View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Dagstuhl Research Online Publication Server

Fleet Management for Autonomous Vehicles Using Multicommodity Coupled Flows in Time-Expanded Networks

Sahar Bsaybes
Université Grenoble Alpes
Institute of Engineering (Grenoble INP), G-SCOP F-38000 Grenoble, France
sahar.bsaybes@grenoble-inp.fr

Alain Quilliot
Université Clermont Auvergne
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
BP 10125, 63173 Aubière Cedex, France
alain.quilliot@uca.fr

Annegret K. Wagler
Université Clermont Auvergne
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
BP 10125, 63173 Aubière Cedex, France
annegret.wagler@uca.fr

Abstract

VIPAFLEET is a framework to develop models and algorithms for managing a fleet of Individual Public Autonomous Vehicles (VIPA). We consider a homogeneous fleet of such vehicles distributed at specified stations in a closed site to supply internal transportation, where the vehicles can be used in different modes of circulation (tram mode, elevator mode, taxi mode). We treat in this paper a variant of the Online Pickup-and-Delivery Problem related to the taxi mode by means of multicommodity coupled flows in a time-expanded network and propose a corresponding integer linear programming formulation. This enables us to compute optimal offline solutions. However, to apply the well-known meta-strategy Replan to the online situation by solving a sequence of offline subproblems, the computation times turned out to be too long, so that we devise a heuristic approach h-Replan based on the flow formulation. Finally, we evaluate the performance of h-Replan in comparison with the optimal offline solution, both in terms of competitive analysis and computational experiments, showing that h-Replan computes reasonable solutions, so that it suits for the online situation.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases fleet management, offline and online pickup and delivery problem, multicommodity flows

Digital Object Identifier 10.4230/LIPIcs.SEA.2018.25

Funding This work was founded by the French National Research Agency, the European Commission (Feder funds) and the Région Auvergne in the Framework of the LabEx IMobS3.

© Sahar Bsaybes, Alain Quilliot, and Annegret K. Wagler;
licensed under Creative Commons License CC-BY

1 Introduction

The project VIPAFLEET aims at contributing to sustainable mobility through the development of innovative urban mobility solutions by means of fleets of Individual Public Autonomous Vehicles (VIPA) allowing passenger transport in closed sites like industrial areas, medical complexes, campuses, or airports. This innovative project involves different partners in order to ensure the reliability of the transportation system [3]. A VIPA is an autonomous vehicle that does not require a driver nor an infrastructure to operate. It is developed by Easymile and Ligier [1, 2] thanks to innovative computer vision guidance technologies [21, 22], whereas the fleet management aspect is studied in [9].

A fleet of VIPAs shall be used in a closed site to transport employees, customers and visitors e.g. between parkings, buildings and from or to a restaurant at lunch breaks. To supply internal transportation, a VIPA can operate in three different transportation modes:

- *Tram mode:* VIPAs continuously run on predefined cycles in a predefined direction and stop at a station if requested to let users enter or leave.
- *Elevator mode:* VIPAs run on predefined lines and move to stations to let users enter or leave, thereby changing their driving direction if needed.
- *Taxi mode:* VIPAs run on a connected network to serve transport requests (from any start to any destination station within given time windows).

This leads to a Pickup-and-Delivery Problem (PDP) in a metric space encoding the considered closed site, where a fleet of servers shall transport goods or persons from a certain origin to a certain destination. If persons have to be transported, we usually speak about a Dial-a-Ride Problem. Many variants are studied in the literature, including the Dial-a-Ride Problem with time windows [14, 15]. In our case, we are confronted with an online situation, where transport requests are released over time [5, 8, 13]. Problems of this type are known to be $\mathcal{NP}$-hard, see e.g. [20], which also applies to the problem variant considered here, see Section 2.

In [11], we focus on the economic aspect of the problem where the objective is to minimize costs; several algorithms are presented and evaluated w.r.t. minimizing the total tour length for tram and elevator mode.

The taxi mode is the most advanced circulation mode for VIPAs in the dynamic fleet management system. The transport requests are released over time (from any start to any destination station within a network G) and need to be served within a specified time window. Due to the time windows, it is not always possible to serve all transport requests (e.g., if more requests are specified for a same time window than VIPAs are available in the fleet). Hence, the studied PDP is an admission problem as it includes firstly to accept/reject requests and secondly to generate tours for the VIPAs to serve the accepted requests. We are confronted with both the quality-of-service aspect of the problem (with the goal to accept as many requests as possible) and the economic aspect (with the goal to serve the accepted requests at minimum costs, expressed in terms of minimizing the total tour length of the constructed tours), see Section 2.

In [10, 12], a variant of the PDP is studied where the tours are supposed to be nonpreemptive and at each time, (at most) one customer can be transported by a VIPA (note: one customer can be a group of people less than the capacity of the VIPA), and a VIPA cannot serve other requests until the current one is delivered. This leads to a load nonpreemptive DARP with time windows and server capacity 1, where the goal is to accept as many requests as possible and to find tours of minimal length to serve all accepted requests.

