THE ARCHIVES OF TRANSPORT *ISSN (print): 0866-9546*
Volume 30, Issue 2, 2014 ***e-ISSN (online):*** *2300-8830*
DOI: *10.5604/08669546.1146978*

BENCHMARKING MINIMUM PASSENGER WAITING TIME IN ONLINE TAXI DISPATCHING WITH EXACT OFFLINE OPTIMIZATION METHODS

Michał Maciejewski[1, 2]

[1]TU Berlin, Faculty of Mechanical Engineering and Transport Systems, Transport Systems Planning and Transport Telematics (VSP), Berlin, Germany
e-mail: maciejewski@vsp.tu-berlin.de

[2]Poznan University of Technology, Faculty of Machines and Transport, Division of Transport Systems, Poznan, Poland
e-mail: michal.maciejewski@put.poznan.pl

Abstract: *This paper analyses the use of exact offline optimization methods for benchmarking online taxi dispatching strategies where the objective is to minimize the total passenger waiting time. 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. Three benchmarks based on the offline problem are presented and compared to two simple heuristic strategies and a hypothetical simulation with teleportation of idle taxis. The benchmarks are evaluated and compared using the simulation scenario of taxi services in the city of Mielec. The obtained (approximate) lower and upper bounds for the minimum total passenger waiting time indicate directions for further research.*

Key words: *online taxi dispatching, dynamic vehicle routing, multi-agent simulation, MATSim*

1. Introduction

The recent developments in ICT (Information and Communication Technology) have opened new opportunities for operating taxi fleets in real time and aroused interest in the online taxi dispatching problem. As lots of optimization methods have got devised over the last decade [1, 3, 6, 7, 10, 12, 13, 14, 15], benchmarking their performance is of the utmost importance. This pertains to development of both test scenarios, including simulation platforms, and reference dispatching strategies, the latter being the subject of this paper.

Competitive analysis is a popular method used for assessing the performance of online algorithms against an optimal offline algorithm, given that the latter has the full knowledge of the future. However, this approach is typically applicable only for small, theoretical instances of online problems, whereas for real-world cases, one has to assess the performance experimentally (e.g. through simulation) by comparing it to that of some selected reference, although not necessarily optimal, algorithms [4]. In this paper, exact offline optimization methods are proposed as benchmarks for online taxi dispatching strategies where the objective is to minimize the total passenger waiting time.

The following sections: *Simulation platform*, *Online taxi dispatching*, *The Offline Taxi Dispatching Problem* and *Test scenario*, are revised versions of respective sections from [8]. The rest of the paper makes an original contribution to the field.

2. Simulation platform

In order to simulate online taxi dispatching or other dynamic vehicle routing and scheduling problems, a dedicated MATSim extension, DVRP, has been developed [9, 11]. 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.

3. Online taxi dispatching

Let $N = \{1,\ldots,n\}$ be the set of taxi requests (customers). The simulation framework assumes the following sequence of events related to serving each request $i \in N$ (illustrated in Fig. 1 and 2): 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 only 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 selected taxi is dispatched to customer *i* (event E_i^{disp}) and reaches location p_i at time τ_i^{ready}. If $\tau_i^{\text{dep}} > \tau_i^{\text{ready}}$, the taxi waits for the customer to come; otherwise, if $\tau_i^{\text{dep}} < \tau_i^{\text{ready}}$, the customer waits for the taxi to arrive. The pickup starts at time $\tau_i^{\text{pick0}} = \max\left(\tau_i^{\text{dep}}, \tau_i^{\text{ready}}\right)$ and is represented by event E_i^{pick0}. 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, times $\tau_i^{\text{disp}}, \tau_i^{\text{ready}}, \tau_i^{\text{pick0}}, \tau_i^{\text{pick1}}, \tau_i^{\text{drop0}}$ 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 online vehicle tracking.

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 the last customer in its schedule, *i*, has not provided d_i on submission (time τ^{call}). In such cases, both o_k and a_k remain unknown until the pickup is finished (time τ_i^{pick1}).

Let $t_{ki}^{\text{O}}(t)$, $k \in M$, $i \in N$, 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.

