Michał Maciejewski[1]
TU Berlin, Transport Systems Planning (VSP)
Poznan University of Technology, Division of Transport Systems

Online taxi dispatching via exact offline optimization

1. INTRODUCTION

Although there are thousands of taxi companies all over the world, managing millions of taxicabs altogether, the problem of online taxi dispatching has not been explored thoroughly. To some extent, this used to be conditioned by the lack of technical means to coordinate a fleet of taxis. Nowadays, due to the recent developments in mobile technologies (mobile devices, geopositioning, wireless communication, etc.), taxi companies are able to centrally coordinate the dispatching process. This raises the need for developing both efficient online algorithms and simulation environments where the performance of those algorithms may be evaluated [13].

A formulation of the Online Taxi Dispatching Problem, based on the Online *k*-Server Problem, can be found in [10]. Several simulation-based studies have been carried out to examine different dispatching strategies, assuming either partial autonomy of taxi drivers [1, 3, 15] or a fully centralized dispatch process [8, 12, 16]. To the best knowledge of the authors, detailed microscopic simulation of taxi fleets combined with other traffic has been carried out only for Singapore [8, 15] and Mielec [12, 13, 14]. Regardless of the limited research on taxi dispatching, many ideas and algorithmic solutions can be found in studies on Personal Rapid Transport (PRT) [9] and Demand Responsive Transport (DRT) [7]. Also dynamic truckload pickup and delivery problems [4, 17] and dynamic management of emergency services [5] share some similarities with online taxi dispatching.

This paper proposes and evaluates a dispatching strategy that makes decisions by finding an exact solution of the offline problem at each decision epoch. This strategy is meant to serve as a benchmark for real-time strategies. Similar idea have been proposed for the Real-Time Multivehicle Truckload Pickup and Delivery Problem in [17].

2. SIMULATION OF DYNAMIC TRANSPORT SERVICES

In order to simulate online taxi dispatching or other dynamic vehicle routing and scheduling problems, a dedicated MATSim extension, DVRP, has been developed [11, 12]. MATSim is an agent-based system that provides means for microscopic, activity-based simulation of transport systems through an iterative process of day-to-day learning [2]. The DVRP extension allows for modelling a wide spectrum of dynamic vehicle routing/scheduling problems, by introducing a simulation framework where:

- each driver is modelled as an agent that follows his/her dynamic schedule; this is in contrast to the standard day-to-day learning approach used in MATSim
- a driver's/vehicle's schedule is a sequence of tasks of different types, such as driving from one location to another or waiting at a given location
- optimization is triggered by events that reflect changes in the system
- a fleet of vehicles comprises one component of the whole traffic flow simulated by means of one of the queue-based simulators available in MATSim
- each vehicle can be monitored online as it moves from link to link; this information may be used to update the timing of its schedule and possibly trigger re-optimization
- in the case of passenger transport (e.g. taxi, DRT), interaction between the dispatcher, drivers and passengers is simulated in detail, including calling a ride, picking up/dropping off passengers, etc.

[1] michal.maciejewski@put.poznan.pl; maciejewski@vsp.tu-berlin.de

3. ONLINE TAXI DISPATCHING

Let $N = \{1,\ldots,n\}$ be the set of taxi requests (customers). Customers, in contrast to taxi drivers, are modelled as the standard MATSim agents, each of them having a daily plan (consisting of legs and activities) that does not change during simulation. The simulation framework assumes the following sequence of events related to serving each request $i \in N$ (illustrated in Fig.1): Taxi customer *i* calls a taxi at time τ_i^{call} (event E_i^{call}) specifying the pickup location, p_i, and the time of departure, $\tau_i^{\text{dep}} \geq \tau_i^{\text{call}}$. The dropoff location, d_i, is specified if requested by the dispatcher (*destination known a priori*). For an *immediate* request, we have $\tau_i^{\text{dep}} = \tau_i^{\text{call}}$, whereas for an *advance* request, $\tau_i^{\text{dep}} > \tau_i^{\text{call}}$. At time $\tau_i^{\text{disp}} \geq \tau_i^{\text{call}}$, a taxi is dispatched to customer *i* (event E_i^{disp}) and picks him/her up at time $\tau_i^{\text{pick0}} \geq \tau_i^{\text{dep}}$ (event E_i^{pick0}; start of the pickup). Once the customer is picked up (time τ_i^{pick1}, event E_i^{pick1}), he/she specifies the destination, d_i, unless provided before (*destination unknown a priori*). Next, the taxi sets out towards d_i and after reaching it at time τ_i^{drop0}, the dropoff starts (event E_i^{drop0}). Once the passenger gets out (time τ_i^{drop1}, event E_i^{drop1}), the taxicab becomes ready to serve the next request.

