finding such cruising paths, often ending at taxi ranks, that increase *live miles* (with passengers aboard) and simultaneously reduce *cruising miles*. Powell et al. (2011) have proposed *Spatio-Temporal Profitability* (STP) maps for guiding taxicabs to the most profitable taxi locations. Yuan et al. (2011) have developed a recommendation system for both taxi drivers and passengers.

Despite the growing interest in taxi dispatching, this area of research still lacks extensive studies. However, as problems related to on-line taxi dispatching are special cases of the *General Vehicle Pickup and Delivery Problem* (e.g. Savelsbergh and Sol 1995, Berbeglia et al. 2010), there are many similar problems within the GVPDP family that have been studied more thoroughly. One of them is the *Dynamic Single Load Pickup and Delivery Problem* (Fleischmann et al. 2004), often referred to as *Real-time Multivehicle Truckload Pickup and Delivery Problem* (Yang et al. 2004), where a vehicle, like in taxi dispatching, can serve only one request at once, but the time windows are usually much broader and the demand is less dynamic. Another closely-related problem is the *On-line Dial-a-Ride Problem* (Ascheuer et al. 2000), where trips can be shared and the demand is less dynamic (relatively more request are submitted in advance). Also taxis are often included into the *Demand-Responsive Transport* services, as an alternative transport mode to minibuses (Horn 2002). Krumke et al. (2002) have investigated real-time dispatching of guided and unguided automobile service units with soft time windows, which also has a lot of in common with taxi dispatching. Last but not least, dynamic management of emergency services, especially assignment and dynamic redeployment of vehicles (Gendreau et al. 2001), is closely related to dynamic dispatching of taxis.

1.2. Simulation of dynamic vehicle dispatching and routing

Since theoretical analyses of on-line optimization methods, such as competitive analysis, are of limited applicability, the simulation approach is frequently used to evaluate, compare and refine dynamic algorithms (Grötschel et al. 2001). However, the quality of results obtained with this approach depends on the simulation method. In transport-related problems, simulation has to incorporate realistically modeled dynamism of customer demand, traffic flow phenomena and fleet management operations. These aspects are even more crucial when considering urban areas due to high dynamics of traffic flow resulting in continuously changing travel times and often, depending on the type of services, in high volatility of demand (e.g. taxi).

In recent years, several approaches that combine vehicle routing and traffic simulation have been proposed and implemented. In one of the first works in this field, Regan et al. (1998) have proposed a simplified simulation framework for the evaluation of dynamic fleet management systems for truckload carrier operations. Taniguchi et al. (2001) have analyzed the integration of vehicle routing optimization and traffic simulation for optimization of city-logistics oriented problems. Another study is an application of the AIMSUN simulator for optimization of the VRP in cities (Barcelo et al. 2007). Liao et al. (2008) have developed a system for the simulation-based evaluation of DVRP strategies using real-time information.

Particularly in the case of taxi dispatching, the use of microscopic traffic simulators allows for evaluating the performance of the service under different, often extreme, scenarios, such as

sport/cultural events, bad weather conditions or public transport strikes. These issues, however, remain almost unexplored. To the best knowledge of the authors, traffic flow simulators have been applied only in Singapore; Lee et al. (2004) have used Paramics[1] to simulate taxi dispatching in Singapore's Central Business District, whereas Seow et al. (2010) have coupled NTuCab taxi dispatching software with the MITSIMLab microscopic traffic simulator (Ben-Akiva et al. 2001) and run simulations for a 15 km x 10 km urban area.

All these approaches do not include realistic on-line demand generation; one cannot, for instance, model the impact of traffic or transport service availability on customer demand. Moreover, the systems were used only for small-scale problems, where a road network was of limited size and the number of customers was not high.

1.3. Large scale microscopic traffic simulation

In order to keep track of the taxicab vehicles, it is necessary to simulate them individually. For the same reason, requests need to be microscopic, i.e. there need to be discrete requests with the parameters time of request, pickup location, and drop-off location.[2] If one wishes, e.g. for future developments, that demand is able to switch between taxicab and other modes, then all the demand that could potentially use a taxicab needs to be microscopic. This line of argument motivates to embed the taxicab simulation into a microscopic, behavior-oriented traffic simulation where *all* travelers are simulated individually.

Although several companies offer micro- or "nano-"simulations,[3] they are either not able to perform network loadings with millions of persons/vehicles, or they do not trace persons or vehicles throughout the whole day. Non-commercial approaches include TRANSIMS[4], SUMO[5], MEZZO (Burghout 2004, Burghout and Koutsopoulos 2009), and MATSim[6]. Out of these, MATSim is arguably the one with least focus on traffic flow realism but with the highest computing speed and the best behavioral model on the trip planning side (the input for the optimized service). MATSim will be used for the present investigation since it is planned to investigate large scenarios and thus high computing speed is imperative. Also see (Maciejewski and Nagel 2012) and references therein for additional justification why MATSim was selected.

[1] `http://www.paramics.com`

[2] The time of the request could, in fact, be different from the desired pickup time. This will be considered in future studies.

[3] `http://www.vissim.de`, `http://azalient.com`, `http://www.savannah-simulations.com`, `http://dynust.net`, `http://www.aimsun.com`, `http://www.paramics.com`

[4] `http://code.google.com/p/transims/`

[5] `http://sumo.sourceforge.net`

[6] `http://www.matsim.org`

1.4. Outline

This paper describes how MATSim can be used to generate problem instances for dynamic vehicle routing. This is achieved by coupling a DVRP Optimizer (Maciejewski and Nagel 2012, Sec. 3), Sec. 3 into MATSim. Thus, during a normal MATSim traffic flow simulation, there will be synthetic travelers that have taxicab as their mode of transport. They will call for a taxi when their preceding activity has ended. The DVRP Optimizer will dispatch a taxi, the taxi will drive to the customer, the customer will be delivered to his or her destination, etc. Since MATSim can be started with different random seeds, this can provide a new instance every time it is run. Also, the system can be run with different demands, e.g. for normal workdays vs. weekend days, and with different mode choice functions, leading to larger or smaller shares of taxicab trips. The overall approach is a bit similar to tournament platforms such as for soccer(Kim 1997), rescue operations (Takahashi et al. 2002), or energy trading (Block et al. 2010).

The paper starts by short descriptions of MATSim and the DVRP Optimizer, and a longer description of how taxicabs are modeled (Sec. 2). Sec. 3 discusses taxi dispatching algorithms. In particular, it describes several on-line strategies that are used in the simulations. Sec. 4 describes the simulation scenario, based on the Polish city of Mielec, and the performance measures that are collected during the simulations. Results are presented and discussed in Sec. 5. The paper finishes with a discussion and conclusions.

2. Integration of MATSim and the DVRP Optimizer

The system consists of two fundamental components, namely MATSim and the DVRP Optimizer. The first one is used for modeling transport supply (including the taxicab fleet) and demand (including the taxi demand) and providing queue-based traffic flow simulation at microscopic level. The other one is responsible for managing a fleet of taxis (or, in general, any vehicle fleet) within the simulation. Since both components are separate and independent programs written in Java, their coupling was done by means of the MATSim–DVRP Optimizer connector module that is responsible for setting up the whole system and managing data flow and control flow.

The aim of this section is to provide the reader with details about relationship between the MATSim and DVRP Optimizer data domains. Algorithms for the management of the taxicab fleet are then presented in the following section.

2.1. MATSim

MATSim is an agent-based system for transport simulation with the primary focus on transport planning. It allows for disaggregate activity-based modeling that consists of three main phases which are run iteratively: planning, traffic flow simulation (also called network loading), and scoring. The first phase (the planning phase) is used to create (zeroth iteration) or modify (subsequent

iterations) the agents' daily plans, each consisting of activities and legs connecting the locations of subsequent activities. Next, during the simulation phase, all planned legs (along with activities) are executed by means of a queue-based traffic flow simulator. Within this simulation links are represented as FIFO queues with a set of parameters, among them: length, free-flow speed, flow capacity, and storage capacity. The result of the traffic flow simulation is a set of events documenting changes in the state of any object (not only the agents) having been simulated. During the third phase (the scoring phase) the plans are evaluated against their actual execution (recorded in event logs). The obtained scoring is then used for choosing and modifying plans in the planning phase in the next iteration. The simulation ends after a termination criteria, based on a measure of the system relaxation, is met.

In this paper, relaxation will be used to generate a "relaxed" initial state. For the taxicab runs, the taxicab demand and supply will be added, and the DVRP Optimizer will be responsible to dispatch the taxicabs. All other agent learning will be switched off at that point, i.e. there are not more iterations.

2.2. DVRP Optimizer

The DVRP Optimizer is a Java framework for optimizing problems related to vehicle routing, dispatching and scheduling. The software is intended to be as general and customizable as possible. The framework is divided into three main parts, namely *data* (represents instances of various problems), *optimizer* (creates routes/schedules for given input data), *simulator* (runs different scenarios).

Currently, the framework allows to model a wide range of one-to-many (many-to-one) routing problems, while many-to-many problems are limited to single load cases (dispatching). Currently, hard time windows and time-dependent travel times and costs are supported. Due to the framework's flexibility, one can easily extend the model to cover other specific cases. By default, the DVRP Optimizer uses a memetic algorithm for solving the *Dynamic Multi-Depot Vehicle Routing Problem with Time Windows and Time-Dependent Travel Times and Costs*. In this study, however, this algorithm was replaced with a queue-based one created exactly for taxi dispatching (see Sec. 3). The standard simulator offers simple event-based simulation of *customer service* (request submission, modification and cancellation), *fleet management* (dispatching and monitoring of vehicles) and *traffic monitoring* (changes in the travel times and costs). To obtain more realistic behavior, one can use a more sophisticated simulator, which has been done in this study by integrating MATSim and the DVRP Optimizer. More details on the framework are provided in (Maciejewski and Nagel 2012).

2.3. The integration idea

The DVRP Optimizer takes requests by customers as well as vehicle positions as input and provides schedules to customers and vehicles as output. At the same time, it monitors the traffic state. Schedules can either be provided at arbitrary times, e.g. after each customer request or after every customer drop-off.

Since both the DVRP Optimizer and MATSim are written in JAVA, the integration was done at the software level. MATSim agents perform sequences of activities and legs. When an activity ends, the following leg is started by calling a departure handler corresponding to the leg's mode. In consequence, a departure handler was added that forwards the agent's taxicab mode request to the DVRP Optimizer. The DVRP Optimizer will decide on a taxicab vehicle, and send that vehicle to the waiting customer. The customer will be picked up and driven to his or her destination, at which point the customer starts his or her next activity while the taxicab waits for the next customer.

An early version of the coupling is described in (Maciejewski and Nagel 2012). There, the coupling was fully *off-line*. That is, MATSim was run and the taxicab requests as well as the traffic states (time-dependent link travel times) where recorded. The DVRP Optimizer was subsequently run on that output. There was, however, no feedback into the traffic flow simulation. In consequence, the performance of the traffic system was exactly as anticipated by the DVRP Optimizer – taxicab vehicles would always arrive exactly as predicted. In consequence, the real-time reactive capabilities of the DVRP Optimizer were not tested. The present investigation now inserts the taxicab simulation fully into the MATSim context. Taxicabs are now real vehicles inside MATSim, and are moved as part of the normal traffic flow dynamics. In consequence, they are now also part of the normal stochastic delays that occur inside the simulation.

