# Fleet management for autonomous vehicles using flows in time-expanded networks

Sahar Bsaybes, Alain Quilliot, Annegret K Wagler

# Fleet management for autonomous vehicles using flows in time-expanded networks

Sahar Bsaybes[1a], Alain Quilliot[b], Annegret K. Wagler[b]

[a] *Université Grenoble Alpes, CNRS, Grenoble INP[2], G-SCOP, F-38000 Grenoble, France*
[b] *Université Clermont Auvergne (LIMOS UMR CNRS 6158), Clermont-Ferrand, France*

**Abstract**

The VIPAFLEET project aims at developing a framework to manage a fleet of Individual Public Autonomous Vehicles (VIPA). We consider a fleet of cars distributed at specified stations in an industrial area to supply internal transportation, where the cars can be used in different modes of circulation (tram mode, elevator mode, taxi mode). We treat in this paper the pickup and delivery problem related to the taxi mode by means of flows in time-expanded networks. This enables us to compute optimal offline solutions, to propose strategies for the online situation, and to evaluate their performance in comparison with the optimal offline solution.

*Key words:* fleet management, offline and online pickup and delivery problem

## 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 [20]. A VIPA is an autonomous vehicle that does not require a driver nor an infrastructure to operate, it is developed by Easymile and Ligier [18, 19] thanks to innovative computer vision guidance technologies [25, 24], whereas the fleet management aspect is studied in [9]. A fleet of VIPAs shall be used in a closed site to transport employees and visitors e.g. between parkings, buildings and from or to a restaurant. The fleet is distributed at specified stations within the site. To supply internal transportation, a VIPA can operate in three different circulation modes:

- *Tram mode:* VIPAs continuously run on predefined lines or 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 react to requests by moving to a station to let users enter or leave, thereby changing their driving direction if needed.

[1] 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.

[2] Institute of Engineering Univ. Grenoble Alpes

*Email addresses:* sahar.bsaybes@grenoble-inp.fr (Sahar Bsaybes), alain.quilliot@uca.fr (Alain Quilliot), annegret.wagler@uca.fr (Annegret K. Wagler)

- *Taxi mode:* VIPAs run on a connected network to serve transport requests (from any start to any destination station in the network within given time windows).

This leads to a Pickup-and-Delivery Problem (PDP) under special constraints 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 [13, 14]. In our case, we are confronted with an online situation, where the transport requests are released over time [3, 6, 12]. Problems of this type are known to be $\mathcal{NP}$-hard, see e.g. [23], which also applies to the problem variant considered here, see Section 2 for details.

In [10], 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 the tram and elevator mode that handle the requests coming online, solve a PDP, and generate tours for the VIPAs in order to serve the transport requests.

In this paper, we treat the PDP related to the taxi mode as the most advanced circulation mode for VIPAs in the dynamic fleet management system[11]. The transport requests are released over time and need to be served within a specified time window. We consider the case that, at each time, at most one customer can be transported by a VIPA (where 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. Note that, due to the time windows and the above additional restrictions, it is not always possible to serve all transport requests. Hence, the studied PDP includes firstly to accept/reject requests and secondly to generate tours for the VIPAs to serve the accepted requests. Thus, we treat here 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 again Section 2 for details.

In Section 3, we provide a solution approach for the studied PDP by means of flows in time-expanded networks as, e.g., proposed by [15, 16, 22] for other variants of PDPs. Hereby, we need to distinguish between the online and the offline version of the problem: the online version occurs in practice (since the transport requests become known over time), whereas the offline version is important in theory to rate the quality of solutions for the online problem, by comparison with the optimal offline solution (computed knowing the entire request sequence already in advance).

In Section 3.1, we present a way to compute optimal offline solutions for the PDP related to the taxi mode. In Section 3.2, we consider three approaches: besides a simple heuristic, we apply the two well-known meta-strategies Replan and Ignore (which have been analysed in [2, 4, 5] for the Online Traveling Salesman Problem and can be applied to any online problem in time-stamp model, see e.g. [2, 4, 17, 26]). Both solve the online version of the PDP by computing a sequence of offline subproblems on certain subsequences of requests.

In Section 4, we evaluate the performance of the proposed strategies in comparison with the optimal offline solution both in theory (with the help of competitive analysis, see Section 4.1), and in practice (with the help of some computational results, see Section 4.2). We close with some concluding remarks on our approaches and some future lines of research.

Parts of the here presented results are taken from [9, 11].

## 2. Problem Description and Model

As proposed in [9], 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 pickup time,
- $q_j \in [0, T]$ is the latest possible delivery 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 $z_j \leq$ Cap needs to be satisfied [4]. The operator monitors the evolution of the requests over time and

- decides which requests can be accepted (note 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 some station 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 an accepted 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.

We denote by $\mathcal{T}_A$ the set of tasks generated by the operator in order to serve the accepted requests.

In order to fulfill the tasks, the operator creates *tours* for the VIPAs. Each tour consists of *moves* from one station in $V$ to another station in $V$ and of *actions* to pickup and deliver passengers. Hereby, we require that

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

- each move carries at most one request (to fulfill the idea of a transportation by taxi, not shared with any other customer),
- all $z_j$ passengers of one request $r_j$ are transported by the same VIPA in a direct way, i.e., on a shortest path from $x_j$ to $y_j$.

That means, we do not allow load preemption, but we are interested in serving each request in a load non-preemptive manner. That way, the capacity of a VIPA is respected in any move due to the constraint $z_j \leq \text{Cap}$ for all $r_j \in \sigma$.

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

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

Our goal is to construct transportation schedules $S$ for the VIPAs operating in taxi mode respecting all the above constraints with the objective to accept as many requests as possible and to serve the accepted requests at minimum costs. This leads to the following problem:

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

Note that the Taxi Mode Problem is always feasible (since rejecting all requests is a solution).

## 3. Solving the Taxi Mode Problem

The input for the Online or Offline Taxi Mode Problem $(M, \sigma, p, T, k, \text{Cap})$ consists of the following data:

- a weighted graph $G = (V, E, w)$ where the nodes correspond to stations, edges to their links, and edge weights $w : E \to \mathbb{R}_+$ determine the driving times between two neighbored stations $u, v \in V$,
- a sequence $\sigma = \{r_1, \ldots, r_h\}$ of requests $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ with $t_j \leq p_j$, $p_j + d(x_j, y_j) \leq q_j$, as well as $z_j \leq \text{Cap}$,
- per request a profit $p(r_j)$ for serving the request $r_j$,
- a time horizon $[0, T]$,
- the total number $k$ of VIPAs, and the capacity Cap of the VIPAs as the maximum number of passengers which can be simultaneously transported in one VIPA.

The output of the Online or Offline Taxi Mode Problem is the decision to accept/reject the requests (in terms of a subset $\sigma_A \subseteq \sigma$ of accepted requests) and a feasible transportation schedule $S$ serving all accepted requests.

The goal is to accept as many requests as possible and to serve them at minimum costs by a transportation schedule $S = \{\Gamma^1, \ldots, \Gamma^k\}$ of minimum total tour length.

Hereby, choosing sufficiently high profits, e.g. $p(r_j) > \text{diam}(G) d(x_j, y_j)$ with $\text{diam}(G)$ diameter of the network $G$, and sufficiently small costs, e.g. $c(a)$ equal to its length in $G$, guarantees

that indeed as many requests as possible are accepted, while small but positive costs ensure that unnecessary movements of VIPAs are avoided[5].