In order to solve this problem, three approaches are considered in [12]:

- a simple Earliest Pickup Heuristic that incrementally constructs tours by always choosing from the subsequence $\sigma(t')$ of currently waiting requests this request with smallest possible start time and appending it to the tour with shortest distance from its current end to the requested origin;
- the two well-known meta-strategies Replan and Ignore (which have been analysed in [4, 6, 7] for the Online Traveling Salesman Problem and can be applied to any online problem in time-stamp model [1], see e.g. [4, 6, 18, 23]) that determine which requests from $\sigma(t')$ can be accepted, and compute optimal (partial) tours to serve them, where Replan performs these tours until new requests are released, but Ignore completely performs these tours before it checks for newly released requests.

It turned out that Ignore is not suitable for the studied admission problem since the decision to accept/reject a request may be taken late, even after the time window to serve the request which does not comply to the quality-of-service aspect of the fleet management. Computational results from [12] show that Replan beats the Earliest Pickup Heuristic in terms of the number of accepted requests, but can only accept 64% of requests compared to the optimal offline solution.

This motivates to study another variant of the PDP related to the taxi mode without the requirement of constructing nonpreemptive tours and transporting, at each time, at most one customer in a VIPA on a direct way along a shortest path from its origin to its destination in the network G.

This problem variant is subject of the present paper. It leads to a more complex model and also computing solutions is more involved, but the expectation is to achieve a higher rate of accepted requests and, therefore, a better quality-of-service level for the fleet management.

It is natural to interprete the studied PDP by means of flows in a time-expanded version G_T of the original network G as, e.g., proposed by [17, 16, 19] for other variants of PDPs. In Section 3, we formulate the offline version of the problem as multicommodity coupled flows in the time-expanded network G_T, using one commodity per request coupled to the flow of VIPAs.

In order to solve the online version of the problem, we apply in Section 4 a Replan-like strategy that solves the online problem by computing a sequence of offline subproblems on certain subsequences of requests. (Recall that Ignore turned out to be not suitable for the studied admission problem, hence we focus here on Replan only.)

Computational experiments revealed that computing optimal offline solutions for the subproblems requires already long computation times, too long and thus not suitable for the online situation. However, we observed that only a small percentage of arcs in the time-expanded network G_T is used in the optimal solutions, so the idea is to reduce G_T to a network containing only arcs which are taken in the optimal solution with high probability, and then to compute the multicommodity coupled flows in the reduced network only. This leads to the flow-based heuristic h-Replan for the offline version of the studied problem.

[1] There are two common online paradigms, the sequence model and the time-stamp model, which differ in the way how information becomes available to the online algorithm: in the sequence model, the requests are given one by one and need to be served immediately and in this order, whereas in the time-stamp model, the requests become known over time at their release dates which allows the online algorithm to postpone and revoke decisions.

In Section 5, we evaluate the performance of h-Replan in comparison with the optimal offline solution both in theory (with the help of competitive analysis) and in practice (with the help of some computational results). We close with some concluding remarks on our approaches.

The results presented here were studied in [9].

2 Problem description and model

As proposed in [9, 11], we embed the VIPAFLEET management problem in the framework of a metric task system.

We encode the closed site where the VIPAFLEET system is running as a *metric space* $M = (V, d)$ induced by a connected network $G = (V, E)$, where the nodes correspond to stations, edges to their physical links in the closed site, and the distance d between two nodes $v_i, v_j \in V$ to the length of a shortest path from v_i to v_j in G. In V, we have a distinguished origin $v_o \in V$, the depot of the system, where all VIPAs are parked when the system is not running, i.e., outside a certain time horizon $[0, T]$.

An operator manages a fleet of k VIPAs each with a capacity for Cap passengers. The fleet management shall allow the operator to decide when and how to move the VIPAs in the network, and to assign requests to VIPAs. Hereby, any request r_j is defined as a 6-tuple $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ where

- $t_j \in [0, T]$ is the release date (i.e., the time when r_j becomes known),
- $x_j \in V$ is the origin node,
- $y_j \in V$ is the destination node,
- $p_j \in [0, T]$ is the earliest possible start time,
- $q_j \in [0, T]$ is the latest possible arrival time,
- z_j specifies the number of passengers,

where t_j, p_j, and q_j are certain discrete time points within $[0, T]$ that satisfy $t_j \leq p_j$, $p_j + d(x_j, y_j) \leq q_j$ and where $z_j \leq \text{Cap}$ needs to hold[2]. The operator monitors the evolution of the requests over time and

- decides which requests can be accepted (recall that some requests may have to be rejected if, e.g., more requests are specified for a same time window than VIPAs are available in the fleet), and
- creates tasks to serve accepted requests by moving the VIPAs to go to some station and to pickup, transport and deliver users.

More precisely, a *task* is defined by $\tau_j = (t_j, x_j, t_j^{pick}, y_j, t_j^{drop}, z_j)$. It is created by the operator in order to serve request $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ and is sent at time t_j to a VIPA indicating that z_j passengers have to be picked up at station x_j at time t_j^{pick} and delivered at station y_j at time t_j^{drop}, where $p_j \leq t_j^{pick} \leq q_j - d(x_j, y_j)$ and $p_j + d(x_j, y_j) \leq t_j^{drop} \leq q_j$ must hold.