2.4. Network

In MATSim, the network consists of nodes connected by one-way links. Since the traffic flow simulation uses a queue-based approach (Simão and Powell 1992, Gawron 1998, Simon et al. 1999), links constitute the most basic network elements on which the simulation operates. Each link is a FIFO queue and has a set of parameters describing traffic flows (e.g. length, number of lanes, flow capacity, free-flow speed, etc.). Any trip in the network is a sequence of links, where the first link is the one closest to the origin location and the last link is the one closest to the destination. Each link may be a possible activity location. Because of the queue nature of the simulation, parking can be modeled either at the beginning or at the end of a link; in MATSim the second behavior is used. In consequence, the last links in routes are traversed entirely, while the first ones are skipped.

The DVRP Optimizer, on the other hand, operates on a directed graph made up of vertices and directed arcs. The graph structure serves as an abstraction layer constructed over the network

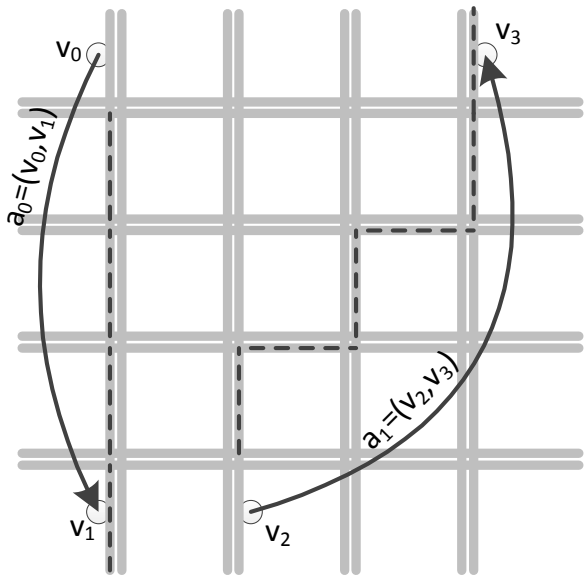

Figure 1 VRP graph built upon a MATSim network

structure, which is crucial to preserve the independence of both modules. Each vertex represents one location (which corresponds to a link in MATSim) and each arc is the shortest path (a sequence of links in the MATSim network) leading from the link represented by the arc's tail to the link represented by the arc's head (see Fig. 1).

Since arc travel times are time-dependent, they are calculated for a given departure or arrival time by means of shortest path search algorithms available in MATSim (Jacob et al. 1999, Lefebvre and Balmer 2007). Since MATSim calculates travel time statistics for links for every 15-minute time period by default, and these statistics are used in the next iteration (i.e. the next simulation) for shortest paths computations, the 15-minute time bin is also used for computing shortest paths and associated travel times and costs in the DVRP Optimizer.

Given that it is possible to travel between any two links (locations) in the network, the corresponding VRP graph is complete. However, building and storing a complete graph, along with the all shortest paths for each arc for each time period, may be prohibitively time- and memory-consuming. For instance, for a small-size network of 1000 links, the complete graph would consist of 1000 vertices and 999'000 arcs. Furthermore, a 24-hour simulation with the 15-minute shortest path calculation resolution, would lead to 96 shortest paths per arc, and altogether to 95'904'000. If one wanted additionally to search paths backwards (i.e. based on arrival times), that number would even double. Although calculating and storing shortest paths for a real-size network would pose a real challenge itself, most of the efforts would be wasted since only a tiny fraction of the calculated data would be later used in the taxi dispatching algorithm. To avoid that inefficiency, in the default implementation of the graph, both vertex and arc objects are created on demand.

Moreover, after the creation, arc objects do not contain any shortest path data, since they are calculated on-the-fly and then cached for possible future re-use. Since travel times change from iteration to iteration, that cache is flushed at the end of every iteration.

It should be noted that time-dependent travel times calculated for arcs are not exact. One of the causes is that they are calculated with the 15-minute resolution of the departure time. But even if the departure time is given precisely, the expected travel times are derived from historical link travel-time statistics (from the previous MATSim iteration). Since a proportion of agent plans undergoes modifications between consecutive simulations, traffic changes from iteration to iteration and thus the experienced travel times are not identical with the historical ones. Moreover, the link travel-time statistics are averaged for each time period (usually 15 minutes) and therefore also contribute for limited preciseness. As a result, the FIFO property on the level of links is not preserved on the arc level unless vehicles move along exactly the same path. However, with reasonably fine-grained discretization of time (both for arc travel times and link statistic calculations), validation of the FIFO property should be negligible. Additionally, stochasticity and impreciseness of travel times is typical in real-world applications, hence one should address this problem while developing robust algorithms. Therefore, we do not consider small violations of the FIFO property unacceptable.

2.5. Agents

In MATSim each vehicle is driven by a driver agent. Agents of this type have special functionality that allows them to be embedded into the queue-based simulation and let them decide where to drive.[7] By default, all agents in MATSim arrange their daily plans during the replanning phase, before traffic flow simulation starts. During the traffic flow simulation, they just follow their prearranged plans. This behavior limits the agents' autonomy since they cannot dynamically react to the current situation. Adaptation is achieved by letting the agents learn from one iteration to the next, also called day-to-day learning (e.g. Cascetta and Cantarella 1991).

The day-to-day learning simplification was made since it greatly reduces the demand for processing power and thus allows for running large-scale simulations with millions of agents. However, dynamic adaptation to the current state of the simulation is a prerequisite for dispatching taxicabs: Other than for regular commuter traffic, there is no plausible interpretation for a day-to-day learning approach to taxicab operations. For that reason, instead of having a precomputed plan, each taxi driver agent follows his/her own schedule that changes dynamically over time (the DVRP Optimizer domain). Each schedule is a sequence of the following three types of tasks:

[7] Besides the driver agents, there are other types available in MATSim, e.g. (public transport) passenger agents

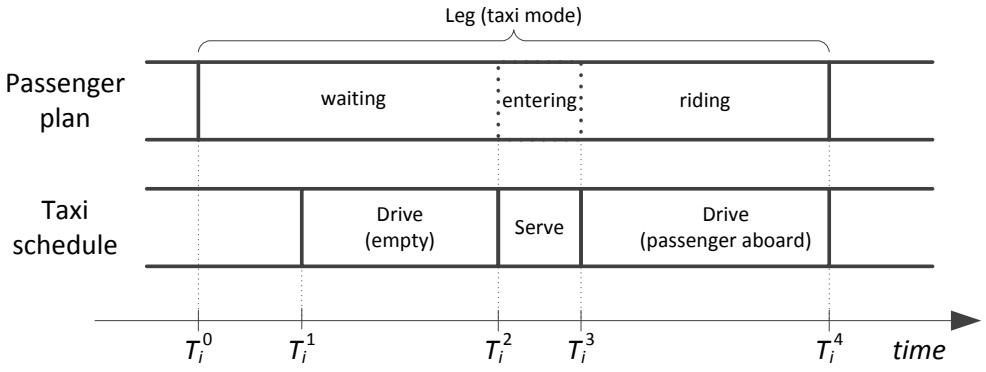

Figure 2 A planned taxi leg and the corresponding sequence of taxi tasks

• `DriveTask` – driving along a given arc (i.e. between two vertices); this task is executed in MATSim as a leg along the shortest path between the corresponding pair of links for a given departure or arrival time

• `ServeTask` – picking-up a passenger at a given vertex; this task is executed in MATSim as an activity at the corresponding link and of a given duration

• `WaitTask` – waiting at a given vertex until new request is assigned to the taxicab; this task is executed in MATSim as an activity at the corresponding link without a given a priori duration[8]

Concerning the demand side, an agent of any type can choose to travel by taxi – no special functionality (interface) is required to become a taxi passenger. The mode choice is performed during the planning phase by assigning a mode to each leg.

A connection between a taxi driver agent and a taxi passenger agent is illustrated in Fig. 2. When an agent wants to take a taxi, he/she calls it. At this moment (time T_i^0), the agent is registered as a customer and his/her request (request i) is assigned to one of the taxis based on the the dispatching algorithms (see Sec. 3). From this moment on, the customer waits until the taxi arrives at the pick-up location (time T_i^2). It may happen that the taxi assigned must first complete other tasks and then (time $T_i^1 > T_i^0$) set out for this customer. When the taxi arrives, it waits till the customer embarks on it; this includes also getting out from the activity location into the street. As soon as the passenger is picked up (time T_i^3), the taxi agent drives along the shortest path to the destination link where the the passenger alights and starts its next activity or leg (time T_i^4). On completion of the the task, the taxi driver agent may be dispatched to serve the next request or may stay and wait. Because of the stochasticity of taxi dispatching and traffic flow, times T_i^1, T_i^2, T_i^3 and T_i^4 are subject to change in the course of simulation.

Depending on the current time τ each request may be in one of four possible states. The sequence of states defining a request's life cycle is as follows:

[8] In the present implementation, no taxi roaming is supported; an idle taxi waits at the destination location of the last served customer or at the home taxi rank.

- *unplanned* – submitted but not scheduled yet
- *planned* – scheduled by the DVRP Optimizer, $\tau \in [T_i^0, T_i^1)$
- *started* – the taxi is on the way to the customer[9] or the customer is already aboard, $\tau \in [T_i^1, T_i^4)$
- *performed* – the customer has been delivered at the destination location, $\tau \geq T_i^4$

3. Taxi dispatching algorithms

3.1. Off-line taxi dispatching problem

In the off-line version of the taxi dispatching problem all data are known a priori and are not subject to any changes.

Let $N = \{1, \ldots, n\}$ be a set of requests. Each request $i \in N$ is submitted at time s_i ($\equiv T_i^0$, see Fig. 2) and has a pickup location p_i and a delivery location d_i. Let $M = \{1, \ldots, m\}$ be a set of vehicles. Each vehicle $k \in M$ is available at its origin location (= depot) o_k from time a_k onwards. It is assumed that a vehicle never returns to its depot and therefore its time window is open-ended. Assuming that τ denotes the instant when the optimization is carried out, the input data must satisfy the following constraints in order to qualify as an off-line problem:

- $s_i \leq \tau, \forall i \in N$ – all the requests are known a priori
- $a_k \geq \tau, \forall k \in M$ – all the vehicles are available from time τ at the earliest.

Concerning the travel times, let $t^{\mathrm{O}}_{ki}(t), k \in M, i \in N$ be a time-dependent travel time from vehicle origin location o_k to customer pickup location p_i, a function of departure time t. The time-dependent travel time from customer delivery location d_i to customer pickup location p_j is defined as $t_{ij}(t), i \in N, j \in N$, where t denotes departure time. Serving request i means spending constant time t^{P} at pickup location p_i (time necessary for the customer to enter a taxi) and then moving from this location to destination location d_i. The travel time of this trip is defined as $t_i(t), i \in N$, a function of departure time t.