In order to solve the Online TMP, three approaches are considered:

- a simple Earliest Pickup Heuristic that incrementally constructs tours by always choosing from the subsequence $\sigma(t')$ of currently waiting requests (i.e., already released but not yet served 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 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.

Therefore, as applying a Replan or Ignore strategy to the Online TMP involves the computation of (partial) optimal offline solutions (i.e. an optimal solution under the condition that the whole sequence of requests is known in advance), we start with considerations on solving the Offline TMP (see Section 3.1).

### *3.1. Solving the Offline Taxi Mode Problem*

In order to solve the Offline TMP, we build a time-expanded request network $G_T = (V_T, A_T)$ based on $\sigma$ and the original network $G$.
The node set $V_T = V_+ \cup V_x \cup V_y \cup V_-$ is composed of

- the nodes $(v_0, 0) \in V_+$ as source and $(v_0, T) \in V_-$ as sink that correspond to the depot at the beginning and at the end of the time horizon,
- all possible origins $(x_j, t_j^{pick})$ of all requests $r_j$ in $\sigma$, for all $p_j \leq t_j^{pick} \leq q_j - d(x_j, y_j)$ in $V_x$,
- all possible destinations $(y_j, t_j^{drop})$ of all $r_j$ in $\sigma$, for all $p_j + d(x_j, y_j) \leq t_j^{drop} \leq q_j)$ in $V_y$.

The arc set $A_T = A_+ \cup A_R \cup A_L \cup A_-$ is composed of

- source arcs from $(v_0, 0)$ to all reachable origins $(x_j, t_j^{pick})$ in $V_x$ with $d(v_0, x_j) \leq t_j^{pick}$ in $A_+$,
- request arcs from each $(x_j, t_j^{pick}) \in V_x$ to $(y_j, t_j^{pick} + d(x_j, y_j)) \in V_y$ in $A_R$,
- link arcs from all destinations $(y_j, t_j^{drop}) \in V_y$ to all reachable origins $(x_i, t_i^{pick}) \in V_x$ with $t_j^{drop} + d(y_j, x_i) \leq t_i^{pick}$ in $A_L$,
- sink arcs from all destinations in $V_y$ to $(v_0, T)$ in $A_-$.

Note, that the time-expanded network $G_T$ is acyclic by construction.

The VIPAs shall form a flow $f$ through this time-expanded request network $G_T$. To correctly initialize the system, we use the node $(v_0, 0) \in V_+$ as source for the flow $f$ and set its balance 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 ensures that a flow of value $k$ is entering the sink $(v_0, T) \in V_-$.

[5]Note that it is even possible to use different profits for requests of different users to ensure e.g. that the requests of disabled people are accepted and served in any case.

A request arc from $(x_j, t_j^{pick})$ to $(y_j, t_j^{pick} + d(x_j, y_j))$ has a capacity 1 for the VIPA flow. We distinguish $|\sigma|$ subsets $A_R^j$ of arcs in $A_R$ where each subset $A_R^j$ consists of the request arcs of the corresponding request $r_j$, so that we have $A_R = \bigcup_{j=1}^{|\sigma|} A_R^j$. To ensure that a request can be rejected and is not served more than once, we require that the sum of the flow traversing all the request arcs in $A_R^j$ of the corresponding request $r_j$ is at most 1, see (1d). By the previous constraints, the flow $f$ is (automatically) bounded on all other arcs by 1.

Note that source, flow conservation and nonnegativity constraints (1b), (1c), (1e) together give rise to a totally unimodular matrix (in fact, to the node/arc incidence matrix of the digraph underlying the network $G_T$), but due to the inequalities (1d) the entire constraint matrix is not totally unimodular s.t. integrality constraints (1f) are required (to prevent fractional solutions).

We consider a max-profit flow problem to decide which requests can be served. Accordingly, our objective function (1a) considers profits $p(a)$ for the flow $f$ on all $a \in A_R$ whereas all other arcs $a = ((u, t), (v, t + d(u, v)))$ 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:

$$\max \sum_{a \in A_R} p(a)f(a) - \sum_{a \in A_T} c(a)f(a) \tag{1a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v_0,0)} f(a) = k \tag{1b}$$

$$\sum_{a \in \delta^-(v,t)} f(a) = \sum_{a \in \delta^+(v,t)} f(a) \qquad \forall (v,t) \notin V_+ \cup V_- \tag{1c}$$

$$\sum_{a \in A_R^j} f(a) \leq 1 \qquad \forall A_R^j \in A_R \tag{1d}$$

$$f(a) \geq 0 \qquad \forall a \in A_T \tag{1e}$$

$$f(a) \in \mathbf{Z} \qquad \forall a \in A_T \tag{1f}$$

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 above integer linear program solves the offline version of the Taxi Mode Problem (where the whole sequence $\sigma$ of requests is known at time $t = 0$) to optimality.

**Theorem 1.** *The integer linear program (1) provides an optimal solution of the Offline Taxi Mode Problem.*

*Proof.* Let $f^*$ be the optimal flow according to (1). Accepted requests clearly correspond to request arcs $a \in A_R$ with $f^*(a) = 1$ so that we have

$$\sigma_A = \{r_j \in \sigma : f^*(a) = 1 \text{ for one } a \in A_R^j\}.$$

Moreover, it is clear that accepted requests are indeed served. The computed flow $f^*$ in the time-expanded request network $G_T$ can be interpreted as transportation schedule, since we can recover the tracks of the $k$ VIPAs over time from the flow $f^*$ on the arcs $a \in A_T$ with $f^*(a) > 0$ by standard flow decomposition as in [1]. In our case, the correspondence between the flow in $G_T$ and the moves in the VIPA tours $\Gamma^i$ is particularly easy to see, because constraints (1d) together

with the flow conservation constraints (1c) imply also for the flow on all source, link and sink arcs an upper bound of 1 so that clearly $f^*(a) \in \{0, 1\}$ holds for all $a \in A_T$. Therefore, a flow of 1 on

- a source arc $a \in A_+$ from $(v_0, 0)$ to an origin $(x_j, t_j^{pick}) \in V_x$ means that one VIPA starts its tour with a move along a shortest path from $v_0$ to $x_j$ and performs a pickup action at $x_j$;
- a request arc $a \in A_R$ from an origin $(x_j, t_j^{pick})$ in $V_x$ to its destination $(y_j, t_j^{pick} + d(x_j, y_j))$ means that request $r_j$ is served by a move of one VIPA along a shortest path from $x_j$ to $y_j$ and that a drop action is performed at $y_j$;
- a link arc $a \in A_L$ from a destination $(y_j, t_j^{drop})$ in $V_y$ to an origin $(x_i, t_i^{pick})$ in $V_x$ means that the VIPA continues its tour by a move along a shortest path from $y_j$ to $x_i$ and performs a pickup action at $x_i$;
- a sink arc $a \in A_-$ from a destination $(y_j, t_j^{drop})$ in $V_y$ to $(v_0, T)$ means that the VIPA closes its tour by returning to the depot via a move along a shortest path from $y_j$ to $v_0$.