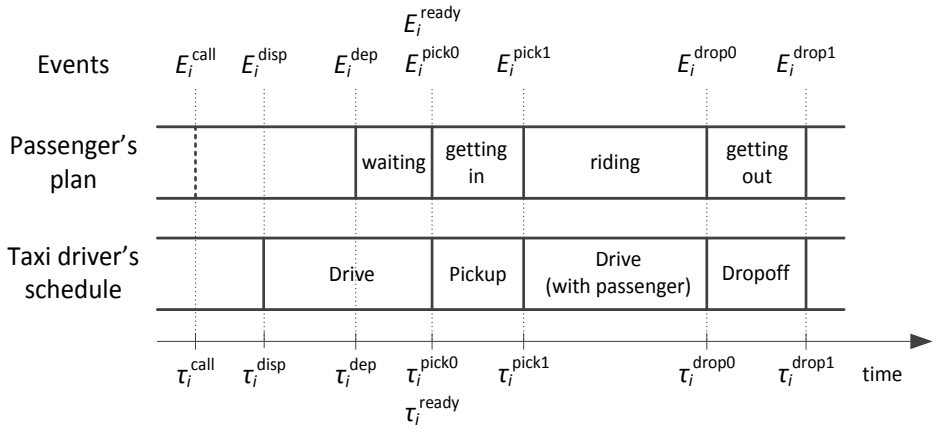

Fig. 1. A taxi driver's schedule and a passenger's plan (the passenger waits for the taxi)

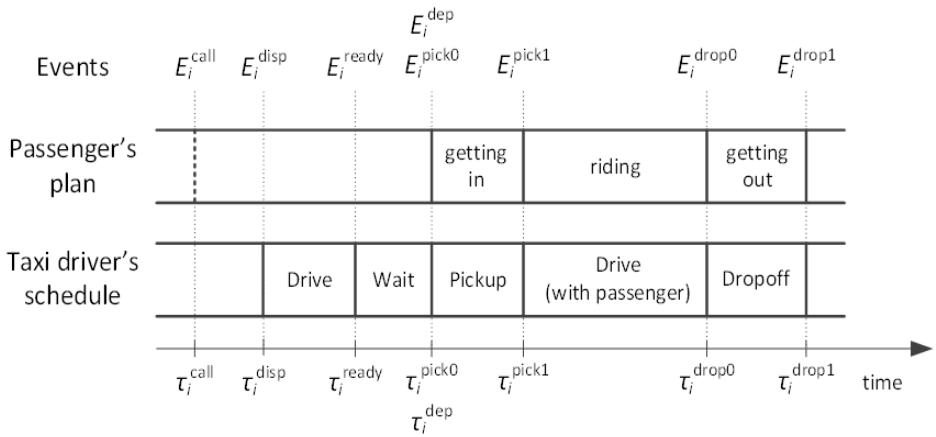

Fig. 2. A taxi driver's schedule and a passenger's plan (the taxi waits for the passenger)

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

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 $m+i$). As in [16], 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 $\tau_i^{\text{pick0}},\ i \in N$, 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, 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 instead of the time-dependent travel/serve time functions, $t_{ki}^{\text{O}}(t)$, $t_{ij}(t)$ and $t_i^S(t)$.

The offline taxi dispatching problem may be stated as:

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

subject to

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

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

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

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

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

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

The objective (1) minimizes 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 $a_k + t_{ki}^{\text{O}}$ 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 $t_i^{\text{S}} + t_{ij}$, 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 vehicles (subtour elimination constraints). Constraints (7) ensure that the pickup of customer *i* starts at time τ_i^{dep} or later.

5. Benchmarks

The MIP problem formulated in the previous section can be used in many different ways to approximate the minimum waiting time. This section describes two approaches and opposes them to three other benchmarks.