At time *t*, request $i \in N$ may be in one of the following four states:

- *unplanned* – $\tau_i^{\text{disp}}, \tau_i^{\text{pick0}}, \tau_i^{\text{pick1}}, \tau_i^{\text{drop0}}$ and τ_i^{drop1} are undefined
- *planned* – $t < \tau_i^{\text{disp}}$
- *started* – $\tau_i^{\text{disp}} \leq t < \tau_i^{\text{drop1}}$
- *performed* – $t \geq \tau_i^{\text{drop1}}$

Times τ_i^{call} and τ_i^{dep} are provided by the customer. On the other hand, event arrival times $\tau_i^{\text{disp}}, \tau_i^{\text{pick0}}, \tau_i^{\text{pick1}}, \tau_i^{\text{drop0}}$ and τ_i^{drop1} are estimated until the respective events take place, and therefore, can change in the course of simulation. The accuracy of these predictions may be improved, especially in congested traffic, by the use of the online vehicle tracking functionality available in the DVRP extension.

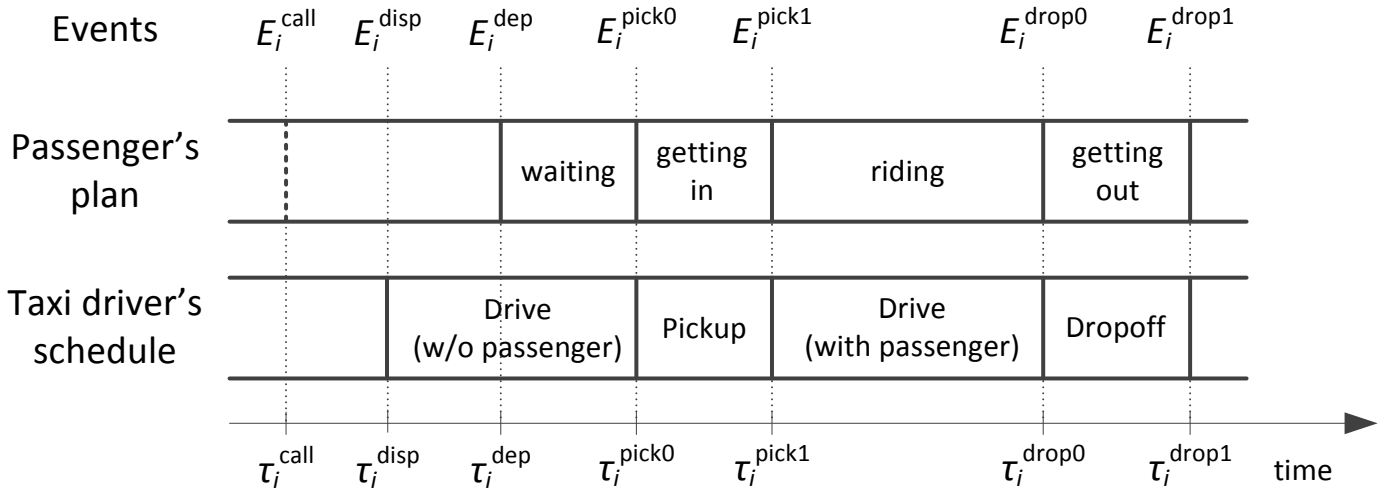

Fig. 1. Connection between a taxi driver's schedule and a passenger's plan when serving request *i*

Let $M = \{1,\ldots,m\}$ be the set of vehicles. Each vehicle $k \in M$ is available at location o_k from time $a_k \geq \tau^{\text{curr}}$ onwards, where τ^{curr} denotes the current time. Assuming that vehicles neither cruise nor return to taxi ranks, o_k is the destination of the last customer served by *k* or *k*'s home location if the taxi has not served any request yet. For an idle vehicle *k*, we have $a_k = \tau^{\text{curr}}$, otherwise, a_k is the time *k* finishes serving its last request. An active vehicle may have temporarily undefined availability if it has been dispatched to *i* while d_i is unknown a priori. In such cases, both o_k and a_k are unknown between τ_i^{disp} and τ_i^{pick1}.

Let $t_{ki}^{O}(t)$, $k \in M$, $i \in N$, be the time-dependent travel time from o_k to p_i, and $t_{ij}(t)$, $i \in N$, $j \in N$, be the time-dependent travel time from d_i to p_j, both being functions of departure time t. Let $t_i^S(t)$, $i \in N$, be the time-dependent total serve time of customer i, including picking up, driving and dropping off, where t is the time when the pickup starts.

Let L be the list of all *unplanned* and *planned* requests in N, ordered by T_i^{dep}. Each request $i \in N$ is inserted into L on submission, τ_i^{call}, and removed from L on taxi dispatch, τ_i^{disp}. Let $M^A \subseteq M$ be the set of available vehicles, i.e. vehicles $k \in M$ of which o_k and a_k are known. Let $M^I \subseteq M^A$ be the set of idle vehicles, i.e. vehicles $k \in M$ that are waiting for the next request at o_k and available from now on, $a_k = \tau^{curr}$.