In order to fulfill the tasks, the operator creates *tours* for the VIPAs. Each tour consists of *moves* from one station in G to another station in G and of *actions* to pickup and deliver passengers. Hereby, we require only that each move carries at most Cap many passengers. That means, we allow

- to serve several requests simultaneously by the same VIPA (as long as the capacity is respected),

[2] Note that a request r_j with $z_j > \text{Cap}$ can be replaced by $\lceil \frac{z_j}{\text{Cap}} \rceil$ many requests r'_j respecting the constraint $z'_j \leq \text{Cap}$.

- detours to stations not lying on a shortest path from the origin of one request to its destination in order to pickup or deliver passengers from other requests,
- vehicle preemption (i.e. that passengers have to change VIPAs on the way to their destination).

A *transportation schedule* for $(M, \mathcal{T})$ consists of a collection of tours $\{\Gamma^1, \ldots, \Gamma^k\}$ and is *feasible* when

- each of the k VIPAs has exactly one tour,
- each accepted request r_j is served within time window $[p_j, q_j]$,
- each tour starts and ends in the depot.

Our goal is to construct transportation schedules for the VIPAs operating in taxi mode respecting all the above constraints:

▶ **Problem 1** (Taxi Mode Problem $(M, \sigma, p, T, k, \mathsf{Cap})$ (TMP))**.** *Given a metric space $M = (V, d)$ induced by a connected network $G = (V, E)$, a sequence of requests σ, profits p for accepted requests, a time horizon $[0, T]$ and k VIPAs of capacity Cap, determine a maximum subset σ_A of accepted requests and find a feasible transportation schedule $\{\Gamma^1, \ldots, \Gamma^k\}$ of minimum total tour length to serve all requests in σ_A.*

Hereby, choosing sufficiently high profits and sufficiently small costs guarantees that indeed as many requests as possible are accepted, while small but positive costs ensure that unnecessary movements of VIPAs are avoided.

In order to solve the Offline TMP (Section 3), we propose to construct a time-expanded network G_T and compute multicommodity coupled flows in G_T.

In order to solve the Online TMP (Section 4), we propose the strategy h-Replan that considers at each moment in time t' the subsequence $\sigma(t')$ of currently waiting requests (i.e., already released but not yet served requests), determines which requests from $\sigma(t')$ can be accepted, and computes (partial) tours to serve them by multicommodity coupled flows in the reduced network related to $\sigma(t')$, performs these tours until new requests are released and recomputes $\sigma(t')$ and the tours (keeping already accepted requests).

3 Solving the Offline TMP

In order to solve the Offline TMP, we build a time-expanded network $G_T = (V_T, A_T)$ based on σ and the original network G. The node set V_T contains, for each station $v \in V$ and each discrete time point $t \in [0, T]$, a node $(v, t) \in V_T$ which represents station v at time t as a station where VIPAs can simply pass, pickup or deliver customers. The arc set $A_T = A_W \cup A_M$ is composed of

- wait arcs, from $(v, t) \in V_T$ to $(v, t + 1)$ with $t \in \{0, 1, \ldots, T - 1\}$ in A_W,
- transport arcs, from $(v, t) \in V_T$ to $(v', t + d(v, v'))$ for each edge (v, v') of G and each time point $t \in T$ with $t + d(v, v') \leq T$, in A_M.

On G_T, we define a VIPA flow F to encode the tour of the VIPAs through G_T. To correctly initialize the system, we use the nodes $(v_0, 0), (v_0, T) \in V_T$ as source and sink for the flow F and set the balance of the source accordingly to the number k of available vehicles, see (1b). For all internal nodes $(v, t) \in V_T \setminus \{(v_0, 0), (v_0, T)\}$, we use normal flow conservation constraints, see (1c), which also automatically ensure that a flow of value k is entering the sink (v_0, T).

In order to encode the routing of each request $r_j \in \sigma$ we consider $|\sigma|$ commodities $f_1 \cdots f_{|\sigma|}$. Each commodity f_j has a single source (x_j, p_j) where x_j is origin and p_j earliest pickup time of the request r_j, also referred to as the commodity's origin, a single sink (y_j, q_j) where y_j is destination and q_j latest possible delivery time of r_j, also referred to as the commodity's destination, and a quantity z_j which is the load of the request r_j that must be routed along a single path from its source to its sink. In order to avoid that a request is partially served by a vehicle, we require that the quantity to be routed by each commodity f_j is equal to z_j but $f_j \in \{0, 1\}$.

To ensure that a request can be rejected and is not served more than once, we require that for each f_j at most one outgoing arc from the commodity's origin is chosen, see (1d). We use normal flow conservation constraints, see (1e), which also automatically ensure that for each commodity f_j the flow leaving the commodity's origin equals the flow entering its destination.

To ensure that the capacity of the VIPA is respected on all arcs $a \in A_M$, we couple the flows by

$$\sum_{r_j \in \sigma} f_j(a) \cdot z_j \leq \text{Cap} \cdot F(a) \ \forall a \in A_M$$

such that the capacities for f_j on the transportation arcs are not given by constants but by a function. Note that due to these flow coupling constraints, the constraint matrix of the network is not totally unimodular (as in the case of uncoupled flows) and therefore integrality constraints for all flows are required (1h) and (1i), reflecting that solving the problem is $\mathcal{NP}$-hard.