Again, due to $f^*(a) \in \{0, 1\} \forall a \in A_T$, the composition of the tours is also particularly easy.
For each source arc $((v_0, 0), (x_j, t_j^{pick})) = a \in A_+$ with $f^*(a) = 1$, there is exactly one request arc $a' \in A_R$ which is the only outgoing arc from $(x_j, t_j^{pick})$ and has, due to flow conservation, also $f^*(a') = 1$. From each destination $(y_j, t_j^{drop})$ with an incoming request arc $a'$ with $f^*(a') = 1$, there is, due to flow conservation, exactly one outgoing link or sink arc $a$ with $f^*(a) = 1$.
Hence, the arcs $a \in A_T$ with $f^*(a) = 1$ exactly correspond to $k$ arc-disjoint directed paths from the source $(v_0, 0)$ to the sink $(v_0, T)$, and each of these paths equals one tour $\Gamma^i$ of one VIPA (composed by an alternating sequence of moves and actions).

Finally, provided that the profits are high enough, the chosen objective function (1a) guarantees that $\sigma_A$ is maximum and that the tours $\Gamma^1, \ldots, \Gamma^k$ have indeed minimum total tour length. □

**Example 2.** Consider an instance $(M, \sigma, p, 11, 2, 1)$ of the Offline TMP with

- the network $G$ with a depot $v_0$ with arcs having uniform distance, see Figure 1,

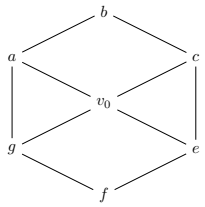


Figure 1: This figure illustrates the network $G$ of the instance $(M, \sigma, p, 11, 2, 1)$ of the Offline TMP from Example 2.

- two unit-speed servers (i.e. two VIPAs that travel 1 unit of length in 1 unit of time) with capacity Cap = 1 originally located at the depot $v_0$,

- the following sequence $\sigma$ of 6 requests:

$$r_1 = (0, a, c, 1, 4, 1) \quad r_3 = (1, e, f, 2, 4, 1) \quad r_5 = (5, b, c, 6, 8, 1)$$
$$r_2 = (1, c, f, 6, 9, 1) \quad r_4 = (3, b, a, 6, 9, 1) \quad r_6 = (5, c, e, 5, 8, 1)$$

- profits $p(r_j) = 4d(x_j, y_j)$ for accepted requests $r_j$.

An optimal solution $f^*$ in the resulting time-expanded request network $G_T$ is illustrated in Figure 2. We have $\sigma_A = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ and the following tours for the two VIPAs:

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 5) \xrightarrow{r_6} (e, 6) \to (c, 7) \xrightarrow{r_2} (f, 9) \to (v_0, T)$$
$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (b, 6) \xrightarrow{r_5} (c, 7) \to (b, 8) \xrightarrow{r_4} (a, 9) \to (v_0, T)$$

The total number of accepted requests is 6 with profit $4 \cdot 8$ served with a total tour length of 18, hence the value of the optimal offline solution is 14.

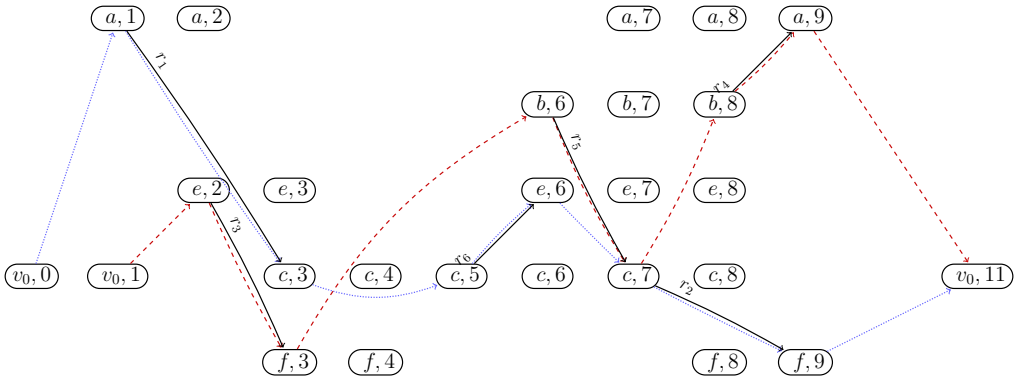


Figure 2: This figure shows the arcs with positive flow in the time-expanded request network $G_T$ for the instance $(M, \sigma, p, 11, 2, 1)$ of the Offline TMP from Example 2. The computed flow $f^*$ in the time-expanded request network $G_T$ can be interpreted as transportation schedule. The tour of the first VIPA is indicated by dashed arcs, and the tour of the second VIPA by dotted arcs. The total number of accepted requests is 6 served with a total tour length of 18.

### *3.2. Solving the Online Taxi Mode Problem*

To handle the online situation (where the requests in $\sigma$ are released over time during a time horizon $[0, T]$), we consider three approaches: besides a simple heuristic, we apply the two well-known meta-strategies Replan and Ignore that solve the online version of the PDP by solving a sequence of offline subproblems for certain time intervals $[t', T']$ within $[0, T]$ on accordingly modified request networks.

*Earliest Pickup Heuristic.* This simple heuristic 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, or rejecting the request if it is not reachable from all tours. Let $\Gamma^i$ be a tour and $(v_i, t_i)$ be its current end. A request $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ is *reachable* from $(v_i, t_i)$ if

$$t_i + d(v_i, x_j) \leq q_j - d(x_j, y_j)$$

is a possible pickup time of $r_j$.

**Algorithm 1 (Earliest Pickup Heuristic (EPH))**

**Input:** $(M, \sigma, p, T, k, \text{Cap})$

**Output:** $\sigma_A$ and tours $\Gamma^1, \ldots, \Gamma^k$

1: initialize $\sigma_A = \emptyset$, $\sigma(t') = \{r_j \in \sigma : t_j = 0\}$, and $\Gamma^i = (v_0, 0)$ for $1 \leq i \leq k$
2: WHILE $\sigma(t') \neq \emptyset$ DO:
  select $r_j \in \sigma(t')$ with $t_j$ minimal
  let $d = \infty$ and $\ell = 0$
  FOR $i = 1$ to $k$ DO:
    IF $r_j$ is *reachable* from current end $(v_i, t_i)$ of $\Gamma^i$ THEN
      let $d_i = d(v_i, x_j)$, IF $d_i < d$ THEN $d = d_i$, $\ell = i$
  IF $d = \infty$ THEN reject $r_j$ and remove it from $\sigma(t')$ (as $r_j$ is not reachable from any $\Gamma^i$)
  ELSE
    accept $r_j$ and move it from $\sigma(t')$ to $\sigma_A$
    update tour $\Gamma^\ell$ to $\Gamma^\ell = \Gamma^\ell \to (x_j, t_j + d) \to (y_j, t_j + d + d(x_j, y_j)$
  update $\sigma(t')$ by newly released requests
3: close all tours by returning to the depot
4: return $\sigma_A$ and $\Gamma^1, \ldots, \Gamma^k$

**Example 3.** Consider the instance $(M, \sigma, p, 11, 2, 1)$ of the TMP from Example 2. EPH proceeds with this request sequence $\sigma$ as follows. At the beginning, EPH initializes $\sigma_A = \emptyset$, the two tours $\Gamma^1 = \Gamma^2 = (v_0, 0)$ and, as $r_1 = (0, a, c, 1, 4, 1)$ is released at time $t' = 0$, $\sigma(t') = \{r_1\}$.