One should note that all collections defined above change over time and should be written as functions of time t, e.g. $N(t)$ instead of N. However, for the sake of readability, a simplified notation is used, assuming that the values are given for the current time, τ^{curr}, for instance, $N \equiv N(\tau^{curr})$.

4. The Offline Taxi Dispatching Problem

Let $V = \{1,\ldots,m,m+1,\ldots,m+n\}$ be the set of vertices representing both vehicles (each vehicle $k \in M$ is represented by vertex k) and requests (each request $i \in N$ is represented by vertex $m+i$). The earliest departure time of customer i is τ_i^{dep}, $i \in N$. As in [17], we model the offline problem as an assignment problem, where the goal is to find cycles of vertices, represented by variables x_{uv}, $u \in V$, $v \in V$, and the pickup times, represented by variables τ_i^{pick0}, $i \in N$, such that the total waiting time is minimized. We assume that taxi k remains at d_i of the last served request, i, or at o_k if it has not been dispatched yet. Each cycle contains $r > 0$ vehicle vertices and thus represents r open-ended routes. The conversion from cycles to routes (list of vertices) is done by removing all arcs leading to vehicle vertices, that is $x_{uk} = 1$, $u \in V$, $k \in M$. As a result, we obtain m routes, where route k starts at vertex k and is served by vehicle k. If route k contains only vertex k, $x_{kk} = 1$, vehicle k does not serve any request, and therefore, stays at o_k.

In order to formulate the offline problem as a mixed integer programming problem, instead of the time-dependent travel/serve time functions, $t_{ki}^{O}(t)$, $t_{ij}(t)$ and $t_i^S(t)$, the average travel/serve times, t_{ki}^{O}, t_{ij} and t_i^S, $i \in N$, $j \in N$, $k \in M$, are used.

The offline taxi dispatching problem may be stated as:

$$\min \sum_{i \in N} \tau_i^{pick0} - \tau_i^{dep} \tag{1}$$

subject to

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

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

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

$$\tau_i^{pick0} - \sum_{k \in M} (a_k + t_{ki}^{O}) \cdot x_{k,m+i} \geq 0 \quad \forall i \in N, \tag{5}$$

$$\tau_j^{pick0} - \tau_i^{pick0} - t_i^{S} - t_{ij} + T \cdot (1 - x_{m+i,m+j}) \geq 0 \quad \forall i \in N, \forall j \in N, \tag{6}$$

$$\tau_i^{pick0} \geq \tau_i^{dep} \quad \forall i \in N. \tag{7}$$

The objective (1) is to minimize the total waiting time of customers. Constraints (2)–(4) ensure that the assignment is valid, which means that each vertex is visited exactly once. Constraints (5) guarantee that taxicab *k* arrives at the pickup location of customer *i* at time $a_k + t^{O}_{ki}$ or later, given that *i* is the first customer in route *k*, $x_{k,m+i} = 1$. Constraints (6) ensure that after picking up customer *i*, the vehicle picks up customer *j* after at least $t^{S}_i + t_{ij}$, the amount of time required to serve *i* and travel from d_i to p_j. Given that *T* is large enough, constraints (6) are not restrictive when $x_{m+i,m+j} = 0$, $i \in N$, $j \in N$. Additionally, constraints (6) eliminate cycles without a vehicle (subtour elimination constraints). Constraints (7) ensure that the pickup of customer *i* starts at time τ^{dep}_i or later.

5. ONLINE TAXI DISPATCHING STRATEGIES

5.1. General assumptions

The following assumptions have been made concerning the dispatching process:

- the goal is to serve all customers $i \in N$ such that the total waiting time is minimized
- only immediate requests are considered, i.e. $\tau^{\text{dep}}_i = \tau^{\text{call}}_i$, $i \in N$
- no knowledge about future requests, no statistics from the past, therefore, idle vehicles remain at the current location instead of moving them towards more attractive areas
- re-optimization may be triggered by any of the events described in Sec.3
- events related to the movement of vehicles (via online taxi monitoring) may be used for updating the timings of schedules; triggering re-optimization is not considered
- strategies may require providing the dropoff location, d_i, at the time of submission, τ^{call}_i; otherwise, the dropoff is provided after the pickup, τ^{pick1}_i, $i \in N$. The former implies $M^{\text{A}} = M$, whereas for the latter, the planning horizon size is limited to $|M^{\text{A}}|$ since only one request can be added to the schedule of an available vehicle (making the vehicle unavailable until the pickup is done).

5.2. The MIP strategy

This paper studies the performance of an online dispatching strategy, called MIP, that at each decision epoch, transforms the online taxi dispatching problem into the offline counterpart formulated as a mixed integer programming problem (Sec.4) and solves it. By decision epoch we understand the period between the arrivals of two subsequent events of selected types.