Let $V = \{1, \ldots, m, m+1, \ldots, m+n\}$ be the set of vertices, representing both vehicles (vehicles $1, \ldots, m$ as nodes $1, \ldots, m$) and requests (requests $1, \ldots, n$ as nodes $m+1, \ldots, m+n$). The analyzed taxi dispatching problem may be formulated as an off-line assignment problem, i.e. without considering the dynamics of the system, where the goal consists in finding cycles over a set of vertices V. The problem involves two types of variables. The first one are binary variables $x_{uv}, u \in V, v \in V$, indicating whether vertex v is the direct successor of vertex u in one of the cycles. The whole set of binary variables determines the cycles involving all vertices. Each cycle must contain at least one vertex representing a vehicle (vertices $1, \ldots, m$). A cycle is converted into routes by removing arcs that lead to vehicles in a given cycle, thus each 'vehicle' vertex is the beginning of a route.

[9] since no vehicle diversion (reassignment to a different customer) is implemented in the current version, once a vehicle sets out for the customer, the corresponding request is treated as *started*

As a result, each route starts with a 'vehicle' vertex and then goes through a series of 'customer' vertices without returning to the the origin position. Variables $x_{k,m+i}, k \in M, i \in N$ indicate which customer i is the first to be served by vehicle k. Variables $x_{m+i,m+j}, i \in N, j \in N$ specify whether request j is to be served directly after i (by the same vehicle). The second group of variables, $w_i, i \in N$, defines the pickup time of customer i (the moment the pickup starts), which adds the time-dimension into routes.

The taxi dispatching problem may be stated as:

$$\min \sum_{i \in N} (w_i - s_i)^2 \tag{1}$$

subject to

$$\sum_{u \in V} x_{uv} = 1 \qquad \forall v \in V, \tag{2}$$

$$\sum_{v \in V} x_{uv} = 1 \qquad \forall u \in V, \tag{3}$$

$$x_{uv} \in \{0,1\} \qquad \forall u \in V, \forall v \in V, \tag{4}$$

$$(w_i - a_k - t^{\mathrm{O}}_{ki}(a_k)) \cdot x_{k,m+i} = 0 \qquad \forall k \in M, \forall i \in N, \tag{5}$$

$$[w_j - w_i - t^{\mathrm{P}} + t_i(w_i + t^{\mathrm{P}}) - t_{ij}(w_i + t^{\mathrm{P}} + t_i(w_i + t^{\mathrm{P}}))] \cdot x_{m+i,m+j} = 0 \qquad \forall i \in N, \forall j \in N. \tag{6}$$

The objective (1) is to minimize the total sum of squares of waiting times for taxi. The square function is used to favor earlier requests over later ones in order to prevent keeping some customers unserved for longer time, which makes the assignment fairer and closer to the real dispatching strategies. The first three groups of constraints, i.e. (2)–(4), ensure that variables $x_{uv}, u \in V, v \in V$ represent an assignment, i.e. a bijective function. The last two constraint groups are related with the time feasibility of the solution. Constraints (5) ensure that if request i is the first to be served by vehicle k then the vehicle will set out for this request as soon as the vehicle is ready (a_k) and will reach the customer after travel time $t^{\mathrm{O}}_{ki}(a_k)$. Constraints (6) determine pickup times w_i and w_j of two directly subsequent requests i and j ($x_{ij} = 1$). The period between time w_i and w_j consists of the following events (see Fig. 2):

1. arriving at customer i origin location o_i at time w_i ($\equiv T_i^2$)
2. departing from location o_i at time $w_i + t^{\mathrm{P}}$ ($\equiv T_i^3$)

3. arriving at customer i destination location d_i at time $w_i + t^{\mathrm{P}} + t_i(w_i + t^{\mathrm{P}})$ $(\equiv T_i^4)$; the taxi instantly departs towards customer j $(\equiv T_j^1)$

4. arriving at customer j origin location o_j at time $w_j = w_i + t^{\mathrm{P}} + t_i(w_i + t^{\mathrm{P}}) + t_{ij}(w_i + t^{\mathrm{P}} + t_i(w_i + t^{\mathrm{P}}))$ $(\equiv T_j^2)$

Since there are no time windows for requests and the goal is to serve customers as soon as possible, no waiting strategies are necessary and therefore constraints (5)–(6) are equalities. Additionally, these constraints prevents creation of cycles without 'vehicle' vertices.

3.2. On-line taxi dispatching problem

The off-line problem formulated in the previous section finds optimal routes when all information is constant and known a priori and when the system behaves in a deterministic way. However, this is not the case in real life where taxi dispatching is a stochastic dynamic process, where both demand and supply change over time. Firstly, new requests may be submitted at any time, and often, only limited assumptions can be made about future demand (e.g. time and space distributions of new requests). Secondly, since only historical approximates of travel times are known, trips may last longer or shorter than expected due to the changes in traffic flow, interaction with a customer (e.g. during the pickup) and the limited precision of the estimates.

Since demand and supply are subject to random change, the general approach to deal with such a dynamic system should update schedules in response to every change. This can be done by means of re-optimization procedures that consider all the requests (within a given time horizon) or fast heuristics focused on small updates of the existing solution rather than constructing a new one from scratch. Usually, re-optimization procedures give solutions of higher quality compared to the local update heuristics, however, when it comes to the real-world applications, where high responsiveness is crucial, broad re-optimization may be prohibitively time consuming. In this paper, three different strategies (from simpler but faster to more sophisticated but slower) are proposed and their performance is investigated by means of simulation in MATSim.

Another important factor that implies the use of fast heuristic methods is that unlike the theoretical vehicle routing problem where costs (e.g. times or distances) are given in the form of a matrix, dispatching a fleet of taxis in an urban network requires them to be determined on the fly by means of shortest path search procedures. However, this causes the problem to scale not only with the number of customers n and vehicles m, but also with the size of the road network, i.e. the larger the network is, the more computationally expensive the search procedure is; Dijkstra's algorithm, for instance, finds the shortest path in is $O(|N|^2)$. Given that the on-line optimization must be time efficient and give results almost instantly, it cannot search shortest paths extensively, which poses additional challenge.

As it was stated, in the dynamic taxi fleet dispatching problem one has to deal with a dynamic and stochastic demand and supply, and a dynamic optimization strategy reacts to changes, represented as events, that occur over time. In the case of the taxi dispatching problem, there are several different types of events that an optimization strategy may react to. In the most simplistic approach, the optimization algorithm (either a re-optimization or update) is triggered when a new request is submitted (event $\mathbf{E_i^0}$ at time T_i^0, see Fig. 2). This would suffice if the demand for taxi was the only source of stochasticity, which is often valid in theoretical problems but not in the real world. Taking the stochasticity of the order execution process into account, which is justified in typical situations (e.g. stochastic travel times), the dynamic optimization strategy should monitor the order execution statuses and call the optimization algorithm if there are differences between the expected and the actual state. In the simplest case, the algorithm may be triggered just after completion of a request ($\mathbf{E_i^4}$) . But to be more precise and to react as early as possible to unexpected changes, the algorithm could be triggered also after a taxi arrives at the pickup location ($\mathbf{E_i^2}$) or even after departing from the pickup location ($\mathbf{E_i^3}$). In the most extreme version, the movement of each taxi vehicle may be monitored in order to anticipate possible delays or speed-ups.

In the present investigation, it was assumed that the pickup duration is constant (t^{P}) and that vehicles are not monitored on-line (which is typical for taxi services) and therefore no vehicle diversion is possible. In consequence, the implemented strategies react always to events $\mathbf{E_i^0}$ and $\mathbf{E_i^4}$, and additionally, they may react to events $\mathbf{E_i^2}$.

3.3. On-line dispatching strategies

3.3.1. Dynamic to static VRP data conversion Since on-line optimization procedures must be time efficient, algorithms designed for traditional off-line global optimization, performing a broad search of the solution space, are usually not well-suited. Hence fast local optimum search methods, or even just local update methods, have to be used instead. Provided that these methods are fast enough, they produce correct assignments in time that is small or even negligible compared to the dynamics of the ongoing process. This enables them to respond to each event in the interim, before the next one arrives. In case of high frequency of new event arrivals, one may consider grouping them in small packages.

Provided that the dynamic optimization algorithm responds almost instantly to events, we can assume that they operate on static data in a similar way as the static algorithms do. Therefore prior to the optimization, all data describing requests, vehicles, travel times etc. have to be pre-processed to form the input data for the optimization procedure, as if it was a snapshot taken at the present time, τ. This can be done by the following steps:

1. Depending on the dispatching policy, a set of requests N may be created of either *unplanned* and *planned* requests (which means that planned requests can be rescheduled), or only the *unplanned* requests (once a request has been planned it cannot be rescheduled), or any subset of these sets.

2. A set of vehicles M is made up of all or only selected taxicabs. Their parameters have to be updated to be consistent with the current state of the dispatching process. If vehicle k is idle at time τ, it means that the cab is available from time $a_k = \tau$ on, and its origin location o_k is equal to the destination location of the last served request or, if the taxi has not performed any task yet, its depot location. Otherwise, when the taxi is not idle, first the currently being served request must be finished and then, in case of policies that do not reschedule requests, all other already scheduled ones must be served before the vehicle may serve additional requests. Therefore, time availability a_k is set to the expected end time of the last request already scheduled for vehicle k ($a_k \geq \tau$) and origin location o_k is set to the destination location of that request.

3. Set V is defined as a concatenation of vehicle and request indices.

4. Time-dependent travel times $t^{\mathrm{O}}_{ki}(t), k \in M, i \in N$ from vehicle k origin location to customer i pickup location have to be updated. The same must be done with time-dependent travel times $t_{ij}(t), i \in N, j \in N$ from customer i delivery location to customer j pickup location and for time-dependent travel times $t_i(t), i \in N$ between customer i pickup and delivery locations.

3.3.2. Strategies As it was stated earlier, customers perform only immediate taxi calls, and then wait for a taxi to come; minimization of the total waiting time is the optimization goal. Only submitted request are visible, i.e. $\forall i \in N, s_i \leq \tau$. To assure fairness of the dispatching process, all taxi requests $i \in N$ are prioritized according to their submission time s_i, and therefore scheduled based on the *first-come, first-served* (FCFS) policy. Three different strategies, namely no-scheduling, one-time scheduling and re-scheduling, were implemented.

No-scheduling strategy (NOS) In this strategy, instead of producing a valid schedule, the optimization algorithm assigns the highest priority task to the *nearest* (according to a given measure; see 3.3.3) idle taxi, while the rest of queued request are pending assignment. In other words, set N consists only of the first request from the FIFO queue and set M consists only of vehicles that are idle. The strategy reacts to the following events:

- $\mathbf{E^0_i}$ – submission of request i – the *nearest* vehicle among the idle ones is dispatched to this request; if no vehicle is available at that time, the request is queued in the FIFO queue
- $\mathbf{E^4_i}$ – completion of request i – the vehicle that served this request is dispatched to the first request in the FIFO queue; otherwise, the vehicle becomes idle

This quite simplistic strategy imitates the way orders are assigned to taxis in a typical taxi company. The advantage of this approach is its low demand for computational power – it just requires executing a shortest path search algorithm and choosing the closest idle taxi. Moreover, this strategy does not require travel times to be known since it does not build schedules; one can apply any distance measure to find the nearest idle taxi, which is another good point. The drawback is that its performance deteriorates as the number of idle taxis decreases. In such situations, a taxi located on the opposite side of a city may be dispatched to serve a request, because all the taxis located near to the customer are busy.