EPH takes $r_1$ and computes $d = 1$ and $l = 1$, updates $\Gamma^1$ to

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3)$$

and moves $r_1$ from $\sigma(t')$ to $\sigma_A$. At time $t' = 1$, $r_2 = (1, c, f, 6, 9, 1)$ and $r_3 = (1, e, f, 2, 4, 1)$ are released and enter $\sigma(t')$. By $p_2 = 6 > 2 = p_3$, EPH selects $r_3$ and computes $d = 1$ and $l = 2$, updates $\Gamma^2$ to

$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3)$$

and moves $r_3$ from $\sigma(t')$ to $\sigma_A$. Now, $\sigma(t') = \{r_2\}$ causes EPH to select $r_2$. EPH computes $d = 0$ and $l = 1$, updates $\Gamma^1$ to

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 6) \xrightarrow{r_2} (f, 8)$$

and moves $r_2$ from $\sigma(t')$ to $\sigma_A$. At time $t' = 3$, $r_4 = (3, b, a, 6, 9, 1)$ is released and enters $\sigma(t')$. EPH takes $r_4$ and computes $d = 3$ and $l = 2$, updates $\Gamma^2$ to

$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (b, 6) \xrightarrow{r_4} (a, 7)$$

and moves $r_4$ from $\sigma(t')$ to $\sigma_A$. At time $t' = 5$, $r_5 = (5, b, c, 6, 8, 1)$, $r_6 = (5, c, e, 5, 8, 1)$ are released and enter $\sigma(t')$. By $p_5 = 6 > 5 = p_6$, EPH selects $r_6$ and rejects it as it is not reachable from the ends of both tours. Then $r_5$ is left and also rejected as it is not reachable, too. Finally,

EPH closes both tours by returning to the depot and returns $\sigma_A = \{r_1, r_2, r_3, r_4\}$ and the following tours for the two VIPAs:

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 6) \xrightarrow{r_2} (f, 8) \to (v_0, 10)$$

$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (b, 6) \xrightarrow{r_4} (a, 7) \to (v_0, 8)$$

The total number of accepted requests is 4 with profit $4 \cdot 6$ served with a total tour length of 14, hence $\text{EPH}(\sigma) = 10$.

*Ignore.* Recall that the overall idea of an Ignore strategy is to construct, starting at time $t' = 0$, for the subsequence $\sigma(t')$ of currently waiting requests a (partial) optimal offline solution (which includes to determine which requests from $\sigma(t')$ can be accepted, and to compute optimal (partial) tours to serve them) and to completely perform these tours before it checks for newly released requests, updates $\sigma(t')$ and computes an optimal offline solution for the new subsequence $\sigma(t')$.

In our case with several servers, some partial tours for $\sigma(t')$ may be shorter than others such that waiting until all partial tours are completed may let some servers idle. Hence, we propose a variant of Ignore that updates $\sigma(t')$ whenever (at least) one server becomes idle (as it finished serving its tour) and plans new partial tours for $\sigma(t')$ with the $k'$ currently idle servers, i.e., on the subset $S(t', k') \subseteq \{\Gamma^1, \dots, \Gamma^k\}$ of finishes subtours. This is summarized in the algorithm IGNORE.

| **Algorithm 2 (IGNORE)** |
|---|
| **Input:** $(M, \sigma, p, T, k, \text{Cap})$ |
| **Output:** $\sigma_A$ and tours $\Gamma^1, \dots, \Gamma^k$ |
| 1: initialize $t' = 0$, $\sigma_A = \emptyset$, $\sigma(t') = \emptyset$, and $\Gamma^i = (v_0, 0)$ for $1 \le i \le k$ |
| 2: WHILE $t' < T$ DO:<br>let $k'$ be the number of currently idle servers<br>IF $k' > 0$ THEN:<br>update $t'$ and $\sigma(t') = \{r_j \in \sigma : t_j \le t'\}$<br>call I-OFFLINE with $\sigma_A$, $\sigma(t')$ and the tours $S(t', k')$ of the idle servers<br>completely perform the (modified) tours in $S(t', k')$ |
| 3: close all tours by returning to the depot |
| 4: return $\sigma_A$ and $\Gamma^1, \dots, \Gamma^k$ |

To compute the optimal solutions for the subsequences $\sigma(t')$ with the help of I-OFFLINE, we build a time-expanded request network $G_I(t')$ in a similar way as $G_T$ for the offline situation. The only difference is that we do not have a single source (as $(v_0, 0)$ in $G_T$), but that we need to use the current positions of the idle VIPAs as sources (i.e., the current ends of the tours in $S(t', k')$), collected in a position vector $P(t')$ with $t'$ as start time. Accordingly, we obtain

$$V_+ = \{(P_i(t'), t') : (P_i(t'), t') \text{ current end of tour } \Gamma^i \in S(t', k')\}$$

as set of source nodes and add the arcs from $(P_i(t'), t')$ to the sink $(v_0, T)$ for all such nodes. In $G_I(t')$, we solve the max profit flow problem (1) where (1b) is replaced by

$$\sum_{a \in \delta^+(v,t')} f(a) = k'(v) \quad \forall (v, t') \in V_+$$

and $k'(v)$ denotes the number of idle VIPAs situated in $v$ at time $t'$ (thus all $k'(v)$ sum up to $k'$).

From the flow computed in $G_I(t')$, it is straitforward to determine newly accepted requests (corresponding to request arcs $a \in A_R$ with $f(a) > 0$) and to construct (partial) tours $\Gamma^1, \ldots, \Gamma^k$ for the VIPAs in the same way as described for the offline situation (thereby ignoring sink arcs).

The whole process can be summarized in the algorithm I-OFFLINE.

| **Algorithm 3 (I-OFFLINE)** |
|---|
| **Input:** $\sigma_A$, $\sigma(t')$, $S(t', k') \subseteq \{\Gamma^1, \ldots, \Gamma^k\}$ |
| **Output:** modified $\sigma_A$ and modified tours in $S(t', k')$ |
| 1: determine VIPA start positions $P(t')$ as ends of tours in $S(t', k')$ |
| 2: create the request network $G_I(t')$ |
| 3: solve the modified max profit flow problem (1) on $G_I(t')$ |
| 4: update $\sigma_A$ and tours in $S(t', k')$ accordingly and return them |

**Example 4.** Consider the instance $(M, \sigma, p, 11, 2, 1)$ of the TMP from Example 2. IGNORE proceeds with this request sequence $\sigma$ as follows. At the beginning, IGNORE initializes $\sigma_A = \emptyset$, $\sigma(t') = \emptyset$, and the two tours $\Gamma^1 = \Gamma^2 = (v_0, 0)$. At time $t' = 0$, both VIPAs are idle s.t. $k' = 2 > 0$. Request $r_1 = (0, a, c, 1, 4, 1)$ is released and moves from $\sigma$ to $\sigma(t')$. IGNORE computes the partial offline solution for $\sigma(t') = \{r_1\}$ and $P(t') = (v_0, v_0)$ on the network $G_I(0)$, see Figure 3. IGNORE solves the max profit flow problem (1) on $G_I(0)$, obtains

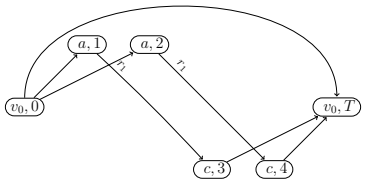


Figure 3: The request network $G_I(0)$ for Ignore.