The strategy uses a rolling horizon of length *h*, which means that only the first $\min(h, |L|)$ requests in *L* are considered in the optimization process. Since dropoff locations are known in advance, which is implied by the offline problem, $M^{\text{A}} = M$. The MIP strategy triggers re-optimization in response to events E^{call}_i and E^{pick1}_i, provided the following conditions are satisfied:

- E^{call}_i – request *i* is inserted at one of the first *h* positions of *L*; *i* is a new request included into the planning horizon
- E^{disp}_i – request *i* is removed from *L*, resulting in shifting request *j* from position *h+1* to *h*; *j* is a new request included into the planning horizon

5.3. Reference strategies

So far many different online taxi dispatching strategies have been created and used with the DVRP extension, including NOS, OTS and RES [12, 13, 14]. Those strategies are real-time heuristics that find solutions almost immediately, even for larger instances, as opposed to MIP that solves the offline problem by means of an exact branch-and-bound algorithm, but at the cost of computational time. On the other hand,

MIP is expected to find solutions that are closer to the optimum. This section shortly describes NOS and RES strategies that serve as the reference point for MIP.

The *no-scheduling strategy* (NOS) imitates a typical dispatching strategy used by many taxi companies. NOS does not use any information about dropoff locations to predict the availability of busy vehicles, hence only the idle vehicles, M^{I}, are considered. It reacts to the following events concerning request *i*:

- E_i^{call} – if $M^{\mathrm{I}} \neq \varnothing$, the nearest vehicle $k \in M^{\mathrm{I}}$ is dispatched to *i*; otherwise *i* is inserted into *L*
- E_i^{drop1} – if $L \neq \langle\rangle$, the vehicle that has served *i* is dispatched to the first request in *L*, $L[1]$; otherwise the vehicle is added to M^{I}

This is the simplest and fastest strategy. Since it does not create schedules, travel times do not have to be given, and a straight-line distance may be used instead to select the closest vehicle. However, in an overloaded system, when $\left|M^{\mathrm{I}}\right| \to 0$, the nearest taxi may appear on the opposite side of a city. Dispatching such a distant taxi leads to significant performance deterioration.

In contrast to NOS, the *re-scheduling strategy* (RES) builds schedules by appending requests to the existing schedules of the available taxis, M^{A}. The strategy reacts to the following events:

- E_i^{call} – request *i* is assigned to vehicle $k \in M^{\mathrm{A}}$ such that τ_i^{pick0} is minimal
- E_i^{pick1} – if destinations are unknown a priori, the vehicle that is currently serving *i* is added to M^{A}, which triggers full re-optimization, i.e. all *planned* requests are removed from schedules and scheduling is performed again for each $i \in L$, according to the order of submission

In this paper, we assume that (a) destinations are unknown a priori[2], and (b) if a vehicle gets ahead of/behind time, only the timing of the schedule is updated. However, RES can be set up to react to all delays/speedups by fully re-optimizing the schedules. This behaviour is particularly desired if dropoff locations are known in advance, and therefore, under high load, schedules can extend to the far future.

Even if destinations are unknown a priori, RES is expected to perform better than NOS because it considers a broader range of vehicles when scheduling requests, $\left|M^{\mathrm{A}}\right| \geq \left|M^{\mathrm{I}}\right|$. Particularly under full load, when no vehicle is idle and NOS generates random assignments, RES has still a considerable number of options to consider. The relation between $\left|M^{\mathrm{A}}\right|$ and $\left|M^{\mathrm{I}}\right|$ can be approximated as:

$$\left|M^{\mathrm{A}}\right| \approx \frac{\sum_{i \in N} \tau_i^{\mathrm{drop1}} - \tau_i^{\mathrm{pick1}}}{\sum_{i \in N} \tau_i^{\mathrm{drop1}} - \tau_i^{\mathrm{disp}}} \left|M\right| \gg \left|M^{\mathrm{I}}\right| \approx 0, \tag{8}$$

and for typical distribution of pickups and dropoff locations, $\left|M^{\mathrm{A}}\right| > 0.5\left|M\right|$.

6. TEST SCENARIO

The computational experiments were run on a small-size toy model of the city of Mielec, Poland, with a population of over 60,000 inhabitants. The demand comprises of over 56,000 private car trips between 6:00 am and 8:00 pm. The performance of the dispatching strategies was tested at different levels of taxi demand, where the set of requests, *N*, consisted of *n* = 406 (1% of all trips), 636 (1.5%), 840 (2%), 1069 (2.5%), 1297 (3%), 1506 (3.5%) and 1719 (4%) randomly selected private car trips, hence the spatiotemporal distributions of taxi and private car trips were virtually identical. The set of vehicles, *M*, comprises *m* = 25 taxis.

To obtain reliable travel time estimates for simulating taxis, first the scenario was run for 20 iterations without taxis to reach a state of relaxation. Next, after changing the mode of *n* trips from *private car* to *taxi*, which introduces additional traffic due to the movement of empty taxis, 5 additional *warmup* iterations (with the day-to-day learning switched off) were run to calculate 5-day averages of travel times. To provide

[2] In general, the performance of RES with and without destinations provided a priori is similar [13]

comparability, this step was carried out independently for each dispatching setup, since a different way of dispatching may result in different travel times, especially for higher shares of taxi trips, e.g. 4%.