One-time-scheduling strategy (OTS) This strategy updates the existing schedule by appending a new request to the list of requests already assigned to the nearest vehicle. Similarly to the first strategy, set N is also of size 1. The difference is that this strategy considers all vehicles (both idle or busy) and to do that it has to build schedules, which in turn, requires the knowledge of travel times. After executing the strategy, all requests are planned, hence the schedule is valid. Since the process of taxi dispatching is monitored, when any divergence between the plan and its execution is detected, the schedule timeline is updated, however requests are not rescheduled/reassigned. The strategy acts in the following way:

- $\mathbf{E_i^0}$ – submission of request i – the request is appended to the schedule of the *nearest* vehicle (can be idle or busy)
- $\mathbf{E_i^2}$ and $\mathbf{E_i^4}$ – arrival at the pickup and delivery location of request i – if the vehicle serving request i is ahead of/behind time ($T_i^2 \neq \tau$ or $T_i^4 \neq \tau$, respectively), the schedule timing is updated (i.e. values T_j^1, T_j^2, T_j^3 and T_j^4 representing all the subsequent events); the assignments remain unchanged

This strategy considers all the available vehicles, not only the idle ones, which broadens the choice of taxis and thus increases the chances of finding a better assignment compared to the first strategy. However, the weak point here is the permanence of assignments. If a vehicle is seriously delayed at a given point in time, none of the awaiting orders can be re-assigned to another vehicle. In such situations only the timing information is updated, which may prevent assigning new orders to this delayed vehicle. As a result, it may happen that this strategy would perform poorly, however it most cases, it is expected to outperform the first one.

Re-scheduling strategy (RES) This strategy is an enhanced version of the previous one. The difference is that schedules are always re-computed in response to any registered delays or speed-ups, and therefore, requests may be re-assigned to another taxi if that other taxi is currently the *nearest* one. The strategy is defined as follows:

- $\mathbf{E_i^0}$ – submission of request i – the request is appended to the schedule of the *nearest* vehicle (can be idle or busy)

• $\mathbf{E_i^2}$ and $\mathbf{E_i^4}$ – arrival at the pickup and delivery location of request i – if the vehicle serving request i is ahead of/behind time ($T_i^2 \neq \tau$ or $T_i^4 \neq \tau$, respectively), the schedule is re-calculated; the assignments are subject to change

In this third strategy the scheduling algorithm is run in two modes. On the arrival of a new request, the scheduling works in exactly the same way as in case the OTS strategy. This means that there is only one request to schedule ($|N| = 1$) and the request is assigned to one of the taxis as the last request in the schedule. However, the operation of the algorithm is different in the second mode when responding to observed delays or speed-up. In this case the schedule is cleared and all requests are re-scheduled. Theoretically, $|N|$ may be considered to be equal to the number of the received orders that remain unserved.[10] In practice, however, $|N| = 1$ as all these requests are processed separately one-by-one according the FCFS policy, as if the were re-submitted in the same order as they originally arrived. The RES strategy overcomes the weaknesses of the previous ones – it creates a schedule (unlike the NOS strategy) and the request-to-taxi assignments are not permanent (in contrast to the OTS strategy). Thus it seems most efficient but at the same time most computationally expensive.

Note that being attached at the end of the schedule does not mean that the customer will be served later than all the previously scheduled vehicles. The FCFS property holds only within a single taxi, i.e. the order in which customers are served by a given taxi is equal to the order in which they were submitted. However, this cannot be assured for requests assigned to different vehicles. Thus FCFS should stand for first-come-first-scheduled rather than first-come-first-served.

3.3.3. Distance measures There are many possible ways of measuring distances between vehicles and customers that can be applied for finding the *nearest* vehicle:

• *Straight line* (SL) — offers the lowest precision, however, it is very popular with taxi companies; it is also fast since it does not require running the shortest path search[11]

• *Travel distance* (TD) — the shortest-distance path; if all links are permanently open (available), the shortest-distance path is constant over a day

• *Fixed travel time* (FT) — the shortest-time path based on fixed speed data, e.g. free speed travel times or 24-hour average link travel times (the latter option was used for the experiments); if all links are permanently open (available), the path is constant over a day

• *Dynamic travel time* (DT) — the shortest-time path computed for a given moment in time; based on, for example, the 15-minute link travel time statistics to reflect the daily changes in traffic, which guarantees the highest precision of finding the vehicle closest in time.

[10] When a taxi heads towards a given request, this request is considered as *being served*.

[11] The straight-line distance is used only for choosing a taxi, then the chosen taxi is routed along the shortest-distance path.

Since the MATSim simulation uses the queue-based traffic flow model, locations of vehicles and request are known within the accuracy of a single link. All paths are calculated according to the shortest path search algorithms available in MATSim, as described in Sec. 2.4, whereas for the straight-line distance the middle points of links are taken into account. As the first two measures do not give any information on the travel time, they cannot be used for building schedules and hence are applicable only to the first strategy (Tab. 1).

Table 1 Distance measure applicability

Strategy	Straight line	Travel distance	Travel time (fixed)	Travel time (dynamic)
No-scheduling	+	+	+	+
One-time-scheduling	−	−	+	+
Re-scheduling	−	−	+	+

4. Simulation scenario and performance metrics

The computational analysis was carried out for Mielec, a city in south-eastern Poland, with a population of over 60'000 inhabitants. The scenario was derived from a macroscopic model of private transport in Mielec for the afternoon 1-hour peak, created originally in PTV VISUM,[12] and used as a small-size test instance in several studies, (e.g. Piatkowski and Maciejewski 2013 (in press). The network model consists of 214 nodes and over 610 links of three types, namely main, bulk and local roads. Besides the urban network, external roads were modeled to allow for inbound, outbound and transit traffic. The whole study area was divided into 13 zones, each one assumed to have homogeneous land use. Nine of them represent city districts, while the rest – external areas (sources and destinations of the non-intra-urban traffic).

In the original model, the transport demand was represented as an OD matrix, which contained over 8'800 trips within the 1-hour afternoon peak. Based on that OD matrix, 14 OD matrices were artificially generated to cover the period between 6:00 am and 8:00 pm. Because of the lack of data, it was assumed that the morning traffic is the opposite of the afternoon traffic and that the non-peak-hour OD matrices are a proportion of the closer peak hour OD matrix. In the end, the generated demand consists of over 42'000 private transport trips that represent a hypothetical case of day traffic, including both the morning and afternoon rush-hour traffic.

The traffic simulation was carried out in MATSim using the standard iterative approach (see Sec. 2.1). The simulation process converged within 20 iterations. Fig. 3 shows a snapshot of traffic flow simulation in the 20th iteration (dots represent agents; dark grey dots indicate agents stuck

[12] `http://vision-traffic.ptvgroup.com/en-uk/products/ptv-visum/`

in congestion.) Fig. 4 shows the resulting daily pattern. From the graph together with knowledge about the network size (no long trips) one can conjecture that there is some congestion, but it is not very strong. This is confirmed by the snapshot (Fig. 4). Besides having a somewhat realistic demand structure, the scenario also demonstrates that our approach is able to read and process realistic network and demand files.

Figure 3 Traffic flow simulation in Mielec at 5:00 pm (iteration 20)

The testing scenarios consisted in running a special (21st) iteration of MATSim simulation, based on the results of the last (20th) regular iteration (agents' plans and link travel time statistics). This additional iteration of simulation begins with a specific re-planning phase, where all plans remain the same with the exception for the the transport mode – the mode of a given fraction of trips is changed from *car* to *taxi*. The probability of shifting to the taxi mode is uniform, regardless of the time, location or purpose of the trip. Next, the traffic simulation phase takes place; the taxi demand is served according to the principles described in Sec. 2.5. In the last phase of the iteration, besides

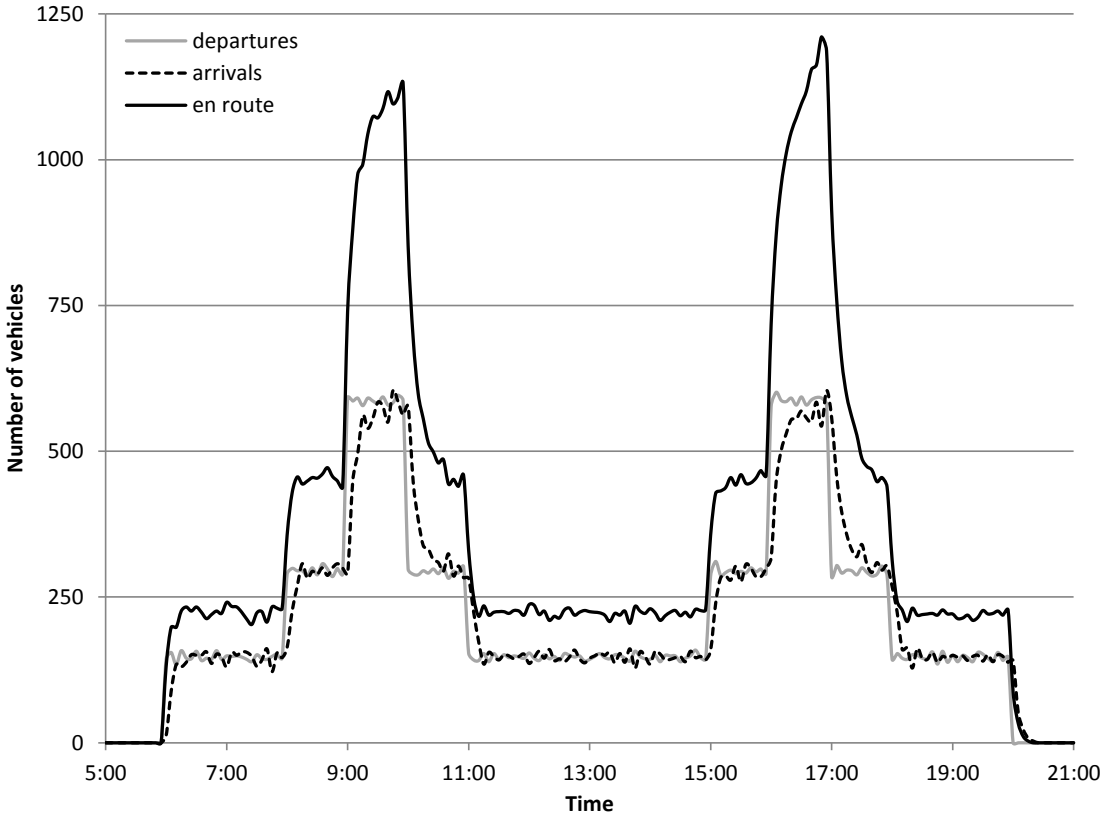

Figure 4 **Vehicle departures and arrivals (per 5 minutes), and number of vehicles en route**

the typical agent plan scoring, a detailed evaluation of the taxi service performance is carried out. This evaluation is used later for the comparison purposes.