Full offline optimization

Given the full knowledge of the future, a complete (e.g. for a whole day) MIP problem instance is solved prior to simulation, resulting in an *a priori* approximation of the minimum total waiting time. Next the scenario is simulated using the pre-computed schedules, which gives an *a posteriori* approximation. Since the simulation is not deterministic (e.g. stochastic travel times), the timing of the a priori (pre-computed) schedules (determined using the average travel times) must be updated on the fly. In general, stochastic travel times make the a posteriori (simulated) passenger waiting times larger compared to those of the a priori solution (reaching p_i before τ_i^{dep} does not reduce the total waiting times, whereas the opposite does increase it). This issue could be fixed by repeatedly re-running the full optimization during simulation in response to incoming events. This, however, would imply prohibitively long computation times.

The MIP strategy

This online strategy consists in repeatedly running re-optimization in response to changes in the system. As this is considered an online strategy, the future requests are unknown and only the open ones are considered in scheduling. The formulation of the offline problem requires the destination locations to be provided in advance (i.e. during request submission).

To ensure that the strategy is responsive under high load, both the planning horizon and computation time are limited. The planning horizon is restricted to the first *h* requests in *L*. Whenever a new request

is included into the planning horizon, due to either E_i^{call} or E_i^{disp}, a respective offline problem is derived the current state and solved. A detailed description of this strategy may be found in [8, 10].

Simple heuristic strategies

The proposed benchmarks are compared with two simple heuristic online strategies, both described and analyzed in [10].

The first one, NOS (*no-scheduling*), dispatches the nearest idle vehicle $k^* \in M^{\text{I}}$ to the first request in the queue. The dispatch is carried out immediately after E_i^{call} or E_i^{drop1}; no advance assignments $\left(\tau_i^{\text{disp}} > \tau^{\text{curr}}\right)$ are made. By default, travel time is used to find the nearest taxi:

$$k^* = \arg\min_{k \in M^{\text{I}}} t^{\text{O}}_{k,L[1]}(a_k) \tag{8}$$

The second strategy, RES (*re-scheduling*), schedules requests to the closest available taxi $k^* \in M^{\text{A}}$, regardless of whether idle or busy. The scheduling, triggered by E_i^{call} or E_i^{pick1}, allows for both immediate and advance dispatches. By default, k^* is the taxi that arrives earliest at p_i:

$$k^* = \arg\min_{k \in M^{\text{A}}} a_k + t^{\text{O}}_{k,L[1]}(a_k) \tag{9}$$

In general, RES is expected to outperform NOS owing to a bigger choice set ($M^{\text{I}} \subseteq M^{\text{A}}$), especially under high load.

Idle vehicle teleportation

By moving idle vehicles towards the pickup locations of the (near) future requests, the total waiting time may be significantly reduced, even down to zero provided that the fleet is large enough. A good approximation of the lower bound for the minimum total passenger waiting time can be obtained by teleporting an idle taxi to a new customer. In this case, passengers wait only if all taxis are busy, and waiting requests are served according to their order in *L*. Theoretically, since travel times are time-dependent and stochastic, the total waiting time obtained with this strategy may be greater than the minimum total waiting times without teleporting. This is, however, very unlikely and in most cases this strategy produces rather over-optimistic (not tight) lower bounds.

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*, comprised m = 25 taxis.

To obtain reliable travel time estimates for simulating taxis, first the Mielec scenario was run without taxis for 20 iterations in order to reach a state of relaxation. Figures 3 shows a snapshot of the traffic state in the 20th iteration at 5:00 pm (afternoon peak hours), whereas Fig. 4 illustrates the resulting the scale of traffic over time. Next, after changing the mode of n trips from *private car* to *taxi*, which introduced 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 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. A branch-and-bound (BB) algorithm from the Gurobi Optimizer [5], version 5.6.2, was used to solve the offline MIP problem. Each time, first the initial feasible solution was calculated with RES, and then BB was run in parallel on all CPU cores. Based on the findings of [8], the planning horizon in the MIP strategy was the same as the number of taxis, h = 25, and the computation time of each optimization was limited to 60 seconds. For the full offline optimization, a 2-hour computation limit was assumed. In both cases, 24-hour travel time averages were used instead of the default 15-minute averages. The experiments were run on a computer with the 6-core Intel Core i7-3930K processor with Hyper-Threading Technology and 64 GB of RAM.