The MIP strategy used the branch-and-bound (BB) algorithm available in the Gurobi Optimizer [6], version 5.6.2, to solve the offline MIP problem. Three different horizon lengths were analysed, all being a multiple of the fleet size, i.e. $h = m$, $2m$, $3m$. The time of a single offline optimization was limited to 30 and 60 seconds in order to comply with the online dispatching requirements. By default, BB was run in parallel on all CPU cores. The initial feasible solution was calculated with RES.

Table 1 presents the setups used for the strategies compared in this study. The experiments were run on a computer with the 6-core Intel Core i7-3930K processor with Hyper-Threading Technology (12 logical cores) and 64 GB of RAM.

Table. 1. Configurations of the strategies

	MIP	NOS	RES
Destinations known a priori	yes	n/a	no
Travel time averaging interval	24 hours	15 minutes	15 minutes
Planning horizon, h	m, $2m$, $3m$	1	$\lvert M^{\mathrm{A}}\rvert^{*}$
Time limit [s]	30, 60	n/a (real-time)	n/a (real-time)

[*] *If dropoff locations are known a priori, the length of the planning horizon is* $|L|$

7. SIMULATION RESULTS

Several different performance measures proposed in [12] were investigated, representing the perspectives of either the customers or the taxi company, or of both. The following two measures are analysed in depth in this section:

- average passenger waiting time, $T_{\mathrm{W}} = \sum_{i \in N} \left(\tau_i^{\mathrm{pick0}} - \tau_i^{\mathrm{dep}} \right) \big/ n$
- average pickup (nonrevenue) trip time, $T_{\mathrm{P}} = \sum_{i \in N} \left(\tau_i^{\mathrm{pick0}} - \tau_i^{\mathrm{disp}} \right) \big/ n$

Although both measures seem mutually exclusive, for immediate requests, where $\tau_i^{\mathrm{dep}} \le \tau_i^{\mathrm{disp}}$, $i \in N$, the minimization of T_{W} implies implicitly the minimization of T_{P}, whereas the opposite is not the case.

Figures 2 and 3 present the results, in terms of T_{W} and T_{P}, obtained for different n and the following configurations:

- *MIP h1 t60* – MIP, $h = 1m$, 60-second time limit (best performing MIP configuration)
- *MIP h3 t30* – MIP, $h = 3m$, 30-second time limit (worst performing MIP configuration)
- NOS
- RES

For the sake of readability, only the best and worst performing MIP configurations are presented. The performance of the remaining four MIP configurations is placed in between for both T_{W} and T_{P}.

Figure 2 shows that, in general, MIP performs better than NOS and RES. At low load, NOS is slightly better than the rest, which is supposedly caused by the following two factors:

- inaccuracies of travel time estimation – NOS does not rely on them since it takes into account idle vehicles only. With online vehicle monitoring turned on, these inaccuracies occur only at the level of individual links (not whole paths), as a result the estimation bias is significantly reduced.
- spatial asymmetry of taxi demand – the spatiotemporal distribution of dropoffs (and therefore idle vehicles) is different than that of pickups. NOS does not take into account busy vehicle. Consequently, it moves idle vehicles more eagerly than other strategies towards the zones with higher demand.

At medium and high load, $n > 800$, MIP is better than RES, while NOS performs worst. The differences between the MIP configurations come into play at high load, $n > 1200$, when more and more frequently the solver was not able to find the optimal solution within the time limit. The longer the time horizon and the

shorter the time limit, the more deteriorated the performance was. For the *MIP h3 t30* configuration and $n = 1719$, the offline optimization was regularly terminated before any improvement relative to the initial solution generated with RES was made, thus MIP's performance is close to that of RES.