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3)$$
$$\Gamma^2 = (v_0, 0)$$

moves $r_1$ from $\sigma(t')$ to $\sigma_A$ and starts VIPA 1 to serve $\Gamma^1$, whereas VIPA 2 stays idle in the depot.

At time $t' = 1$, $r_2 = (1, c, f, 6, 9, 1)$ and $r_3 = (1, e, f, 2, 4, 1)$ are released and move from $\sigma$ to $\sigma(t')$. As VIPA 2 is idle, IGNORE computes the partial offline solution for $\sigma(t') = \{r_2, r_3\}$ and $P(t') = (-, v_0)$ on the network $G_I(1)$, see Figure 4. IGNORE solves the modified max profit flow problem (1) on $G_I(1)$, obtains

$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (c, 6) \xrightarrow{r_2} (f, 8)$$

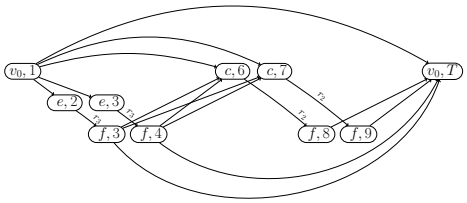


Figure 4: The request network $G_I(1)$ for Ignore.

moves $r_2, r_3$ from $\sigma(t')$ to $\sigma_A$ and starts VIPA 2 to serve $\Gamma^2$.

At time $t' = 3$, $\Gamma^1$ is served by VIPA 1, $r_4 = (3, b, g, 6, 9, 1)$ is released and moves from $\sigma$ to $\sigma(t')$. IGNORE computes the partial offline solution for $\sigma(t') = \{r_4\}$ and $P(t') = (c, -)$ on the network $G_I(3)$, see Figure 5. IGNORE solves the modified max profit flow problem (1) on $G_I(3)$,

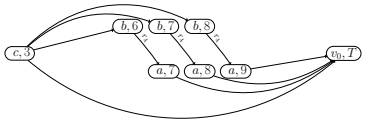


Figure 5: The request network $G_I(3)$ for Ignore.

obtains

$$\Gamma^1 = (c, 3) \to (b, 6) \xrightarrow{r_4} (a, 7)$$

moves $r_4$ from $\sigma(t')$ to $\sigma_A$ and starts VIPA 1 to serve $\Gamma^1$.

At time $t' = 7$, $\Gamma^1$ is served by VIPA 1, hence IGNORE checks for newly released requests and updates $\sigma(t') = \{r_5, r_6\}$. IGNORE computes the partial offline solution for $\sigma(t') = \{r_5, r_6\}$ and $P(t') = (a, -)$ on the network $G_I(7)$, see Figure 6. and obtains that none of $r_5, r_6$ is reachable

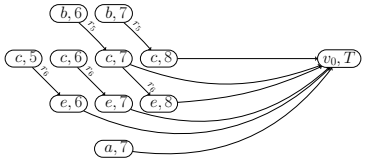


Figure 6: The request network $G_I(7)$ for Ignore.

from $(a, 7)$ such that both $r_5, r_6$ are rejected. Finally, IGNORE closes both tours by returning to the depot and returns $\sigma_A = \{r_1, r_2, r_3, r_4\}$ and the following tours for the two VIPAs:

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (b, 6) \xrightarrow{r_4} (a, 7) \to (v_0, 8)$$
$$\Gamma^2 = (v_0, 2) \to (e, 3) \xrightarrow{r_3} (f, 4) \to (c, 6) \xrightarrow{r_2} (f, 8) \to (v_0, 10)$$

The total number of accepted requests is 4 with profit $4 \cdot 6$ served with a total tour length of 14, hence IGNORE$(\sigma) = 10$.

*Replan.* Recall that the overall idea of a replan strategy is to consider at each moment in time $t' \in [0, T]$ the subsequence $\sigma(t')$ of currently waiting requests, to determine which requests from $\sigma(t')$ can be accepted, to compute optimal (partial) tours to serve them, and to perform these tours until new requests are released and to recompute $\sigma(t')$ and the tours (keeping already accepted requests).

Hereby, finding optimal (partial) tours corresponds to solve, in each replanning step, an optimal offline solution on the subsequence $\sigma(t')$. This is summarized in the algorithm REPLAN.

| **Algorithm 4 (REPLAN)** |
|---|
| **Input:** $(M, \sigma, p, T, k, \text{Cap})$ |
| **Output:** $\sigma_A$ and tours $\Gamma^1, \dots, \Gamma^k$ |
| 1: initialize $\sigma_A = \emptyset$, $\sigma(t') = \{r_j \in \sigma : t_j = 0\}$, and $\Gamma^i = (v_0, 0)$ for $1 \le i \le k$ |
| 2: WHILE $t' < T$ DO: call R-OFFLINE($\sigma_A$, $\sigma(t')$, $\Gamma^1, \dots, \Gamma^k$)<br>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 optimal solutions for the subsequences $\sigma(t')$, we build a time-expanded request network $G_R(t') = (V', A')$ based on $\sigma(t')$ and the original network $G$ and consider a flow in $G_R(t')$ that corresponds to the studied (partial) tours.

We construct $G_R(t') = (V', A')$ in a similar way as $G_T$ for the offline situation. The main difference is that we do not have a single source (as $(v_0, 0)$ in $G_T$), but that we need to use the possible start positions and possible start times of the VIPAs as sources.

For that, we extract the possible start positions $P(t')$ and start times $S(t')$ for the VIPAs from the current tours $\Gamma^1, \dots, \Gamma^k$. At the beginning, i.e. at time $t = 0$, we clearly have $P(t')_i = v_0$ and $S(t')_i = 0$. At any later time point $t'$, the start positions and start times are as follows: if VIPA $i$ is currently serving a request $r_j$, then we have $P(t')_i = y_j$ and $S(t')_i = t_j^{drop}$; otherwise, $P(t')_i$ is the current position $v$ of VIPA $i$ and $S(t')_i = t'$.

Accordingly, the node set $V' = V_+ \cup V_x \cup V_y \cup (v_0, T')$ is composed of

- the VIPAs start positions and start times $(P(t')_i, S(t')_i)$ for $1 \le i \le k$ as sources in $V_+$,
- all possible origins $(x_j, t_j^{pick})$ of all $r_j \in \sigma(t')$ and all $p_j \le t_j^{pick} \le q_j - d(x_j, y_j)$ in $V_x$,
- all possible destinations $(y_j, t_j^{drop})$ of all $r_j \in \sigma(t')$ and all $p_j + d(x_j, y_j) \le t_j^{drop} \le q_j$ in $V_y$,
- a sink node $(v_0, T')$ with $T' = \max\{t_j^{drop}, r_j \in \sigma(t')\}$.

The arc set $A' = A_+ \cup A_R \cup A_L \cup A_-$ is composed of

- source arcs from all nodes $(P(t')_i, S(t')_i) \in V_+$ to all reachable origins $(x_j, t_j^{pick}) \in V_x$ with $t' + d(v, x_j) \le t_j^{pick}$,
- request arcs from each $(x_j, t_j^{pick}) \in V_x$ to $(y_j, t_j^{pick} + d(x_j, y_j)) \in V_y$ in $A_R$,
- link arcs from all destinations $(y_j, t_j^{drop}) \in V_y$ to all reachable origins $(x_i, t_i^{pick}) \in V_x$ with $t_j^{drop} + d(y_j, x_i) \le t_i^{pick}$ in $A_L$,
- sink arcs from all destinations $(y_j, t_j^{drop}) \in V_y$ to $(v_0, T')$ in $A_-$.