The performance of the proposed optimization strategies was tested against different variants (called *experiments*) of the Mielec scenario. In order to take into consideration different demand-supply relations, the taxi demand was modeled as 3, 5 or 7% of the private intra-urban transport demand (917, 1528 and 2175 orders, respectively), and the taxi fleet size varied from 50 to 100 taxis. Tab. 2 shows demand-to-supply ratios, defined as n/m, in different simulation experiments. Since the fleet size was constant over the entire simulation and the taxi demand peak coincided with the private transport demand peak, the rush hours were the most challenging for taxi dispatching.

Table 2 **The demand-to-supply ratio in different simulation experiments**

Fleet size	Taxi demand [%] 3%	5%	7%
50	18.34	30.56	43.5
100	9.17	15.28	21.75

Different measures of performance were examined during the simulation studies. From the taxi customers' point of view the most important indicators are:

- average passenger waiting time, $T_{\mathrm{W}} = \sum_{i \in N} \frac{T_i^2 - T_i^0}{n}$
- maximum passenger waiting time, $T_{\mathrm{W}}^{\max} = \max_{i \in N}(T_i^2 - T_i^0)$
- average delivery (i.e. occupied) trip time, $T_{\mathrm{D}} = \sum_{i \in N} \frac{T_i^4 - T_i^3}{n}$
- average passenger-waiting-time-to-taxi-leg-time ratio, $R_{\mathrm{W}} = \sum_{i \in N} \frac{T_i^2 - T_i^0}{(T_i^4 - T_i^0)n}$

On the other hand, taxi service provider's performance may be assessed based on the following statistics:

- average pickup (i.e. empty) trip time, $T_{\mathrm{P}} = \sum_{i \in N} \frac{T_i^2 - T_i^1}{n}$
- average pickup-to-all-trips-time ratio, $R_{\mathrm{P}} = \sum_{i \in N} \frac{T_i^2 - T_i^1}{((T_i^2 - T_i^1) + (T_i^4 - T_i^3))n}$
- average non-idle-to-total-time ratio (14-hour simulation), $R_{\mathrm{NI}} = \sum_{i \in N} \frac{T_i^4 - T_i^1}{m \cdot 14 \cdot 3600[\mathrm{s}]}$

In all cases, the lower the indicator values the better. The customers' and company's performance measures are often conflicting. For instance, a taxi company might be interested in deferring the service of an order until there are taxis close to the pickup location; this, however, results in higher passenger waiting times, and ultimately, may end in shifting to a different taxi company or even transport mode. For the evaluation purposes, it was assumed that T_{W} is the most important indicator of quality of service, whereas T_{P} constitutes the major contribution to the company's operating costs, and R_{NI} reflects the overall efficiency of resource (taxicab) utilization. To provide more accurate statistics, all the indicators are averages – each experiment was carried out 20 times for all possible *strategy–distance* pairs, each time with a different random number sequence.

5. Simulation results

Tab. 3 presents the results obtained for low n/m levels (i.e. experiment *3%:100*). In all cases, taxis are idle for almost 90% of time ($1 - R_{\mathrm{NI}}$). Except for the traffic peak period, the OTS and RES strategies behave almost as the NOS strategy as only few taxicabs are busy. Using the dynamic travel time estimates, on average, all strategies (NOS, OTS, RES) perform similarly with a slight advantage of NOS. The main difference lies in $T_{\mathrm{W}}^{\max}$, which points out that the prioritization of orders in OTS is not as effective as in the other strategies. This is caused by the fact that OTS cannot reassign already scheduled orders. In consequence, when compared to RES, it is more likely that some old requests, assigned to delayed taxis, are served later than newer requests.

The use of less accurate (i.e. fixed) travel times deteriorates the performance of both scheduling strategies (i.e. OTS and RES), especially in terms of T_W. Analysing T_{P} and T_{D} one can notice an improvement in the quality of routes when the dynamic travel times are used, while the straight-line distance measure performs worst.

Table 3 Results for experiment 3%:100

Strategy	Distance	T_{W} [h:mm:ss]	$T_{\mathrm{W}}^{\max}$ [h:mm:ss]	T_{D} [h:mm:ss]	R_{W} [%]	T_{P} [h:mm:ss]	R_{P} [%]	R_{NI} [%]
NOS	SL	0:01:46	0:11:56	0:06:33	17.1%	0:01:46	21.2%	11.3%
NOS	TD	0:01:34	0:09:58	0:06:33	15.5%	0:01:34	19.3%	11.1%
NOS	FT	0:01:34	0:13:34	0:06:30	15.6%	0:01:34	19.5%	11.0%
NOS	DT	0:01:30	0:09:31	0:06:19	15.3%	0:01:30	19.2%	10.7%
OTS	FT	0:01:45	0:15:54	0:06:28	17.2%	0:01:34	19.6%	11.0%
OTS	DT	0:01:33	0:15:03	0:06:19	15.8%	0:01:29	19.1%	10.7%
RES	FT	0:01:44	0:14:43	0:06:29	17.0%	0:01:34	19.5%	11.0%
RES	DT	0:01:33	0:10:48	0:06:19	15.8%	0:01:30	19.1%	10.7%

To conclude, the differences between most of the results are not significant. One can notice that in the case of low demand-to-supply ratio, the use of simple strategies (like NOS) is justified. Also, the choice of distance measure has a bigger impact on the outcomes than the choice of the strategy.

Significantly different results were obtained for the medium n/m experiments, like *7%:100* (Tab. 4). With higher demand the number of idle taxis is significantly reduced to slightly more than 70% on average (and much lower values during the peaks). In this experiment, the advantage of the NOS strategy over the others, in terms of quality of service (T_{W} and R_{W}), is clearly visible, which may be surprising at first. On the other hand, all cost indicators ($T_{\mathrm{P}}, R_{\mathrm{P}}$ and R_{NI}) are better when using the letter strategies. This is caused by the fact that the arrival time estimator for on-going trips is biased. The end time of a *started* pickup trip, for instance, is estimated as:

$$\widehat{T}_i^4(\tau) = max(T_i^3 + t_i(T_i^3), \tau) \qquad \forall i \in N, \forall \tau \in [T_i^3, T_i^4). \tag{7}$$

Suppose that the trip starts at $T_i^3 = 0$ and is expected to end at $\widehat{T}_i^4(0) = 100$. If the taxi is still en route at time $\tau = 98$, it is more likely that the trip will end later than at 100, whereas the estimator will still predict the same value, i.e. $\widehat{T}_i^4(98) = 100$. Moreover, if the taxi is still en route at time $\tau = 105$, the estimator will predict the trip to end at this very moment, i.e. $\widehat{T}_i^4(105) = 105$, which is actually very unlikely. As one can see, $\widehat{T}_i^4(\tau)$ for $\tau \in (T_i^3, T_i^4)$ is underestimated, and hence, the schedule timings are are usually too optimistic. As a result, the OTS and RES strategies give preference to non-idle vehicles, which results in shorter pickup trips, but at the cost of longer taxi awaiting. It is also worth pointing out that like in the first experiment, the RES strategy gives 30% lower $T_{\mathrm{W}}^{\max}$, compared to OTS, when the dynamic travel times are used.

Analysing the influence of the distance measures used, again one can notice the dynamic travel times produce the best performance. The use of the averaged travel times is even less effective compared to calculating shortest-distance paths.

Similar relations between individual *strategy–distance* cases were also observed for other experiments, like *3%:50* (Tab. 5). The main difference between them lies in R_{NI}, as there are 21.8 requests

Table 4 Results for experiment 7%:100

Strategy	Distance	T_W [h:mm:ss]	T_W^{max} [h:mm:ss]	T_D [h:mm:ss]	R_W [%]	T_P [h:mm:ss]	R_P [%]	R_{NI} [%]
NOS	SL	0:02:09	0:18:35	0:06:39	19.9%	0:02:09	24.4%	28.0%
NOS	TD	0:02:01	0:15:13	0:06:39	18.9%	0:02:01	23.2%	27.6%
NOS	FT	0:02:09	0:18:56	0:06:43	19.8%	0:02:09	24.3%	28.1%
NOS	DT	0:01:57	0:14:03	0:06:30	18.7%	0:01:57	23.1%	27.0%
OTS	FT	0:02:59	0:27:02	0:06:39	25.6%	0:01:52	21.9%	27.2%
OTS	DT	0:02:21	0:21:27	0:06:27	21.8%	0:01:47	21.7%	26.5%
RES	FT	0:03:00	0:27:39	0:06:38	25.8%	0:01:53	22.1%	27.3%
RES	DT	0:02:21	0:14:12	0:06:28	21.8%	0:01:47	21.7%	26.5%

per taxi in the former experiment and only 18.3 in the latter. However, the general conclusion is that with the growing demand-to-supply ratio, the choice of optimization strategy is decisive. The use of the biased estimator for the arrival time of on-going trips significantly deteriorates the the performance of more sophisticated strategies. When the estimations are based on less inaccurate (averaged) travel times, the outcomes even worsen.

Table 5 Results for experiment 3%:50

Strategy	Distance	T_W [h:mm:ss]	T_W^{max} [h:mm:ss]	T_D [h:mm:ss]	R_W [%]	T_P [h:mm:ss]	R_P [%]	R_{NI} [%]
NOS	SL	0:02:17	0:13:52	0:06:34	21.0%	0:02:17	25.8%	23.7%
NOS	TD	0:02:05	0:11:35	0:06:33	19.6%	0:02:05	24.1%	23.2%
NOS	FT	0:02:14	0:15:05	0:06:31	20.8%	0:02:14	25.5%	23.5%
NOS	DT	0:02:01	0:10:25	0:06:19	19.5%	0:02:01	24.2%	22.6%
OTS	FT	0:02:36	0:20:38	0:06:29	23.4%	0:02:09	24.9%	23.2%
OTS	DT	0:02:06	0:20:26	0:06:18	20.1%	0:01:55	23.3%	22.3%
RES	FT	0:02:37	0:18:00	0:06:29	23.6%	0:02:09	25.0%	23.2%
RES	DT	0:02:06	0:12:43	0:06:18	20.2%	0:01:55	23.4%	22.3%

The computations run for the high demand-to-supply ratios (*7%:50*; Tab. 6) illustrate the operation of the designed strategies under heavy load, when R_{NI} exceeds 60% and there is a shortage of taxis during the morning and afternoon rush hours; $T_P \ll T_W$ proves that there are longer periods where no taxi is dispatched for awaiting customers. In such circumstances, the NOS strategy behaves randomly – when no vehicle has been idle, it assigns an order to the first taxi that completes its tasks. A more detailed inspection of T_P and T_W shows that the scheduling strategies dispatches a taxi 8–9 minutes after the request ($T_W - T_P$), while for the NOS it takes around 15–16 minutes, on average. This proves that far from optimal, often random, dispatching decisions increase the overload of taxis (larger T_P), which makes the shortage of taxis even more acute.

Interestingly, T_D is the only indicator that improved compared to the previous experiments, and is particularly smaller for the NOS strategy. This is, however, a side effect of much larger T_W, i.e.