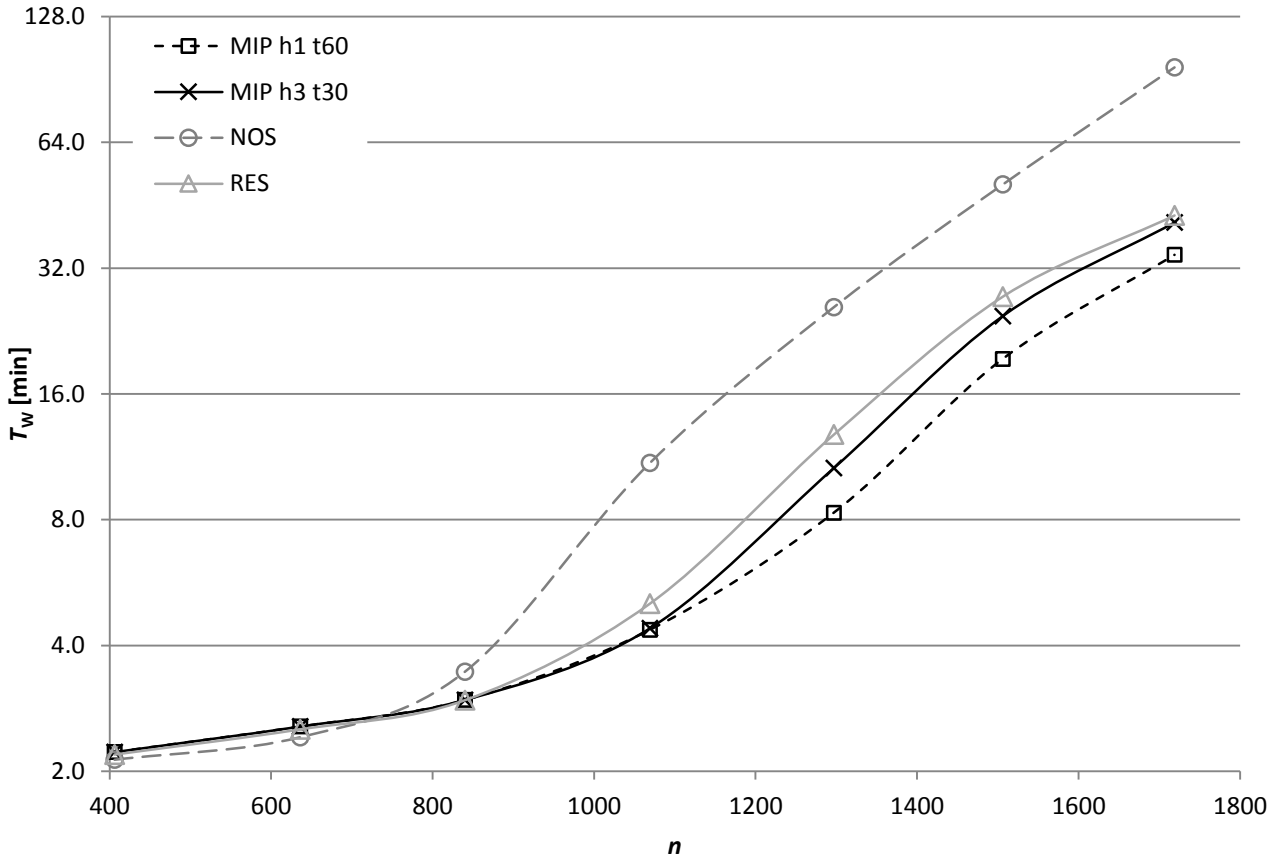

Fig. 2. Average waiting time, T_W, as a function of n for the selected strategies

Figure 3 shows that NOS always performs worst due to not taking into account busy vehicles. Considering all available taxis at current time, τ^{curr}, helps minimizing the remaining (future) waiting time of request i, $\tau_i^{\text{pick0}} - \tau^{\text{curr}}$, and thus T_W, and additionally allows for $\tau_i^{\text{disp}} > \tau^{\text{curr}}$, which reduces the pickup trip time, $\tau_i^{\text{pick0}} - \tau^{\text{disp}}$, and thus T_P, even further. Moreover, at high load, the average pickup trip time is almost equal to the average dropoff trip time (5.9 minutes), which shows that the assignments are virtually random.

Under low load, RES and MIP perform similarly in terms of T_P, whereas at medium to high load, MIP is better. Interestingly, for $n = 1297$, *MIP h1 t60* is still able to find the optimal solutions, while *MIP h3 t30* is often terminated, especially at peak hours. However, under high load, $n > 1400$, the former also is unable to find the optimum at peak hours. The drop in T_P at $n = 1297$ for *MIP h1 t60* is the side effect of the increased number of unserved (both unplanned and planned) requests resulting in the decrease of the average distance between each vehicle and its nearest requests. T_P drops since vehicles are usually dispatched to nearby requests. This downward trend might have continued at higher n but for the limited optimization time preventing finding the exact solutions (MIP behaves more and more like RES).

8. CONCLUSIONS

The paper presents a general framework for simulating taxi dispatching at the microscopic level, embedded into traffic simulation. The simulation scenario for Mielec was run in MATSim with the DVRP extension module. By modelling each individual, either a taxi driver or a customer, as a separate agent, we can simulate their behaviour and measure their performance at a high level of detail.