Fig. 3. Simulated traffic in Mielec at 5:00 pm (iteration 20) [10]

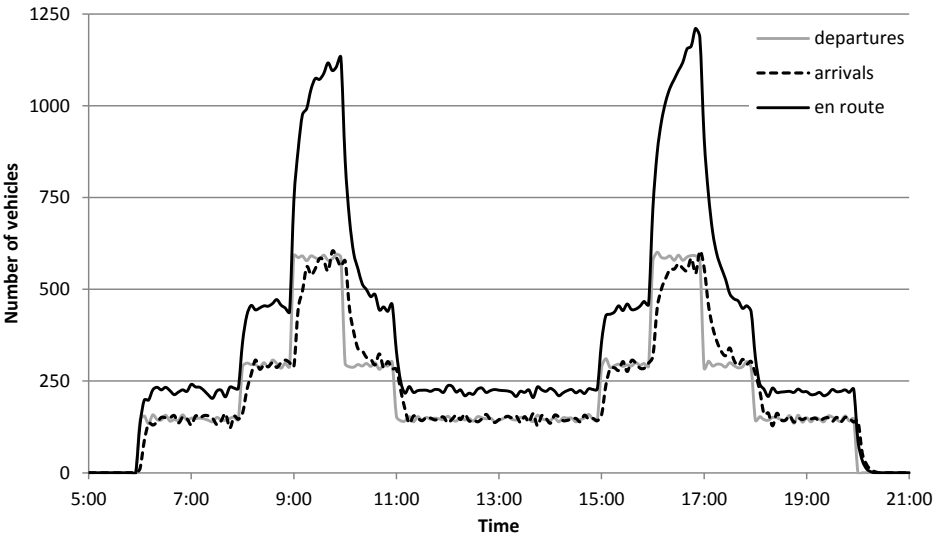

Fig. 4. Vehicle departures and arrivals (per 5 minutes) and vehicles en route [10]

7. Computation results

To provide mutual comparability of different experiments, instead of the total passenger waiting time, the average passenger waiting time, defined as

$$T_{\mathrm{W}} = \sum\nolimits_{i \in N} \left(\tau_i^{\mathrm{pick0}} - \tau_i^{\mathrm{dep}} \right) \Big/ n, \tag{10}$$

is referred to in this section.

Figure 5 presents the obtained average waiting times for different n and the following configurations:

– *offline a priori* – complete offline optimization with 2-hour computation time limit
– *offline a posteriori* – the results of the full offline optimization after fitting them into simulation
– *MIP* – the MIP strategy with the planning horizon limited to h = 25 taxis and 60-second computation time limit
– *NOS* – the no-scheduling strategy
– *RES* – the re-scheduling strategy; destinations unknown a priori
– *teleportation* – a NOS-like naïve strategy; idle taxis teleported to pickup locations

The results indicate that out of three online benchmarking strategies, namely NOS, RES and MIP, the third one performs best. Although the required a priori knowledge of destinations and long response times (not a real-time strategy) constrain the practical applications of MIP, it still may be successfully used as a yardstick for assessing the performance of different sophisticated real-time dispatching strategies.

Running one complete offline optimization before simulation starts has limited usability. First of all, the solver could not find the optimum within the time limit for the medium and high load scenarios (n > 800), or even improve the initial solution for the high load scenarios (n > 1200). Secondly, applying the obtained results in the stochastic simulation, without any possibility to react (due to long computation times) causes a significant performance degradation (especially under higher load where the schedule is very tight), which is depicted with the growing gap between *offline a priori* and *offline a posteriori* in Fig. 5.