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 $t'$,

and partition the request arcs accordingly in $A_{R_A}$ and $A_{R_N}$. Moreover, we distinguish the subsets $A^j_{R_A}$ and $A^j_{R_N}$ of request arcs of the corresponding previously accepted request $r_j \in \sigma_A(t')$ resp. newly released request $r_j \in \sigma_N(t')$.

In $G_R(t')$, we solve the following max profit flow problem

$$\max \sum_{a \in A_R} p(a)f(a) - \sum_{a \in A'} c(a)f(a) \tag{2a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v,t)} f(a) = k(v) \qquad \forall (v,t) \in V_+ \tag{2b}$$

$$\sum_{a \in \delta^-(v,t)} f(a) = \sum_{a \in \delta^+(v,t)} f(a) \qquad \forall (v,t) \in V_x \cup V_y \tag{2c}$$

$$\sum_{a \in A^j_{R_A}} f(a) = 1 \qquad \forall A^j_{R_A} \subseteq A_{R_A} \tag{2d}$$

$$\sum_{a \in A^j_{R_N}} f(a) \leq 1 \qquad \forall A^j_{R_N} \subseteq A_{R_N} \tag{2e}$$

$$f(a) \geq 0 \qquad \forall a \in A' \tag{2f}$$

$$f(a) \in \mathbf{Z} \qquad \forall a \in A' \tag{2g}$$

where again $\delta^-(v,t)$ denotes the set of outgoing arcs of $(v,t)$, $\delta^+(v,t)$ the set of incoming arcs of $(v,t)$ and $k(v)$ the number of VIPAs initially situated in $v$.

Constraints (2d) ensure that previously accepted requests are served whereas constraints (2e) allow to reject newly released requests.

Source, flow conservation and nonnegativity constraints (2b), (2c), (2f) together give again rise to a totally unimodular matrix, but due to (2d) and (2e) the entire constraint matrix is not totally unimodular s.t. integrality constraints (2g) are again required.

From the computed flow $f'$ in the request network $G_R(t')$, it is straitforward to determine newly accepted requests (corresponding to request arcs $a \in A_{R_N}$ with $f'(a) > 0$) and to construct (partial) tours $\Gamma^1, \ldots, \Gamma^k$ for the VIPAs in the same way as described for the offline situation.

The whole process can be summarized in the algorithm R-OFFLINE.

| **Algorithm 2 (R-OFFLINE)** |
|---|
| **Input:** $\sigma_A$, $\sigma(t')$, $\Gamma^1, \ldots, \Gamma^k$ |
| **Output:** modified $\sigma_A$ and modified tours $\Gamma^1, \ldots, \Gamma^k$ |
| 1: determine VIPA start positions $P(t')$ and start times $S(t')$ from $\Gamma^1, \ldots, \Gamma^k$ |
| 2: create the request network $G_R(t')$ |
| 3: solve the max profit flow problem (2) on $G_R(t')$ |
| 4: update $\sigma_A$ and $\Gamma^1, \ldots, \Gamma^k$ accordingly and return them |

**Example 5.** Consider the instance $(M, \sigma, p, 11, 2, 1)$ of the TMP from Example 2. REPLAN proceeds with this request sequence $\sigma$ as follows. At the beginning, REPLAN initializes $\sigma_A = \emptyset$, and the two tours $\Gamma^1 = \Gamma^2 = (v_0, 0)$. At time $t' = 0$, $r_1 = (0, a, c, 1, 4, 1)$ is released. REPLAN computes the partial offline solution for $\sigma_A(0) = \emptyset$, $\sigma_N(0) = \{r_1\}$, $S(0) = (0, 0)$ and $P(0) = (v_0, v_0)$ on the network $G_R(0)$, see Figure 7. REPLAN solves the max profit flow problem (2) on $G_R(0)$,

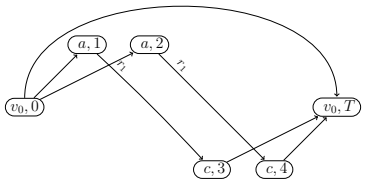


Figure 7: The request network $G_R(0)$ from Example 5.

obtains

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (v_0, T)$$
$$\Gamma^2 = (v_0, 0) \to (v_0, T)$$

accepts $r_1$ and moves VIPA 1 towards $a$.
At time $t' = 1$, $r_2 = (1, c, f, 6, 9, 1)$ and $r_3 = (1, e, f, 2, 4, 1)$ are released. REPLAN computes the partial optimal offline solution for $\sigma_A(1) = \{r_1\}$, $\sigma_N(1) = \{r_2, r_3\}$, $S(1) = (1, 1)$ and $P(1) = (a, v_0)$ on the network $G_R(1)$, see Figure 8.

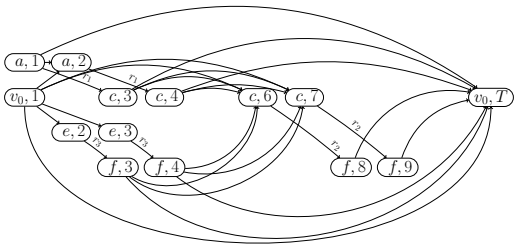


Figure 8: The request network $G_R(1)$ from Example 5.

REPLAN solves the max profit flow problem (2) on $G_R(1)$, obtains

$$\Gamma^1 = (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 6) \xrightarrow{r_2} (f, 8) \to (v_0, T)$$
$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (v_0, T)$$

accepts $r_2$ and $r_3$, moves VIPA 1 towards $c$ (serving $r_1$) and VIPA 2 towards $e$.
At time $t' = 3$, $r_1$ and $r_3$ are served and $r_4 = (3, b, a, 6, 9, 1)$ is released. REPLAN computes the partial optimal offline solution for $\sigma_A(3) = \{r_2\}$, $\sigma_N(3) = \{r_4\}$, $S(3) = (3, 3)$ and $P(3) = (c, f)$ on the network $G_R(3)$, see Figure 9.
REPLAN solves the max profit flow problem (2) on $G_R(3)$, obtains

$$\Gamma^1 = (c, 3) \to (c, 6) \xrightarrow{r_2} (f, 8) \to (v_0, T)$$
$$\Gamma^2 = (f, 3) \to (b, 6) \xrightarrow{r_4} (a, 7) \to (v_0, T)$$

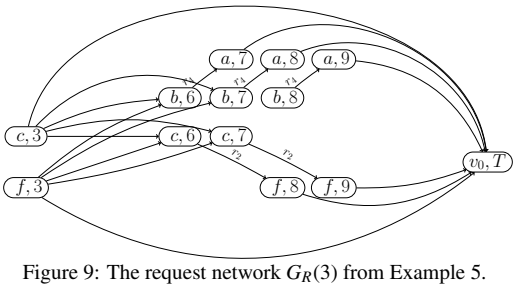


Figure 9: The request network $G_R(3)$ from Example 5.

and accepts $r_4$. REPLAN moves VIPA 2 towards $(b, 6)$ on one of the two shortest paths s.t. VIPA 2 moves either towards $e$ and then towards $c$ or towards $g$ and then towards $a$. In both cases VIPA 2 can reach $(b, 6)$.
At time $t' = 5$, $r_5 = (5, b, c, 6, 8, 1)$, $r_6 = (5, c, e, 5, 8, 1)$ are released. REPLAN computes the partial optimal offline solution for $\sigma_A(5) = \{r_2, r_4\}$, $\sigma_N(5) = \{r_5, r_6\}$, $S(5) = (5, 5)$ and $P(5) = (c, c)$ on the network $G_R(5)$, see Figure 10. Note that the tours obtained do not change whether $P(5) = (c, a)$ or $P(5) = (c, c)$. In Figure 10, $P(5) = (c, c)$.