Our objective function (1a) considers profits $p(j)$ on arcs $a \in \delta^-(x_j, p_j)$ for each commodity f_j to serve a request r_j, whereas all other arcs have zero profits. The costs correspond to the traveled distances $c(a) := d(u, v)$ on all arcs. The corresponding integer linear program is as follows:

$$\begin{aligned}
\max \ & \sum_{r_j \in \sigma} \sum_{a \in \delta^-(x_j, p_j)} p(j) f_j(a) - \sum_{a \in A_T} c(a) F(a) & & \text{(1a)} \\
\text{s.t.} \ & \sum_{a \in \delta^+(v_0, 0)} F(a) = k & & \text{(1b)} \\
& \sum_{a \in \delta^-(v,t)} F(a) = \sum_{a \in \delta^+(v,t)} F(a) & \forall (v,t) \neq (v_0, 0), (v_0, T) & \text{(1c)} \\
& \sum_{a \in \delta^-(x_j, p_j)} f_j(a) \leq 1 & \forall r_j \in \sigma & \text{(1d)} \\
& \sum_{a \in \delta^-(v,t)} f_j(a) = \sum_{a \in \delta^+(v,t)} f_j(a) & \forall r_j \in \sigma \forall (v,t) \neq (x_j, p_j), (y_j, q_j) & \text{(1e)} \\
& \sum_{r_j \in \sigma} f_j(a) \cdot z_j \leq \text{Cap} F(a) & \forall a \in A_M & \text{(1f)} \\
& F(a) \geq 0 & \forall a \in A_T & \text{(1g)} \\
& F(a) \in \mathbf{Z} & \forall a \in A_T & \text{(1h)} \\
& f_j(a) \in \{0, 1\} & \forall a \in A_T, \forall r_j \in \sigma & \text{(1i)}
\end{aligned}$$

where $\delta^-(v, t)$ denotes the set of outgoing arcs of (v, t), and $\delta^+(v, t)$ denotes the set of incoming arcs of (v, t).

The integer linear program (1) solves the Offline TMP (where the whole sequence σ of requests is known at time $t = 0$) to optimality:

▶ **Theorem 2.** *The integer linear program (1) provides an optimal solution of the Offline TMP.*

4 Solving the Online TMP

To handle the online situation, where the requests in σ are released over time during a time horizon $[0, T]$, we propose a heuristic to solve a sequence of offline subproblems for certain time intervals $[t', T']$ within $[0, T]$ on accordingly modified time-expanded networks. A usual replan strategy is based on computing the optimal solution on the subsequence $\sigma(t')$ of requests released in each replanning step. Computing an optimal solution by multicommodity coupled flows is generally very slow and, thus, not applicable in online situations. In the proposed algorithm h-Replan, we thus use a heuristic to compute offline solutions on $\sigma(t')$. As experiments have shown that only a small percentage of arcs in G_T is used in the optimal solution while solving the Offline TMP, the idea is to reduce G_T to a network $G_R(t')$ containing only arcs which are taken in the optimal solution with high probability. Afterwards, we solve the flow problem on this reduced network $G_R(t')$. This does not lead to a globally optimal solution, but provides reasonable solutions in short time.

▶ **Algorithm 3** (h-Replan).

```
Input: $(M, \sigma, p, T, k, Cap)$
Output: $\sigma_A$, and tours $\Gamma^1, \dots, \Gamma^k$
 1: initialize $t' = 0$, $\sigma_A = \emptyset$, $\sigma(t') = \{r_j \in \sigma : t_j = 0\}$, $\Gamma^i = (v_0, 0)$ for $1 \le i \le k$
 2: WHILE $t' < T$ DO:
        compute offline solution for $\sigma_A$, $\sigma(t')$, and $\Gamma^1, \dots, \Gamma^k$
        perform the (modified) tours until new requests become known
        update $t'$ and $\sigma(t')$
 3: return $\sigma_A$ and $\Gamma^1, \dots, \Gamma^k$
```

To compute those offline solutions for the subsequences $\sigma(t')$, we build a reduced time-expanded network $G_R(t')$ based on $\sigma_A, \sigma(t')$ and the original network G that has the possible start positions of the VIPAs as source nodes in V_+, internal nodes (v, t) for time points $t \in [t', T']$ relevant for the requests in $\sigma_A \cup \sigma(t')$, but far less arcs than G_T:

- To determine the possible source nodes in V_+ for the VIPAs from the current tours $\Gamma^1, \dots, \Gamma^k$, we proceed as follows. At the beginning, i.e. at time $t = 0$, we clearly have $(v_0, 0)$ as source for each VIPA. At any later time point t', we have: If a VIPA is currently serving a request r_j, then (y_j, t_j^{drop}) is its source; if a VIPA is currently idle and situated at v, then (v, t') is its source.
- To determine the internal nodes and arcs in $A'_M \subseteq A_M$ and $A'_W \subseteq A_W$ which are taken in the optimal solution with high probability, we compute classic multicommodity flows with adjusted profits and costs taking only the request commodities into account, but not coupled to a VIPA flow. The reason is that we intend to construct "interesting" paths for the request commodities, starting from the commodity's origin and ending at the commodity's destination, without taking the route of the VIPAs into consideration:
  - a min cost multicommodity flow in G_T to determine a shortest path for the commodity of each request r_j from (x_j, p_j) to (y_j, q_j),
  - a max profit multicommodity flow in G_T to determine for each request r_j a path from (x_j, p_j) to (y_j, q_j) that has the potential to partially share paths of other commodities.