Table 6 Results for experiment 7%:50

Strategy	Distance	T_{W} [h:mm:ss]	$T_{\mathrm{W}}^{\max}$ [h:mm:ss]	T_{D} [h:mm:ss]	R_{W} [%]	T_{P} [h:mm:ss]	R_{P} [%]	R_{NI} [%]
NOS	SL	0:20:03	1:05:42	0:06:06	71.2%	0:04:06	40.1%	63.2%
NOS	TD	0:19:44	1:06:27	0:06:07	70.9%	0:04:03	39.8%	62.9%
NOS	FT	0:20:16	1:06:02	0:06:07	71.4%	0:04:04	40.0%	63.1%
NOS	DT	0:18:31	1:02:42	0:05:59	69.9%	0:03:51	39.1%	61.3%
OTS	FT	0:11:30	0:51:22	0:06:15	58.2%	0:02:26	28.0%	55.3%
OTS	DT	0:10:13	0:45:18	0:06:06	55.8%	0:02:22	27.9%	54.1%
RES	FT	0:11:24	0:48:15	0:06:15	58.0%	0:02:25	27.9%	55.3%
RES	DT	0:10:10	0:43:51	0:06:06	55.7%	0:02:21	27.8%	54.1%

the passenger pickups are delayed, sometimes up to an hour ($T_{\mathrm{W}}^{\max}$), and therefore, the taxi peak is shifted beyond the traffic rush hour, which results in shorter travel times. Another observation is that unlike the previous experiments, $T_{\mathrm{W}}^{\max}$ is much bigger for the NOS strategy, and the difference between the OTS and RES strategies is reduced. Moreover, the relative differences between T_{W} and $T_{\mathrm{W}}^{\max}$ are smaller than in the previous experiments. This can be accounted for by the relatively short period of the highest traffic (only one hour both in the morning and afternoon) – after one hour traffic decreases and it becomes easier to reach the waiting passengers.

A comparison of different distance measures shows similar relations as in experiment *7%:100*, i.e. the dynamic travel times perform best and the shortest-distance measure is slightly better than the remaining two.

To sum up, the NOS strategy does not anticipate the future state of the system and makes very myopic decisions that are good only at the current time τ. In the overloaded taxi dispatching system this deteriorates its performance in the long term.

Relations between the demand-to-supply ratio and selected indicators are presented in Figs. 5 to 7. In order to provide a clear picture, overlapping curves were aggregated according to the following scheme:

- *NOS-non-DT* – the NOS strategy and all distance measures except for the dynamic times (DT)
- *NOS-DT* – the NOS strategy and the DT distance measure
- *w/S-FT* – both schedule-oriented strategies (i.e. OTS and RES) and the fixed time (FT) distance measure
- *w/S-DT* – both schedule-oriented strategies and the DT measure

Furthermore, separate curves were plotted for the taxi fleet size of 50 (n/m equals 18.34, 30.56 and 43.5) and of 100 (where n/m equals 9.17, 15.28 and 21.75). Looking at Fig. 5, one can notice the influence of knowing the accurate travel times on the quality of service at low n/m (*NOS-DT* and *w/S-DT* vs. *NOS-non-DT* and *w/S-DT*). At medium range of the ratio of n/m (particularly

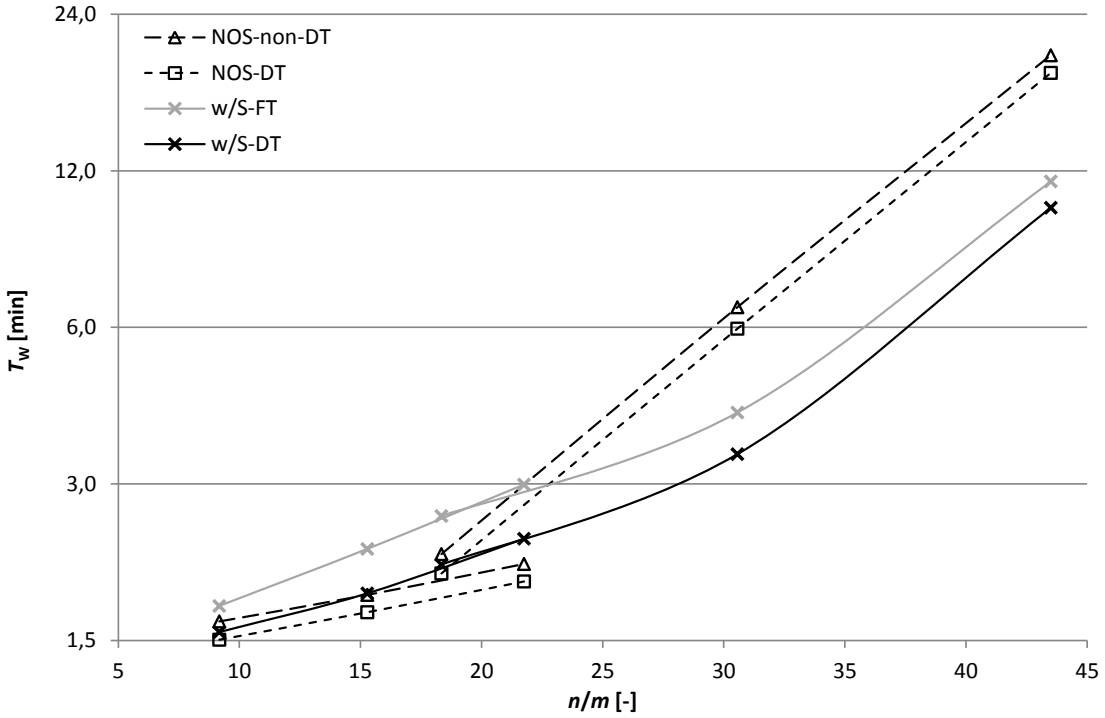

Figure 5 **Average passenger waiting time T_{W} at different demand-supply ratios (logarithmic scale)**

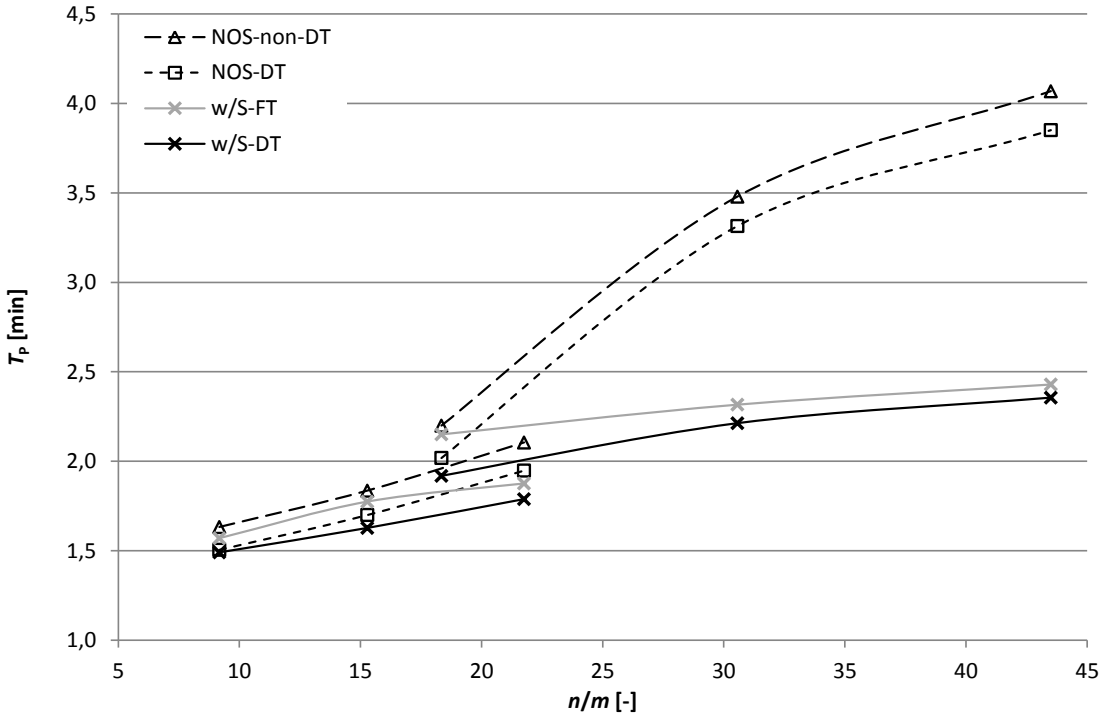

Figure 6 **Average pickup trip time T_{P} at different demand-supply ratios**

in experiment *7%:100*), due to the use of the biased arrival time estimator for on-going trips, both scheduling algorithms result in higher T_{W}; especially the use of the imprecise travel time estimates

(FT) seriously deteriorates scheduling. However, as n/m increases further and the taxi system gets overloaded, the situations turns exactly the opposite. In such cases, the use of scheduling (regardless of the distance measure chosen) is superior to the NOS approach. The logarithmic plot is, in fact, somewhat hiding the success of the scheduling strategies: For the highest load, the advantage is nearly a factor of two, reducing the average waiting times from about 18.5 minutes to about 10.2 minutes (for DT).

Looking from the taxi company's perspective, a different picture emerges. Although the main motivation of using the scheduling strategies is reduction of passenger waiting times, they have a beneficial side effect of reducing the operating costs. Both with DT or FT, both strategies yield lower T_{P} (Fig. 6), and the difference widens with the increasing ratio of n/m. This can be explained as follows. If a non-idle vehicle is closer (in time) to a given customer by an amount of time $t > 0$ than an idle vehicle, the pickup trip duration of the non-idle is shorter at least by t.

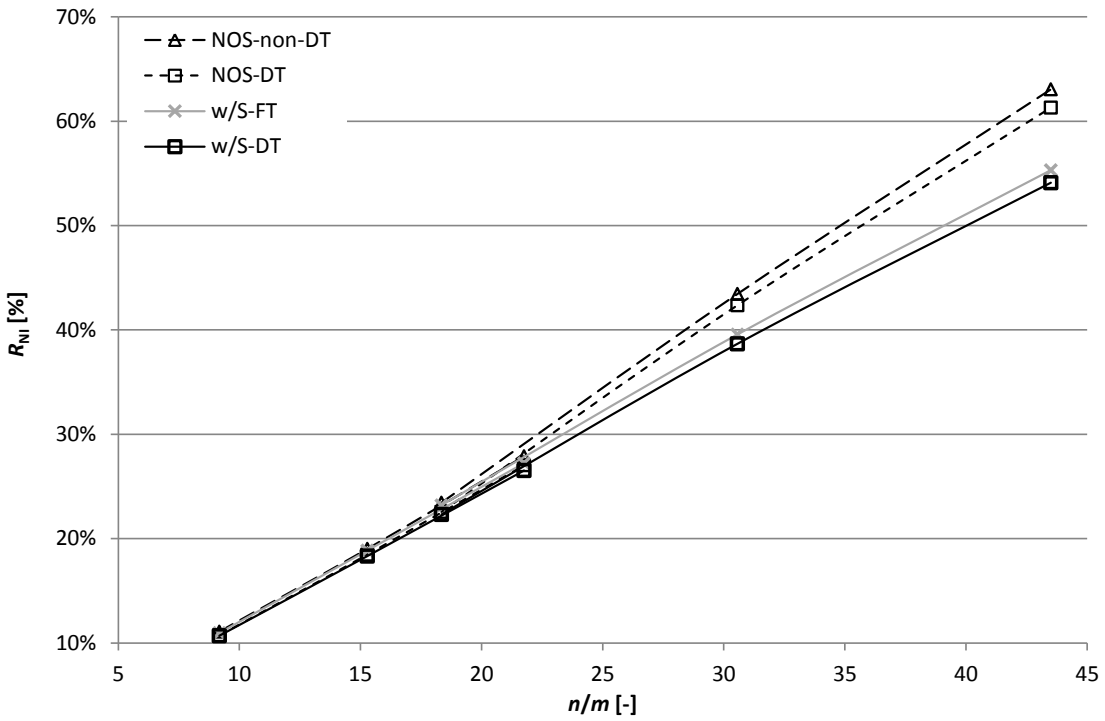

Figure 7 **Average non-idle-to-total-time ratio R_{NI} at different demand-supply ratios**

With lower T_{P} and comparable T_{D} (as the origins and destinations of taxi trips are fixed), the OTS and RES strategies achieve lower values of R_{NI} (Fig. 7), which indicates that, on average, they have more idle taxis on their disposal. Additionally, both strategies take busy cabs into consideration, so there is a wider choice of vehicles for dispatching to new customers, and thus the dearth of taxis during the high peak is shorter and less acute.