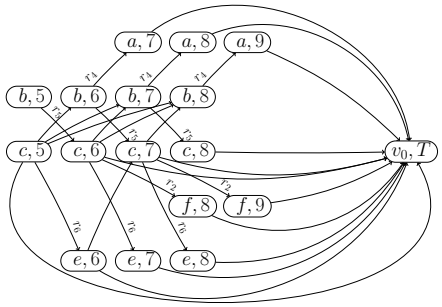


Figure 10: The request network $G_R(5)$ from Example 5.

REPLAN solves the max profit flow problem (2) on $G_R(5)$, obtains

$$\Gamma^1 = (c, 5) \xrightarrow{r_6} (e, 6) \to (c, 7) \xrightarrow{r_2} (f, 9) \to (v_0, T)$$
$$\Gamma^2 = (c, 5) \to (b, 6) \xrightarrow{r_5} (c, 7) \to (b, 8) \xrightarrow{r_4} (a, 9) \to (v_0, T)$$

and accepts $r_5$ and $r_6$.
In total, REPLAN accepts all 6 requests with $\sigma_A = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ and serves them by the tours

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 5) \xrightarrow{r_6} (e, 6) \to (c, 7) \xrightarrow{r_2} (f, 9) \to (v_0, T)$$
$$\Gamma^2 = (v_0, 0) \to (v_0, 1) \to (e, 2) \xrightarrow{r_3} (f, 3) \to (b, 6) \xrightarrow{r_5} (c, 7) \to (b, 8) \xrightarrow{r_4} (a, 9) \to (v_0, T)$$

with a total tour length of 18, and a profit $4 \cdot 8$ hence REPLAN$(\sigma) = 14$.

*Discussion of the approaches.* In view of the behavior of the three proposed algorithms observed on the instance from Example 2, we note that requests that are released long time before the time window to serve them cause difficulties for all three algorithms. Such a request $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ (like $r_2$ in the example) causes

- EPH and IGNORE to construct a tour $\Gamma^i$ serving $r_j$ at its end where VIPA $i$ stays idle within $\Gamma^i$ until $p_j$ (see $\Gamma^1$ of EPH between $(c, 3)$ and $(c, 6) = (x_2, p_2)$ when VIPA 1 starts to serve

$r_2$) whereas other requests released meanwhile are not reachable from the end of $\Gamma^i$ (like $r_5, r_6$ in the example).

- REPLAN to accept $r_j$ which may cause later to reject other requests released later on if they cannot be integrated into a tour serving $r_j$.

Moreover, while EPH and REPLAN check newly released requests immediately and also decide immediately about their acceptance / rejection, IGNORE checks for newly released requests only when a VIPA becomes idle. This may result in late decisions about the acceptance / rejection of requests (like for $r_5, r_6$ in the example which are rejected shortly before the end of their time window, at the latest possible pickup time), and may even result in a rejection after the time window of the request.

Hence, we conclude that (even the here considered variant of) IGNORE is not suitable for the Online TMP since the way how to construct tours may result in many rejected requests and the decision to accept/reject a request may be taken late, which does not comply to the quality-of-service aspect of the fleet management. Therefore, we focus on the other two approaches and perform computational results only for EPH and REPLAN, with the expectation that EPH is faster, but REPLAN achieves a higher acceptance rate.

## 4. Evaluating the performance of the online strategies

We shall evaluate the online algorithms EPH and REPLAN in a two-fold manner:

- in theory with the help of competitive analysis (Section 4.1),
- in practice with the help of some computational results (Section 4.2).

### *4.1. Competitive analysis*

It is standard to evaluate the quality of online algorithms with the help of competitive analysis. A detailed introduction to online optimization and competitive analysis can be found e.g. in the book by Borodin and El-Yaniv [8].

Competitive analysis can be viewed as a game between an online algorithm ALG and a malicious adversary who tries to generate a worst-case request sequence $\sigma$ which maximizes the ratio between the online cost $\mathrm{ALG}(\sigma)$ and the optimal cost $\mathrm{OPT}(\sigma)$ where the adversary knows the entire request sequence $\sigma$ in advance.

An online algorithm ALG for an online maximization problem is called *c-competitive* if ALG produces for any request sequence $\sigma$ of the studied problem a feasible solution with costs $\mathrm{ALG}(\sigma)$ such that

$$\mathrm{OPT}(\sigma) \leq c \cdot \mathrm{ALG}(\sigma)$$

for some given $c \geq 1$. The *competitive ratio* of ALG is the infimum over all $c$ such that ALG is $c$-competitive.

We are interested in analyzing the online algorithms EPH and REPLAN for the Online TMP.

In fact, we obtain the more general result that no (deterministic) online algorithm ALG for the Online TMP is competitive against a common type of adversary.

An *oblivious adversary* 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 $r_j$).

We show 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.

**Theorem 6.** *There is no competitive (deterministic) online algorithm for the Online TMP against an oblivious adversary.*

*Proof.* The idea for a worst case sequence for an online algorithm ALG is as follows. The adversary releases the requests $r_j \in \sigma$ in such a way that the delay between the release date $t_j$ and the latest possible pickup time $q_j - d(x_j, y_j)$ is smaller than the distance $d(v_0, x_j)$. That way, ALG has to reject all requests (and its VIPA stays in the depot $v_0$), whereas the adversary moves its VIPA already towards the origin $x_0$ of the first request $r_0$ before $r_0$ has been released and is able to arrive at $x_0$ at time $q_0 - d(x_0, y_0)$ and can accept and serve $r_0$ and all following requests in the sequence $\sigma$. For that, we consider an instance $(M, \sigma, p, T, 1, 1)$ of the Online TMP with

- the network $G$ with depot $v_0$ from Figure 11

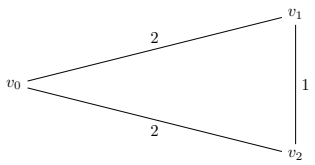


Figure 11: This figure illustrates the network $G$ of the instance $(M, \sigma, p, T, 1, 1)$ of the Online TMP with an oblivious adversary.

- the following sequence $\sigma = \{r_0, r_1, r_2, \ldots, r_\ell\}$ of requests with

$$r_j = (j+1, v_1, v_2, j+2, j+3, 1) \text{ for all even } j \text{ with } 0 \leq j \leq \ell,$$
$$r_j = (j+1, v_2, v_1, j+2, j+3, 1) \text{ for all odd } j \text{ with} 1 \leq j \leq \ell,$$

- profits $p(r_j) = 2d(x_j, y_j)$ for accepted requests $r_j$.