  In both cases, the constraint matrices are totally unimodular such that the computations can be done in short time, see [9] for details. Besides using the arcs with positive flow from these two problems, we add further transport arcs from the destination of requests to reachable origins of other requests to ensure that requests can be served sequentially in one tour.

Thus, compared to the original time-expanded network $G_T = (V_T, A_T)$, we reduce in $G_R = (V'_T, A'_T)$ both the total number of nodes as well as of wait and transport arcs. We compute a transportation schedule by solving the max profit flow problem in $G_R(t')$ detailed in (2). Hereby, to keep previously accepted requests, we partition $\sigma(t')$ into the subsequences

- $\sigma_A(t')$ of previously accepted but until time t' not yet served requests and
- $\sigma_N(t') = \{r_j \in \sigma : t_j = t'\}$ of requests that are newly released at time t'.

$$\begin{aligned}
\max \quad & \sum_{r_j \in \sigma(t')} \sum_{a \in \delta^-(x_j,p_j)} p(a) f'_j(a) - \sum_{a \in A'_T} c(a) F'(a) && && \text{(2a)}\\
\text{s.t.} \quad & \sum_{a \in \delta^+(v,t)} F'(a) = k(v) && \forall (v,t) \in V_+ && \text{(2b)}\\
& \sum_{a \in \delta^-(v,t)} F'(a) = \sum_{a \in \delta^+(v,t)} F'(a) && \forall (v,t) \neq V_+, t < T' && \text{(2c)}\\
& \sum_{a \in \delta^-(x_j,p_j)} f'_j(a) \leq 1 && \forall r_j \in \sigma_N(t') && \text{(2d)}\\
& \sum_{a \in \delta^-(x_j,p_j)} f'_j(a) = 1 && \forall r_j \in \sigma_A(t') && \text{(2e)}\\
& \sum_{a \in \delta^-(v,t)} f'_j(a) = \sum_{a \in \delta^+(v,t)} f'_j(a) && \forall r_j \in \sigma, \forall (v,t) \neq (x_j,p_j),(y_j,q_j) && \text{(2f)}\\
& \sum_{r_j \in \sigma(t')} f'_j(a) \cdot z_j \leq \mathrm{Cap} F'(a) && \forall a \in A'_M && \text{(2g)}\\
& F'(a) \geq 0 && \forall a \in A'_T && \text{(2h)}\\
& F'(a) \in \mathbf{Z} && \forall a \in A'_T && \text{(2i)}\\
& f'_j(a) \in \{0,1\} && \forall a \in A'_T, \forall r_j \in \sigma(t') && \text{(2j)}
\end{aligned}$$

where $A'_T = A'_W \cup A'_M$ and $k(v)$ denotes the number of VIPAs initially situated in v. Constraints (2e) ensure that previously accepted requests are served whereas constraints (2d) allow to reject newly released requests.

From the computed flows F' and f'_j in the reduced network $G_R(t')$, it is again straightforward to determine newly accepted requests and to construct (partial) tours $\Gamma^1, \ldots, \Gamma^k$ for the VIPAs in the same way as for the offline situation.

5 Evaluation of online algorithms for the Online TMP

5.1 Competitive Analysis

It is standard to evaluate the quality of online algorithms with the help of competitive analysis. This can be viewed as a game between an online algorithm ALG and a malicious adversary who tries to generate a worst-case request sequence σ which maximizes the ratio between the online cost ALG(σ) and the optimal offline cost OPT(σ) knowing the entire request sequence σ in advance. ALG is called c-competitive for an online maximization problem if ALG produces for any request sequence σ a feasible solution with OPT(σ) $\leq c$ ALG(σ) for some given $c \leq 1$. The competitive ratio of ALG is the infimum over all c such that ALG is c-competitive.

In [12], we consider an *oblivious adversary* who knows the complete behavior of a (deterministic) online algorithm ALG and chooses a worst-case sequence for ALG. Hereby, an oblivious adversary is allowed to move VIPAs towards the origins x_j of not yet released requests r_j (but also has to respect the time windows $[p_j, q_j]$ to serve accepted requests r_j).

In [12], we showed that an oblivious adversary can force any (deterministic) online algorithm ALG for the Online TMP to reject all requests of a sequence while the adversary can accept and serve all requests, implying that ALG is not competitive.

Here, we consider a weaker adversary, called *non-abusive adversary*, who also knows the complete behavior of ALG and chooses a worst-case sequence for ALG, but is only allowed to move VIPAs towards origins (or destinations) of already released requests (and has also to respect the time windows).