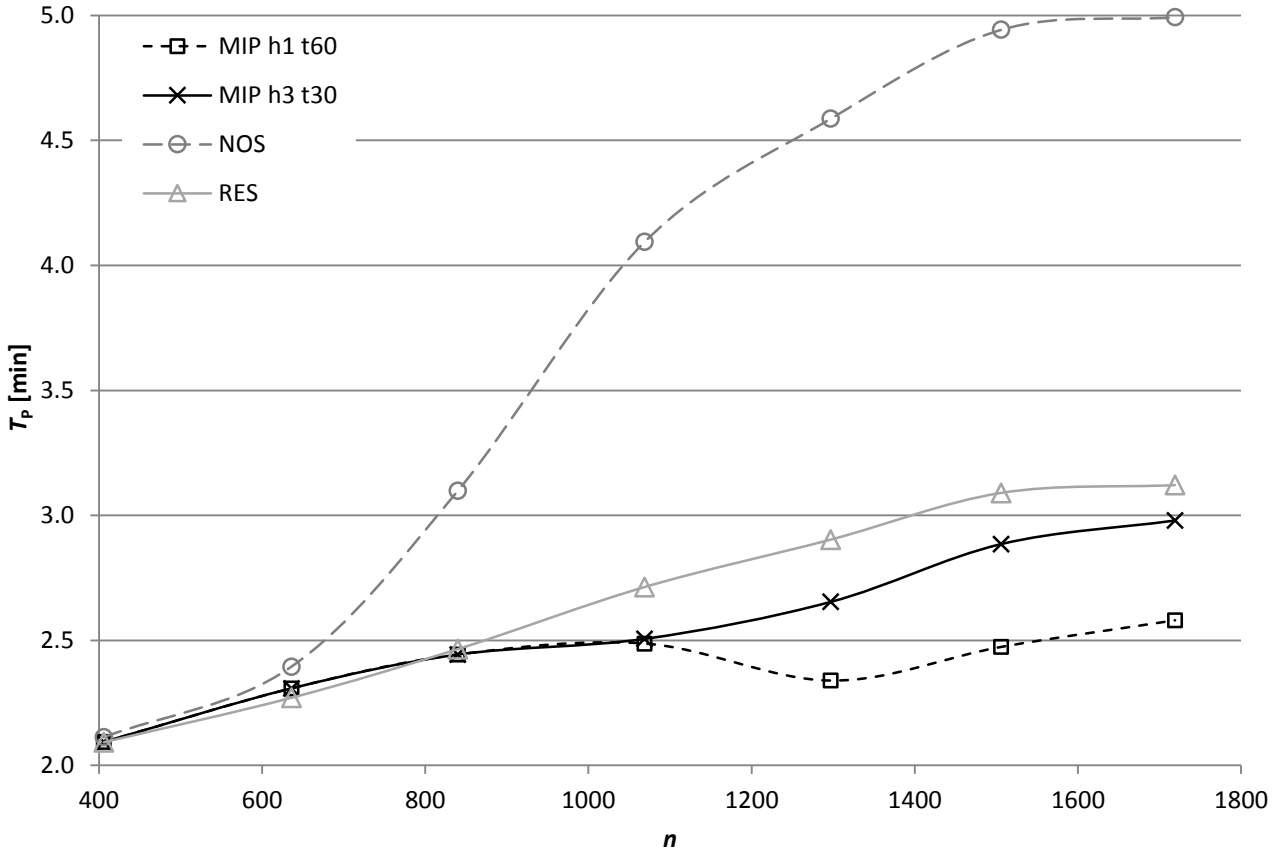

Fig. 3. Average pickup trip time, T_P, as a function of n for the selected strategies

The main purpose of this paper, however, was to evaluate the possibility of dispatching taxis online by solving the offline problem with an exact branch-and-bound algorithm at each decision epoch (the MIP strategy). Out of six different MIP configurations, the best results were obtained for the shortest rolling horizon, $h = m$, and the highest time limit of a single optimization, 60 seconds. A slightly worse performance was observed for $h = m$ and the 30-second time limit. Regardless of the configurations, for $n > 1400$, MIP often (especially during peak hours) could not find the optimal solution, or even improve the initial one (calculated with RES), which limits the use of MIP to scenarios of low to medium load. When compared to the two reference real-time heuristics, MIP proves valuable at medium load due to the best performance, whereas under low load, RES is a better choice because of shorter computation time.

MIP may be used for larger instances provided that the time limit is longer and the planning horizon is kept short (does not scale with m). Moreover, given a short horizon, the offline optimization can be improved by, for instance, using travel time estimates for the following time period (e.g. the next hour) instead of the 24-hour averages, or imposing upper limits on the variables τ_i^{pick0}. The results obtained for MIP prove that there is room for improvement in the real-time taxi dispatching (the gap between MIP and RES).

Abstract

The paper proposes an online taxi dispatching strategy that is based on solving exactly the equivalent offline problem at each decision epoch. First, a general framework for simulating dynamic transport services in MATSim (Multi-Agent Transport Simulation) is described. Next, the model of online taxi dispatching is defined, followed by a formulation of the offline problem as a mixed integer programming problem. Then the MIP strategy that solves the offline problem with a finite planning horizon at each decision epoch is described. The performance of MIP was evaluated on the simulation scenario of taxi services in the city of Mielec and compared with that of two selected strategies (RES and NOS). The obtained results show the advantage of MIP over the reference strategies in terms of the solution quality, but at the cost of response time.