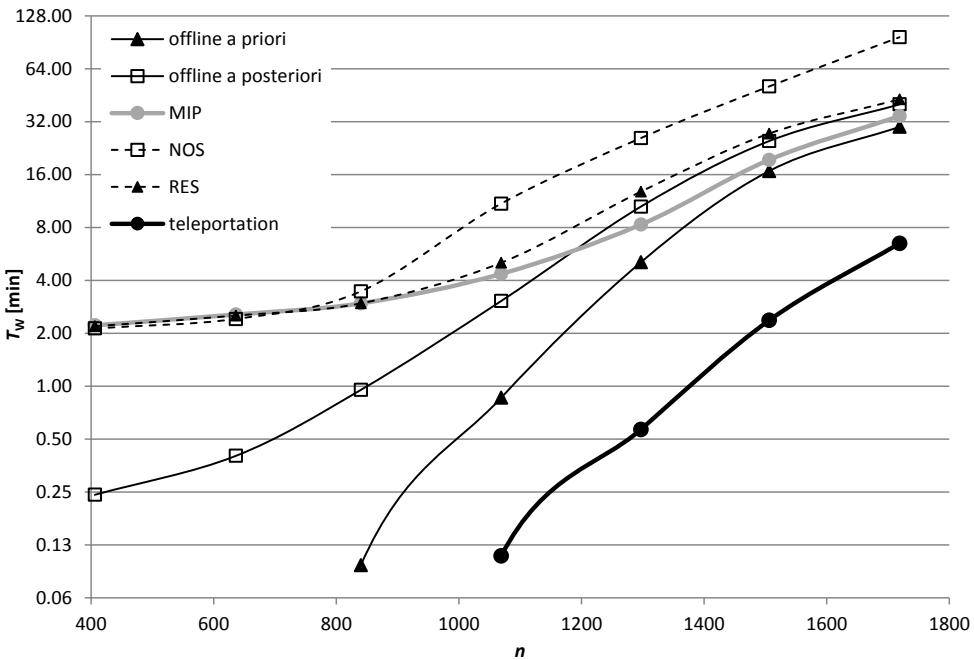

Fig. 5. Average waiting time, T_{W}, as a function of n

The results obtained for both the *offline a priori* and *teleportation* options indicate that even at medium loads ($800 < n \leq 1200$) there are periods with demand exceeding supply. This undersupply intensifies as n grows. When teleportation is enabled, the average waiting time for n = 1719 exceeds 6.5 minutes whereas the average dropoff trip time is slightly below 5.9 minutes. This illustrates the scale of undersupply.

8. Conclusion

Out of six different benchmarks, two approaches, namely *MIP* and *teleportation*, are particularly useful in approximating the upper and lower bounds for the minimal total passenger waiting time. Although the former uses dispatching strategy of limited applicability (due to long computation times) and the latter is purely hypothetical, both may be used for evaluating online taxi dispatching algorithms.

The MIP strategy can be enhanced by incorporating the knowledge of future requests. This would probably result in lower waiting times, at least under low load when the number of open request is usually lower than the length of the horizon. However, such a modified MIP strategy could not be used as an approximation of the upper bound for the optimal online algorithms.

The analysis of the results obtained for all the benchmarks indicates that in the case of low load, when vehicles remain idle for most of the time, any significant improvement in the total passenger waiting time is possible only through anticipating future request and moving idle vehicles towards them. On the other hand, under high load, the most important issue is providing the highest possible throughput, which can be done only by minimizing the total pickup time, so that more time is spent effectively on serving customers. However, under extremely high load, even the complete reduction of the pickup trip times via teleportation of idle vehicles would only minimize but not eliminate the undersupply.

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] Groetschel M., Krumke S., Rambau J., Winter T., Zimmermann U.: Combinatorial online optimization in real time. Online, 16, pp. 679–704, 2001.

[5] Gurobi Optimizer Reference Manual, Version 5.6, 2013.

[6] 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.

[7] Ma W., Wang K.: On the on-line weighted k-taxi problem. In: Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Springer, pp. 152–162, 2007.

[8] Maciejewski M.: Online taxi dispatching via exact offline optimization. Logistyka, 4/2014, pp. 2133–2142, 2014.

[9] 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.

[10] 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.

[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] Wong K. I., Bell M. G. H.: The optimal dispatching of taxis under congestion: A rolling horizon approach. Journal of Advanced Transportation, 40, pp. 203–220, 2006.
[16] Yang J., Jaillet P., Mahmassani H.: Real-time multivehicle truckload pickup and delivery problems. Transportation Science, 38(2), pp. 135–148, 2004.