We show that no (deterministic or non-deterministic) online algorithm ALG for the Online TMP is competitive against a non-abusive adversary, since the adversary can force ALG to accept at most one request and to reject all other requests of a sequence while the adversary can accept and serve all requests but one of the sequence.

▶ **Theorem 4.** *There is no competitive online algorithm for the Online TMP against a non-abusive adversary.*

Since the worst-case request sequence used to show the non-competitivity result is only based on the reachability of requests, but not on a particular strategy of an online algorithm, we conclude:

▶ **Corollary 5.** *The online algorithm h-Replan is not competitive for the Online TMP against an oblivious or non-abusive adversary.*

5.2 Computational Results

This section deals with computational experiments for the optimal offline solutions of the (non-preemptive and preemptive) TMP and the two replan strategies, Replan studied for the non-preemptive case in [10, 12] and h-Replan proposed here for the preemptive case of the Online TMP. In fact, due to the very special request structures of the worst-case instances to prove the non-competitivity of any online algorithm for the Online TMP, we can expect a better behavior of the proposed replan strategies for the Online TMP in average.

The computational results presented in this section support this expectation. They compare the total number of accepted (and thus served) requests by Replan and h-Replan with the optimal offline solutions OPT-NP for the non-preemptive case and OPT-P for the preemptive case. The computations use randomly generated instances with 20 stations, 5 to 10 VIPAs, time-horizons between 180 and 240 time units, and between 90 and 300 customer requests. These instances are based on the network from the industrial site of Michelin at Clermont-Ferrand and randomly generated request sequences resembling typical instances that occurred during an experimentation in Clermont-Ferrand performed from October 2015 until February 2016 [21].

The operating system for all tests is Linux CentOS with kernel version 2.6.32 clocked at 2.40 GHz, with 1 TB RAM. The approaches are implemented in Python and Gurobi 8.21 is used for solving the ILPs.

In the first resp. second set of 180 instances each, the requests have a random load between

- 4 and 10 (with 72% of the requests with a load above 5),
- 1 and 10 (with only 21% of the requests with a load above 5),

Table 1 This table shows the percentage of improvement of the average number of accepted requests between the non-preemptive and preemptive optimal solutions and between Replan and h-Replan for the first set of instances.

req	T	k	OPT-NP	UB(OPT-P)	Imp (%)	Replan	h-Replan	Imp (%)
94	180	10	77	82,8	7,53	39	41,8	7,18
188	180	10	112	121,08	8,11	55	59,12	7,49
295	180	10	146,86	160,54	9,31	75,85	90,8	19,71
97	240	5	62,04	66,48	7,16	25,19	29,7	17,90
194	240	5	93,76	104,34	11,28	45,84	51,2	11,69
290	240	5	115,94	129,22	11,45	47,64	54,4	14,19

Table 2 This table shows the percentage of improvement of the average number of accepted requests between OPT-NP and OPT-P and between Replan and h-Replan for the second set of instances.

req	T	k	OPT-NP	UB(OPT-P)	Imp (%)	Replan	h-Replan	Imp (%)
94	180	10	65,31	86,70	32,75	36,54	47,54	30,10
180	180	10	107,48	158,65	47,61	47,16	77,80	64,97
295	180	10	153,20	283,50	85,05	79,14	124,10	56,81
97	240	5	61,76	84,40	36,66	24,10	32,50	34,85
194	240	5	100,32	154,23	53,74	45,38	72,67	60,14
290	240	5	123,67	275,47	122,74	46,21	88,50	91,52

and in both cases VIPAs of capacity 10. The two replan strategies compute solutions within a reasonablely short time (even for hReplan in less than 60 seconds in average for each replanning step).

As already reported in [10, 12], Replan achieves in average an acceptance rate of about 64% compared with OPT-NP for the first set of instances, and about 45% for the second. Our interest is whether or not allowing preemptive tours can significantly improve this acceptance rate.

Unfortunately, due to the long computation time for OPT-P, only an upper bound UB can be presented for most cases, obtained by computing an uncapacitated preemptive TMP with a time limit of four hours. Thus, we can mainly compare the improvements of the acceptance rate for OPT-NP and h-Replan only with this upper bound.

In the first set of instances, we observe that the percentage of improvement between OPT-NP and the upper bound of OPT-P is high (in average around 41% compared to UB), but it is not the case for the percentage of improvement between Replan and h-Replan (in average around 13%), see Table 1. The reason why there is no remarkable improvement in the acceptance rate is that in 72% of the requests, the load is greater than Cap/2 such that, in most of the times, the requests cannot be accumulated together to be served in one VIPA (recall that we allow vehicle preemption but not load preemption).

This changes in the second set of instances with in general smaller loads that allow us to serve more than one request simultaneously in one VIPA. Accordingly, we observe that the percentage of improvement between OPT-NP and OPT-P/UB increases to in average around 43% and between Replan and h-Replan to in average around 57%, (see Table 2). Detailed computational results are summarized in Table 3 and Table 4.