6. Discussion

Pickup trip times The average pickup trip time, T_{P}, essentially depends on the average distance of the taxicab when it is assigned to the customer. If we assume that m_{idle} taxis are distributed homogeneously over the service area A, then each idle taxi serves an area $a = A/m_{\mathrm{idle}}$. Assuming homogeneous conditions, the average pickup distance, and thus the average pickup trip time, is proportional to the square root of this:

$$T_{\mathrm{P}} \sim \left(\frac{A}{m_{\mathrm{idle}}}\right)^{1/2} . \tag{8}$$

This leads to the following consequences:

- Fig. 6 shows that having 100 taxicabs for 7% of the population leads to better pickup trip times than 50 taxicabs for 3% of the population in spite of the fact that the supply-to-demand ratio in the first case is actually worse. This is due to the fact that, with 100 taxis, the chance that an idle taxi is close by is larger than with 50 taxis. This effect is, albeit to a lesser extent, also reflected in the passenger waiting times (Fig. 5).
- For high loads, the pickup trip time eventually levels out.

Without scheduling (= NOS), the next available taxi is matched with the first customer in the queue, irrespective of the distance, and because of $m_{\mathrm{idle}} = 1$, Eq. (8) immediately leads to $T_{\mathrm{P}} \sim A^{1/2}$: The average pickup trip time levels out at the average time necessary to connect two randomly selected points in the city.

With scheduling: Let us define $\ell(k), \forall k \in K$ as the last customer assigned to taxi k, according to the current schedule. Therefore, taxi k has final destination $d_{\ell(k)}$ at the current end of its schedule, and will reach that destination at time $T^4_{\ell(k)}$. The taxi that will be assigned the next request j, at location p_j, is the one that minimizes $t_{\ell(k),j} + T^4_{\ell(k)}$. Assuming that the $T^4_{\ell(k)}$ are independently and identically distributed with a mean μ, this becomes

$$\min_{k \in M} t_{\ell(k),j} + \mu + \epsilon_{\ell(k)} ,$$

where the constant μ is irrelevant for the optimization, and the expectation value of $\epsilon_{\ell(k)}$ is zero.[13] The expectation value $E(\min_{k \in M} t_{\ell(k),j} + \epsilon_{\ell(k)})$ is clearly smaller than $E(t_{\ell(k),j})$, and since the latter is the average pickup time without scheduling, the expectation value for T_{P} *with* scheduling is smaller than *without* scheduling.

Thus, the simulations confirm the tendencies of Eq. (8), without providing enough material to accept or reject the specific mathematical form.

[13] The problem is curiously similar to location choice in random utility models (Ben-Akiva and Lerman 1985, Horni et al. 2012), where person i selects a location j that maximizes $-t_{ij} + \epsilon_j$ and thus minimizes $t_{ij} - \epsilon_j$.

These considerations also clarify that under high loads another strategy is possible: Every taxi that becomes available just picks the closest waiting customer, irrespective of the customer's position in the queue. This would make the average T_P ever smaller for ever larger loads, and would thus increase the capacity (rate with which passengers are served) of the system. It would, however, be considered unfair. Since, however, as we will also reconfirm below, a load above the capacity of the system has drastic consequences for the waiting time, it may make sense to consider such strategies anyway, e.g. for exceptional situations.

Waiting time Fig. 5 implies that there is a low and a high load regime.

In the **high load regime**, the demand rate d is larger than the supply rate s for some periods. Ignoring the spatial aspects of the problem, the expected total overload waiting time depends both on the extent and the duration of the overload:

$$T_\mathrm{W}^\mathrm{overload} \sim \int_{t_0}^{t_1} (d(t) - s)\, dt \ , \tag{9}$$

where t_0 is the time when the overload starts, and t_1 is the time when the backlog has been served.

Let us look, inspired by Fig. 4, at a queueing system with large demand rate $d > s$ for some period τ, and some smaller demand rate $d' < s$ for all other times. One can state the following:

- At time τ, the queued excess demand is $(d-s)\cdot\tau$.
- After the high load period is over, the excess demand is dissolved with rate $s-d'$. The necessary time for this is

$$\tau' = \frac{(d-s)\cdot\tau}{s-d'} \ . \tag{10}$$

- Inserting these considerations into Eq. (9) first leads to

$$T_\mathrm{W}^\mathrm{overload} = \frac{1}{2}\left[(d-s)\cdot\tau\right](\tau+\tau') \ ,$$

which is due to the linear nature both of the queue build-up and the queue dissolution. Inserting Eq. (10) then leads to

$$= \frac{1}{2}\left[(d-s)\cdot\tau\right]\left(\tau + \frac{(d-s)\cdot\tau}{s-d'}\right) \sim d^2 \ .$$

- Thus, the average waiting time *per customer* would scale, to leading order, as $\sim d$.

This would imply, in the overload regime, that average waiting times grow linearly in the demand. Fig. 5, however, rather implies that they grow exponentially in the demand. Fig. 8 provides an even cleaner indication of an exponential law: The figure shows the average waiting minus the average pickup times and thus, in fact, that part of the waiting time that is due to the overload. That is, the simulations imply that there is something going on that is more damaging to performance than the simple model from above implies. At this point, we are unable to say what this could be, but it confirms the necessity of real world simulations.

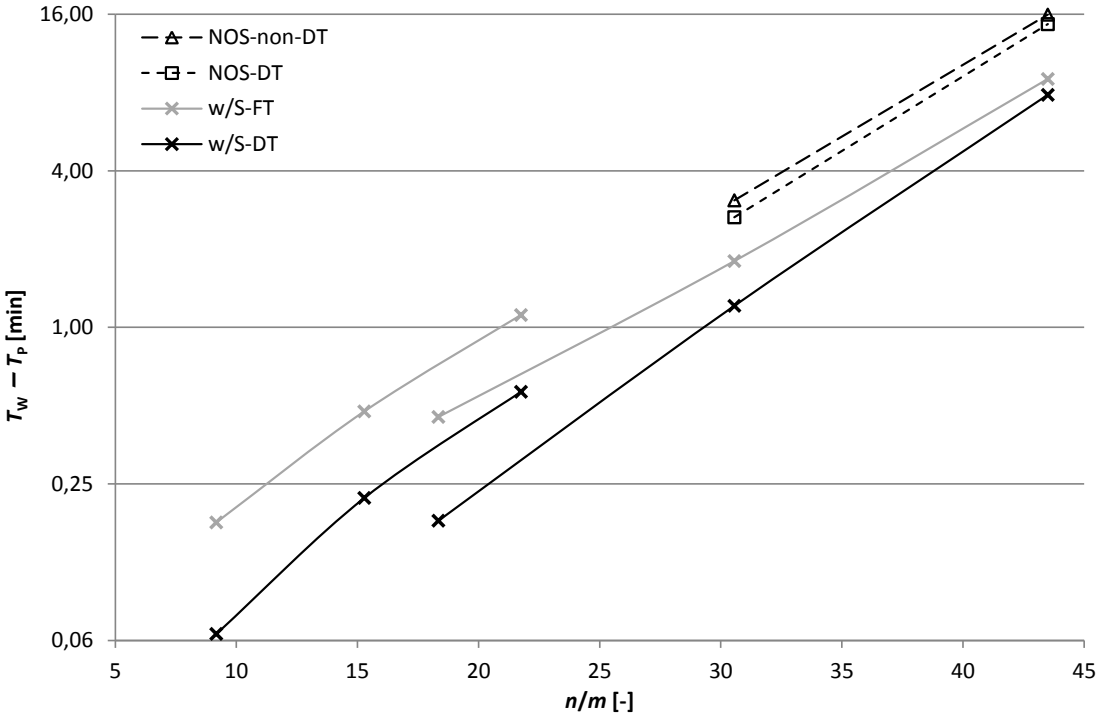

Figure 8 **Average dispatching delay time, $T_{\mathrm{W}} - T_{\mathrm{P}}$, at different demand-supply ratios (logarithmic scale)**

The **low load regime** can be understood as a queueing system. In a queueing system, the average waiting time depends on how close the system is to capacity. If d is the demand rate and s is the supply rate, then the average waiting time behaves as

$$T_{\mathrm{W}} \sim \frac{d/s}{s-d} \ .$$

That is, average waiting times increase with increasing d. The divergence for $d \to s$ is, however, not realized in our system, because the high load regime is only active for limited amounts of time, and thus the argument explained earlier leading to $T_{\mathrm{W}} \sim d$ will hold.

Biased estimation The problem of the biased travel time estimation will be addressed in future work by adding vehicle monitoring. Since MATSim uses the queue model, the current position of a taxi can be known within the accuracy of a single link. Such precision is sufficient for more accurate prediction of the arrival times, but also for diverting taxis from their current destination. With those enhancements, the RES strategy should outperform the others regardless of the demand-supply relation.

Knowledge of the delivery location Each taxi request specifies both the pickup and delivery locations. In reality, the destinations are not always available to the dispatcher. Both scheduling strategies, i.e. OTS and RES, could be adapted to this restriction by shortening the time horizon of a vehicle's schedule to the next pickup event; the time horizon is shifted forward after a passenger

enters the vehicle and the destination becomes known. As a result, new requests that cannot be scheduled are queued. This modification brings the scheduling strategies closer to the NOS strategy that does not use the delivery location information and queues new requests when no vehicle is idle. One may expect that this modification would lead to performance losses.

Threshold for re-scheduling Another issue related to the scheduling strategies, particularly to RES, is finding the threshold value of a vehicle's deviation from its schedule for triggering the rescheduling. At present, even a delay or speedup of 1 second invokes the re-optimization. In real life, a 30-second delay may be a good cause for updating the timing of schedules, but rebuilding them from scratch even with a one second deviation seems unjustified.