Keywords: online taxi dispatching, dynamic vehicle routing, rolling horizon, multi-agent simulation, MATSim

Przydział taksówek do zleceń w trybie online poprzez optymalizację dokładną offline

Streszczenie

W artykule zaproponowano strategię przydziału online taksówek do zleceń opartą na dokładnym rozwiązaniu odpowiednika problemu w trybie offline dla każdej epoki decyzyjnej. W pierwszej kolejności opisano ogólny szkielet symulacji dynamicznych usług transportowych w MATSim (Multi-Agent Transport Simulation). Następnie zdefiniowano model przydziału taksówek do zleceń w trybie online oraz sformułowano problem offline jako problem programowania mieszanego całkowitoliczbowego MIP. W oparciu o ten problem stworzono strategię MIP polegającą na rozwiązywaniu problemu offline dla skończonego horyzontu planistycznego dla każdej epoki decyzyjnej. Efektywność strategii MIP oceniono na podstawie scenariusza symulacyjnego usług taksówkowych w Mielcu i porównano z efektywnością dwóch wybranych strategii (RES i NOS). Otrzymane rezultaty wskazują na przewagę MIP nad strategiami referencyjnymi pod względem jakości rozwiązań, ale kosztem czasu odpowiedzi.

Słowa kluczowe: przydział taksówek do zleceń w trybie online, dynamiczna marszrutyzacja pojazdów, ruchomy horyzont planistyczny, symulacja wieloagentowa, MATSim

REFERENCES

[1] Alshamsi A., Abdallah S., Rahwan I.: Multiagent self-organization for a taxi dispatch system. In: 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 21–28, 2009.
[2] Balmer M., Meister K., Rieser M., Nagel K., Axhausen K.: Agent-based simulation of travel demand: structure and computational performance of MATSim-T. In: Innovations in Travel Modeling (ITM) ’08, Portland, Oregon, June 2008, also VSP WP 08-07, www.vsp.tu-berlin.de/publications, 2008.
[3] Cheng S., Nguyen T.: 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, Vol. 2, IEEE Computer Society, pp. 14–21, 2011.
[4] Fleischmann B., Gnutzmann S., Sandvoß E.: Dynamic vehicle routing based on online traffic information. Transportation Science, 38(4), pp. 420–433, 2004.
[5] Gendreau M., Laporte G., Semet F.: A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Parallel computing, 27(12), pp. 1641–1653, 2001.
[6] Gurobi Optimizer Reference Manual, Version 5.6, 2013.
[7] Horn M.: Fleet scheduling and dispatching for demand-responsive passenger services. Transportation Research Part C Emerging Technologies, 10(1), pp. 35–63, 2002.
[8] Lee D., Wang H., Cheu R., Teo S.: Taxi dispatch system based on current demands and real-time traffic conditions. Transportation Research Record, Journal of Transportation Research Board, 1882, pp. 193–200, 2004.
[9] Lees-Miller J.D.: Empty vehicle redistribution for personal rapid transit. PhD thesis, University of Bristol, 2011.
[10] Ma W., Wang K.: On the on-line weighted k-taxi problem. In: Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Springer, pp. 152–162, 2007.
[11] Maciejewski M., Nagel K.: Towards multi-agent simulation of the dynamic vehicle routing problem in MATSim. In: Wyrzykowski R., Dongarra J., Karczewski K., Wasniewski J. (eds.): Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, 7204, Springer Berlin Heidelberg, pp. 551–560, 2012.
[12] Maciejewski M., Nagel K.: Simulation and dynamic optimization of taxi services in MATSim. VSP Working Paper 13-05, TU Berlin, Transport Systems Planning and Transport Telematics. www.vsp.tu-berlin.de/ publications, 2013.
[13] Maciejewski M., Nagel K.: A microscopic simulation approach for optimization of taxi services. In: Albrecht T., Jaekel B., Lehnert M. (eds.): Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems, Dresden, p. 1–10, 2013.
[14] Maciejewski M., Nagel K.: The influence of multi-agent cooperation on the efficiency of taxi dispatching, In: Wyrzykowski R., Dongarra J., Karczewski K., Wasniewski J. (eds.): Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, 8385, Springer Berlin Heidelberg, p. 751–760, 2014.
[15] Seow K., Dang N., Lee D.: A collaborative multiagent taxi-dispatch system. IEEE Transactions on Automation Science and Engineering, 7(3), pp. 607–616, 2010.
[16] Wang H., Lee D., Cheu R.: PDPTW based taxi dispatch modeling for booking service. In: Fifth International Conference on Natural Computation, ICNC’09, vol. 1, pp. 242–247, 2009.
[17] Yang J., Jaillet P., Mahmassani H.: Real-time multivehicle truckload pickup and delivery problems. Transportation Science, 38(2), pp. 135–148, 2004.