Note that computational results presented in Table 3 and Table 4 show only an upper bound for OPT-P. Therefore, this upper bound is sometimes far from the optimal solution

Table 3 This table shows the computational results for the first set of 180 test instances of Replan respectively h-Replan in comparison to OPT-NP respectively to the upper bound of the optimal preemptive offline solution UB(OPT-P). The instances are grouped by the number of requests (1st column), the time horizon (2nd column) and the number of VIPAs (3rd column) with 30 instances per parameter set. Average values are shown for the total number $|\sigma_A|$ of accepted requests and for the total tour length TTL needed to serve the accepted requests. Finally, we provide the average runtime of Replan respectively h-Replan per recomputation step and the maximum runtime of the recomputation steps of Replan respectively h-Replan.

non-preemptive TMP									
			$\vert\sigma_A\vert$			TTL		runtime (s)	
req	T	k	OPT-NP	Replan	ratio %	OPT-NP	Replan	AVG	MAX
94	180	10	77	52,13	67,7	667,5	424	0,49	1,6
188	180	10	112	70,45	62,9	831	580	3	10,83
295	180	10	146,86	97,2	66,19	1005	750,57	13,56	45,54
97	240	5	62,04	39,19	63,17	527,16	298,82	0,29	1,23
194	240	5	93,76	55,84	59,56	680,44	490	1,8	7,85
290	240	5	115,94	80,64	69,55	759,94	500,6	7,18	29,8

preemptive TMP									
			$\vert\sigma_A\vert$			TTL		runtime (s)	
req	T	k	UB(OPT-P)	h-Replan	ratio %	OPT-P	h-Replan	AVG	MAX
94	180	10	83,72	62,21	74,31	678,1	456,54	2,18	9,76
188	180	10	150,08	76,17	50,75	875,43	600,62	6,29	38,77
295	180	10	232	110,54	47,65	1167,54	748,8	21,82	68,54
97	240	5	73,34	42,4	57,81	574,65	322,75	1,34	13,06
194	240	5	134,74	63,83	47,37	885,48	515,5	3,91	24,54
290	240	5	210,63	90,23	42,84	998,75	496,6	19,62	58,39

especially in the first set of instances, where the improvement between OPT-NP and the upper bound of OPT-P is high 41% due to the upper bound calculated by computing an uncapacitated preemptive TMP. Thus, the requests can be accumulated together, without consireding their loads. This cannot be the case in the optimal solution. Note that in the first set of instance, the average acceptance ratio between Replan and OPT-NP is 65% while the average acceptance ratio between h-Replan and OPT-P/UB is 54% while in the second set of instance the average acceptance ratio between Replan and OPT-NP is 46% while the average acceptance ratio between h-Replan and OPT-P/UB is 43%. While Replan is not competitive in not competitve in theory, in practice it achieves a ratio about 2 compared to the optimal offline solution or the upper bound which is an acceptable ratio from a business point of view.

The computations in Table 5 use randomly generated instances with 10 stations, 2 to 3 VIPAs with capacity 10, time-horizon of 60 time units, and between 20 and 30 customer requests. In this set of 120 instances, the requests have a random load between

- 1 and 10 (with only 28% of the requests with a load above 5),

The two replan strategies compute solutions within a short time (less than 5 seconds in average) for each replanning step, therefore the average runtime is not shown in Table 5.

Table 4 This table shows the computational results for the second set of instances.

non-preemptive TMP											
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	OPT-NP	Replan	ratio %	OPT-NP	Replan	OPT-NP	AVG	MAX
94	180	10	65,31	36,54	55,95	667,5	416,7	12,6	0,68	5,32
188	180	10	107,48	47,16	43,888	831	596,35	181,23	2,69	12,45
295	180	10	153,2	79,14	51,66	1005	726,86	73456,5	12,67	27,42
97	240	5	61,76	24,1	39,02	527,16	279,15	697,75	0,86	6,45
194	240	5	100,32	45,38	45,24	680,44	504,7	846,5	3,7	14,6
290	240	5	123,67	46,21	37,37	759,94	527,45	116875,85	14,1	22,46

preemptive TMP										
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	UB	h-Replan	ratio %	UB	h-Replan	UB	AVG	MAX
94	180	10	86,70	47,54	54,83	727,56	460,16	43824,50	2,58	11,58
188	180	10	158,65	77,80	49,04	930,75	649,67	125849,75	7,42	45,45
295	180	10	283(UB)	124,10	43,77	1175,25	878,25	97849(UB)	26,45	82,42
97	240	5	84,40	32,50	38,51	558,45	358,75	90470,67	1,36	14,36
194	240	5	154,23	72,67	47,12	825,74	609,40	156752,58	4,26	27,42
290	240	5	275(UB)	88,50	32,13	957,52	630,86	128417(UB)	21,78	65,80

In this set of instances, Replan achieves in average an acceptance rate of about 54% compared with OPT-NP, and h-Replan achieves in average an acceptance rate of about 70% compared with OPT-P (see Table 5).

6 Conclusion

Regarding the quality of the solutions obtained by the here proposed h-Replan strategy for the Online TMP, we summarize that

- in theory, h-Replan is (as any other online algorithm for the problem) not competitive since there is no finite c s.t. for all instances σ we have that $\mathrm{OPT}(\sigma) \leq c$ h-Replan(σ), but
- in practice, h-Replan leads to a higher rate of accepted requests and, therefore, to a higher quality-of-service level for the fleet management than Replan constructing non-preemptive tours.