Call-ahead requests Currently, the integrated system supports only immediate request. Although the DVRP Optimizer can handle advance requests, the present implementation of MATSim does not provide functionality for calling ahead – taxis are called after finishing an activity. From the passenger's perspective, the lack of advance requests results in longer waiting times. However, on typical days the waiting times are relatively small, while during high-load periods, taxi pre-ordering is often disabled by the operators. Nevertheless, the call-ahead functionality will be addressed in the future.

7. Conclusions

The paper focuses on the two following issues. Firstly, it presents the integration of a microscopic, behavior-based traffic simulation (MATSim) and a dynamic vehicle routing optimizer (DVRP Optimizer). The integrated system serves as a tool for detailed simulation and evaluation of dynamic taxi services as one of several different transport means, all embedded into a realistic environment. The main part of the paper thus deals with the problem of designing on-line taxi dispatching strategies. Three strategies, combined with various distance measures, are implemented within the integrated system and evaluated on the scenario of the Polish city of Mielec. The three strategies are:

- no-schedule (NOS): every new customer is assigned to the nearest idle taxi. If no idle taxi is available, the next idle taxi is assigned to it (no matter of its position).
- one-time schedule (OTS): every new customer is assigned to the taxi which is predicted to be first with the customer after all already assigned trips are served.
- re-schedule (RES): In addition to OTS, the schedule is re-computed every time a passenger is dropped off when the drop-off time is different from the prediction

The results reveal interesting relationships between the level of the taxi demand-to-supply ratio and the performance of the strategies. At a low ratio, the NOS strategy performs minimally better than the other two strategies. The difference at first increases as the ratio grows, which is caused by

the use of a biased arrival time estimator for on-going trips. However, the scheduling strategies OTS and in particular RES become much better than no scheduling at a high demand-to-supply ratio. The inability to predict the availability of taxis, even in the short term, significantly deteriorates the performance of NOS. The computational experiments also confirm that the quality of the distance measure may significantly improve the taxi service.

The created system may serve as a simulation-based optimization platform for benchmarking different vehicle-routing-related problems. The DVRP Optimizer constitutes a flexible framework where various optimization algorithms may be plugged into, and tested against diverse realistic scenarios simulated in MATSim. Similar platforms are widely applied in other areas, i.e. robosoccer (Kim 1997), rescue robot competitions (Takahashi et al. 2002), or energy trading simulations (Ketter et al. 2012, Bichler and Ketter 2010, Collins et al. 2010). In consequence, this would allow for obtaining more objective and comparable results.

Importantly, the approach provides quantitative numbers. Given a scenario where the demand and the vehicular travel times in the network are calibrated, one could work on calibrating the present approach towards the operating characteristics of a taxicab company. This will be the subject of future work.

References

???? Annual meeting preprint, Transportation Research Board, Washington D.C.

Alshamsi, A., S. Abdallah, I. Rahwan. 2009. Multiagent self-organization for a taxi dispatch system. *8th International Conference on Autonomous Agents and Multiagent Systems*. 21–28.

Arnott, Richard. 1996. Taxi travel should be subsidized. *Journal of Urban Economics* **40** 316333.

Ascheuer, N., S. Krumke, J. Rambau. 2000. Online dial-a-ride problems: Minimizing the completion time. *STACS 2000*. Springer, 639–650.

Bailey Jr, W.A., T.D. Clark Jr. 1992. Taxi management and route control: a systems study and simulation experiment. *Proceedings of the 24th conference on Winter simulation*. ACM, 1217–1222.

Barcelo, J., H. Grzybowska, S. Pardo. 2007. Vehicle routing and scheduling models, simulation and city logistics. Springer, NY, 163–195.

Ben-Akiva, M., S. R. Lerman. 1985. *Discrete choice analysis*. The MIT Press, Cambridge, MA.

Ben-Akiva, Moshe, Margaret Cortes, Angus Davol, Haris Koutsopoulos, Tomer Toledo. 2001. Mitsimlab: Enhancements and applications for urban networks. *9th World Conference on Transportation Research (WCTR), Seoul, Korea*.

Berbeglia, Gerardo, Jean-François Cordeau, Gilbert Laporte. 2010. Dynamic pickup and delivery problems. *European Journal of Operational Research* **202**(1) 8–15.

Bichler, A., M. Gupta, W. Ketter. 2010. Research commentary: Designing smart markets. *Information Systems Research* **21**(4) 688–699. doi:doi10.1287/isre.1100.0316.

Block, Carsten, John Collins, Lilia Filipova, Wolfgang Ketter. 2010. A competitive testbed for the development of intelligent agents in the energy domain. *International Conference on Group Decision and Negotiation (GND)*. Delft, The Netherlands, 271–275.

Burghout, W. 2004. Hybrid microscopic-mesoscopic traffic simulation. Ph.D. thesis, Royal Institute of Technology (KTH) Stockholm.

Burghout, W., H.N. Koutsopoulos. 2009. Hybrid traffic simulation models: Vehicle loading at meso-micro boundaries. E. Chung, A.-G. Dumont, eds., *Transport Simulation: Beyond Traditional Approaches*, chap. 2. EFPL Press, 27–41.

Cairns, Robert D, Catherine Liston-Heyes. 1996. Competition and regulation in the taxi industry. *Journal of Public Economics* **59**(1) 1–15.

Cascetta, E., C. Cantarella. 1991. A day-to-day and within-day dynamic stochastic assignment model. *Transportation Research A* **25A**(5) 277–291.

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

Collins, J., W. Ketter, N. Sadeh. 2010. Pushing the limits of rational agents: The trading agent competition for supply chain management. *AI Magazine* **31**(2) 63–80. URL `http://xlarge.rsm.nl/large/Collins10AIMag.pdf`.

Douglas, George W. 1972. Price regulation and optimal service standards: The taxicab industry. *Journal of Transport Economics and Policy* 116–127.

Fleischmann, B., S. Gnutzmann, E. Sandvoß. 2004. Dynamic vehicle routing based on online traffic information. *Transportation science* **38**(4) 420–433.

Gawron, C. 1998. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C* **9**(3) 393–407.

Gendreau, Michel, Gilbert Laporte, Frédéric Semet. 2001. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing* **27**(12) 1641–1653.

Grötschel, Martin, Sven O Krumke, Jörg Rambau, Thomas Winter, Uwe T Zimmermann. 2001. Combinatorial online optimization in real time. *Online* **16**(July) 679–704.

Horn, M.E.T. 2002. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies* **10**(1) 35–63.

Horni, A., K Nagel, K. Axhausen. 2012. High-resolution destination choice in agent-based demand models. Annual Meeting Preprint 12-1989, Transportation Research Board, Washington D.C. Also VSP WP 11-17, see www.vsp.tu-berlin.de/publications.

Jacob, R. R., M. V. Marathe, K. Nagel. 1999. A computational study of routing algorithms for realistic transportation networks. *ACM Journal of Experimental Algorithms* **4**(1999es, Article No. 6). doi: 10.1145/347792.347814.

Ketter, W., J. Collins, M. Gini, A. Gupta, L. Schrater. 2012. Real-time tactical and strategic sales management for intelligent agents guided by economic regimes. *Information Systems Research* **Articles in Advance** 1–21. doi:http://dx.doi.org/10.1287/isre.1110.0415.

Kim, J.H. 1997. Special issue about the first micro-robot world cup soccer tournament, MIROSOT. *Robotics and Autonomous Systems* **21**(2) 137–205.

Krumke, S., J. Rambau, L. Torres. 2002. Real-time dispatching of guided and unguided automobile service units with soft time windows. *AlgorithmsESA 2002* 417–424.

Lee, D.H., H. Wang, R.L. Cheu, S.H. Teo. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board* **1882**(-1) 193–200.

Lefebvre, N., M. Balmer. 2007. Fast shortest path computation in time-dependent traffic networks. *Proceedings of the Swiss Transport Research Conference (STRC)*. Monte Verita, CH. See www.strc.ch.

Liao, T.-Y., T.-Y. Hu, D.-J. Chen. 2008. Object-oriented evaluation framework for dynamic vehicle routing problems under real-time information. Annual Meeting Preprint 08-2222, Transportation Research Board, Washington D.C.

Ma, W., K. Wang. 2007. On the on-line weighted k-taxi problem. *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies* 152–162.

Maciejewski, M., K. Nagel. 2012. Towards multi-agent simulation of the dynamic vehicle routing problem in MATSim. R. Wyrzykowski et al, ed., *Parallel Processing and Applied Mathematics (PPAM), Revised Selected Papers, Part II*. Lecture Notes in Computer Science, Springer, 551–560. doi: 10.1007/978-3-642-31500-8_57.

Piatkowski, Bartlomiej, Michal Maciejewski. 2013 (in press). Comparison of traffic assignment in VISUM and transport simulation in MATSim. *Transport Problems* .

Powell, J., Y. Huang, F. Bastani, M. Ji. 2011. Towards reducing taxicab cruising time using spatio-temporal profitability maps. *Advances in Spatial and Temporal Databases* 242–260.

Regan, A.C., H.S. Mahmassani, P. Jaillet. 1998. Evaluation of dynamic fleet management systems: Simulation framework. *Transportation Research Record* **1645** 176–184.

Savelsbergh, M.W.P., M. Sol. 1995. The general pickup and delivery problem. *Transportation science* **29**(1) 17–29.

Seow, K.T., N.H. Dang, D.H. Lee. 2010. A collaborative multiagent taxi-dispatch system. *Automation Science and Engineering, IEEE Transactions on* **7**(3) 607–616.

Simão, H.P., W.B. Powell. 1992. Numerical methods for simulating transient, stochastic queueing networks. *Transportation Science* **26** 296–311.

Simon, P.M., J. Esser, K. Nagel. 1999. Simple queueing model applied to the city of Portland. *International Journal of Modern Physics* **10**(5) 941–960. doi:10.1142/S0129183199000747.

Takahashi, Tomoichi, Satoshi Tadokoro, Masayuki Ohta, Nobuhiro Ito. 2002. Agent based approach in disaster rescue simulation - from test-bed of multiagent system to practical application. Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro, eds., *RoboCup 2001: Robot Soccer World Cup V*, *Lecture Notes in Computer Science*, vol. 2377. Springer Berlin Heidelberg, 102–111. doi:10.1007/3-540-45603-1_11. URL `http://dx.doi.org/10.1007/3-540-45603-1_11`.

Taniguchi, E., R.G. Thompson, T. Yamada, R. van Duin. 2001. *City logistics – Network Modelling and Intelligent Transport Systems*. Pergamin, Amsterdam.

Wang, H., D.H. Lee, R. Cheu. 2009. PDPTW based taxi dispatch modeling for booking service. *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, vol. 1. IEEE, 242–247.

Yang, H., SC Wong. 1998. A network model of urban taxi services. *Transportation Research Part B: Methodological* **32**(4) 235–246.

Yang, Hai, Min Ye, Wilson Hon-Chung Tang, Sze Chun Wong. 2005. A multiperiod dynamic model of taxi services with endogenous service intensity. *Operations research* **53**(3) 501–515.

Yang, J., P. Jaillet, H. Mahmassani. 2004. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science* **38**(2) 135–148.

Yuan, Jing, Yu Zheng, Liuhang Zhang, XIng Xie, Guangzhong Sun. 2011. Where to find my next passenger. *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 109–118.