However, sometimes the transportation schedule returned by h-Replan contains preemptive tours which causes inconveniences for the users. Therefore, in order to handle the Online Taxi Mode Problem in the studied VIPAFLEET management system it is up to the operator to decide whether it is worth to have preemptive tours in order to increase the number of accepted requests, taking the ratio of request loads and VIPA capacities and, thus, the expected increase of the acceptance rate into account.

Table 5 This table shows the computational results for the first set of 120 test instances of Replan respectively h-Replan in comparison to OPT-NP respectively to the optimal preemptive offline solution OPT-P. The instances are grouped by the number of requests (1st column), the time horizon (2nd column) and the number of VIPAs (3rd column) with 30 instances per parameter set. Average values are shown for the total number $|\sigma_A|$ of accepted requests and for the total tour length TTL needed to serve the accepted requests.

			NP-TaxiMP				
			$\|\sigma_A\|$			TTL	
req	T	k	OPT-NP	Replan-NP	ratio %	OPT-NP	R-NP
18	60	2	11,23	6,58	58,59	95,7	52,85
26	60	2	16,42	8,43	51,34	118,46	65,17
17	60	3	13,57	8,1	59,69	124,78	71,63
28	60	3	20,32	9,74	47,93	138,6	76,25
			P-TaxiMP				
			$\|\sigma_A\|$			TTL	
req	T	k	OPT-P	hReplan-P	ratio %	OPT-P	hReplan-P
18	60	2	14,43	9,64	66,81	90,37	85,83
26	60	2	22,73	16,61	73,08	103,54	91,37
17	60	3	15,72	10,95	69,66	111,35	92,61
28	60	3	25,43	18,13	71,29	123,74	105,75

References

1 Easymile, 2015. URL: `http://www.easymile.com`.

2 Ligier group, 2015. URL: `http://www.ligier.fr`.

3 Viaméca, 2015. URL: `http://www.viameca.fr/`.

4 Norbert Ascheuer, Sven O Krumke, and Jörg Rambau. *The online transportation problem: competitive scheduling of elevators*. ZIB, 1998.

5 Norbert Ascheuer, Sven O Krumke, and Jörg Rambau. Online dial-a-ride problems: Minimizing the completion time. In *STACS 2000*, pages 639–650. Springer, 2000.

6 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Competitive algorithms for the on-line traveling salesman. In *Workshop on Algorithms and Data Structures*, pages 206–217. Springer, 1995.

7 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman 1. *Algorithmica*, 29(4):560–581, 2001.

8 Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European journal of operational research*, 202(1):8–15, 2010.

9 Sahar Bsaybes. *Modèles et algorithmes de gestion de flottes de véhicules VIPA*. PhD thesis, Université Clermont Auvergne, 2017.

10 Sahar Bsaybes, Alain Quilliot, and Annegret K Wagler. Fleet management for autonomous vehicles using flows in time-expanded networks. *Electronic Notes in Discrete Mathematics*, 62:255–260, 2017.

11 Sahar Bsaybes, Alain Quilliot, and Annegret K Wagler. Fleet management for autonomous vehicles: Online PDP under special constraints. *to appear on RAIRO - Operations Research*, 2018.

12 Sahar Bsaybes, Alain Quilliot, and Annegret K Wagler. Fleet management for autonomous vehicles using flows in time-expanded networks. *to appear on Journal of Advanced Transportation*, 2018.

13 Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

14 Samuel Deleplanque and Alain Quilliot. Transfers in the on-demand transportation: the DARPT Dial-a-Ride Problem with transfers allowed. In *Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA)*, pages 185–205, 2013.

15 Anke Fabri and Peter Recht. Online dial-a-ride problem with time windows: an exact algorithm using status vectors. In *Operations Research Proceedings 2006*, pages 445–450. Springer, 2007.

16 Lester R Ford Jr and Delbert Ray Fulkerson. Constructing maximal dynamic flows from static flows. *Operations research*, 6(3):419–433, 1958.

17 Martin Groß and Martin Skutella. Generalized maximum flows over time. In *International Workshop on Approximation and Online Algorithms*, pages 247–260. Springer, 2011.

18 Martin Grötschel, Sven O Krumke, Jörg Rambau, Thomas Winter, and Uwe T Zimmermann. Combinatorial online optimization in real time. In *Online optimization of large scale systems*, pages 679–704. Springer, 2001.

19 Ronald Koch, Ebrahim Nasrabadi, and Martin Skutella. Continuous and discrete flows over time. *Mathematical Methods of Operations Research*, 73(3):301, 2011.

20 Jan Karel Lenstra and AHG Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

21 E Royer, F Marmoiton, S Alizon, D Ramadasan, M Slade, A Nizard, M Dhome, B Thuilot, and F Bonjean. Retour d'expérience après plus de 1000 km en navette sans conducteur guidée par vision.

22 Eric Royer, Jonathan Bom, Michel Dhome, Benoit Thuilot, Maxime Lhuillier, and François Marmoiton. Outdoor autonomous navigation using monocular vision. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1253–1258. IEEE, 2005.

23 Jian Yang, Patrick Jaillet, and Hani Mahmassani. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148, 2004.