The online algorithm ALG treats the sequence $\sigma$ as follows. At time $t = 1$, the first request $r_0 = (1, v_1, v_2, 2, 3, 1)$ is released. As the origin $x_0 = v_1$ of $r_0$ is not reachable from the depot before or at the latest possible pickup time $q_0 - d(v_1, v_2) = 2$ due to $d(v_0, v_1) = 2$, ALG rejects $r_0$ and the VIPA operated by ALG stays in the depot $v_0$. At time $t = 2$, request $r_1 = (2, v_2, v_1, 3, 4, 1)$ is released. Again, the origin $x_1 = v_2$ of $r_1$ is not reachable from the depot before or at the latest possible pickup time $q_1 - d(v_1, v_2) = 3$ due to $d(v_0, v_2) = 2$. Hence, ALG also rejects $r_1$ and the VIPA operated by ALG stays in $v_0$. This is repeated for any further request $r_l \in \sigma$ so that all $r_j \in \sigma$ are rejected by ALG and we clearly have

$$\text{ALG}(\sigma) = 0.$$

In contrary, the adversary moves its VIPA at time $t = 0$ from the depot $v_0$ towards $x_0 = v_1$, arrives at $p_0 = 2$ in $v_1$ and accepts and serves $r_0$ by moving to $y_0 = v_2$, arriving there at time $3 = p_1$, Thus, the adversary can accept and serve $r_1$ by moving to $v_1 = y_1$, arriving there at time $4 = p_2$. This is repeated for any further request $r_j$ in $\sigma$ (that the VIPA operated by the adversary always

arrives at $x_j$ at time $p_j$) so that the adversary can accept and serve all requests $r_j$ in $\sigma$. At the end of the sequence the adversary returns its VIPA to the depot to close its tour. Thus we obtain

$$\mathrm{OPT}(\sigma) = (\ell + 1) \cdot 2d(v_1, v_2) - ((\ell + 1) \cdot d(v_1, v_2) + 2 + 2) = (\ell + 1) \cdot d(v_1, v_2) - 4 = \ell - 3.$$

This shows

$$\frac{\mathrm{OPT}(\sigma)}{\mathrm{ALG}(\sigma)} = \infty$$

so that there is no finite number $c$ bounding the ratio between $\mathrm{OPT}(\sigma)$ and $\mathrm{ALG}(\sigma)$ for all possible request sequences $\sigma$ of the Online TMP. □

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 7.** *Neither EPH nor REPLAN is competitive for the Online TMP against an oblivious adversary.*

### *4.2. Computational results*

This section deals with computational experiments for the optimal offline solution of the TMP, the heuristic EPH and the replan strategy for the Online TMP. In fact, due to the very special request structures of the previously presented worst-case instances to prove the non-competitiveness of any online algorithm for the Online TMP, we expect a better behavior of the proposed strategies for the Online TMP in average.

The computational results presented in Table 1 support this expectation. They compare the total number of accepted (and thus served) requests by EPH and REPLAN with the optimal offline solution OPT. 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 [24].

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. The results are summarized in Table 1.

EPH computes in very short times solutions (always less than 1 second), but can only reach in average an acceptance rate of about 32% compared to the optimal offline solution OPT.

Also REPLAN computes solutions for each replanning step within a short time, and can achieve a reasonable ratio w.r.t. the total number of accepted requests between the optimal offline solution OPT and REPLAN (in average around 65%).

## 5. Concluding remarks

We note for the Offline TMP, that in the special case of tight time windows satisfying $p_j + d(x_j, y_j) = q_j$ (which clearly results in $p_j = t_j^{pick}$ and $q_j = t_j^{drop}$) for all $r_j \in \sigma$, there is exactly one request arc per request s.t. the constraints (1d) reduce to

$$f(a) \leq 1 \text{ for all } a \in A_R. \tag{3}$$

Table 1: This table shows the computational results for 600 test instances of EPH and REPLAN in comparison to OFFLINE for the TMP. The instances are grouped by the number of requests (1st column), the time horizon (2nd column) and the number of VIPAs (3rd column) with 100 instances per parameter set. Average values are shown for the total number $|\sigma_A|$ of accepted requests of OFFLINE, REPLAN and EPH and the ratio between them ($\frac{REPLAN}{OPT}$ and $\frac{EPH}{OPT}$) and for the total tour length $TTL$ needed to serve the accepted requests. Finally we provide the time needed to compute the optimal offline solution, the average runtime of REPLAN per recomputation step and the maximum runtime $max_R$ of the recomputation steps of REPLAN. The average runtime of EPH is not shown as it never exceeds one second.

| | | | $\lvert\sigma_A\rvert$ | | | | | TTL | | | runtime (s) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| req | $T$ | $k$ | $\lvert\sigma_A\rvert_{OPT}$ | $\lvert\sigma_A\rvert_R$ | $ratio_R$ % | $\lvert\sigma_A\rvert_{EPH}$ | $ratio_{EPH}$% | $TTL_{OPT}$ | $TTL_R$ | $TTL_{EPH}$ | OPT | REPLAN | $max_R$ |
| 94 | 180 | 10 | 77 | 52,13 | 67,7 | 25,15 | 32,66 | 667,5 | 424 | 267,05 | 11,9 | 0,49 | 1,6 |
| 188 | 180 | 10 | 112 | 70,45 | 62,9 | 47,26 | 42,19 | 831 | 580 | 430,64 | 151 | 3 | 10,83 |
| 295 | 240 | 10 | 146,86 | 97,2 | 66,19 | 43,63 | 29,7 | 1005 | 750,57 | 458,32 | 76867,86 | 13,56 | 45,54 |
| 97 | 240 | 5 | 62,04 | 39,19 | 63,17 | 18,78 | 30,27 | 527,16 | 298,82 | 187,68 | 1650,94 | 0,29 | 1,23 |
| 194 | 240 | 5 | 93,76 | 55,84 | 59,56 | 27,6 | 29,43 | 680,44 | 490 | 286,55 | 229,76 | 1,8 | 7,85 |
| 290 | 240 | 5 | 115,94 | 80,64 | 69,55 | 31,93 | 27,54 | 759,94 | 500,6 | 291,3 | 121985,32 | 7,18 | 29,8 |

Therefore, in the case with tight time windows, the totally unimodular matrix implied by the source and flow conservation constraints (i.e., the node-arc incidence matrix of the digraph underlying $G_T$) is only composed with identity matrices (for the nonnegativity and the capacity constraints (3)) such that the entire constraint matrix becomes totally unimodular. This implies:

**Corollary 8.** *The Offline Taxi Mode Problem with tight time windows can be solved in polynomial time.*

In the general case, this is not true, but our experiments showed that the running times to solve the Offline Taxi Mode Problem are still reasonable, see Table 1.

Regarding the quality of the solutions obtained by EPH and REPLAN, we summarize from the previous section that

- in theory, neither EPH nor REPLAN is competitive against an oblivious adversary since for all (deterministic) online algorithms ALG for the Online TMP, there is no finite $c$ s.t. for all instances $\sigma$ we have that $\text{ALG}(\sigma) \geq c\ \text{OPT}(\sigma)$;
- in practice, EPH is faster, REPLAN provides solutions of reasonably quality within short time for each recomputation step and achieves a better acceptance rate, see again Table 1.

We can conclude that the here proposed REPLAN strategy is already a promising algorithm to handle the Online TMP for the taxi mode in the studied VIPAFLEET management system.

As future work, we plan to improve the runtime of REPLAN by reducing the time-expanded network built in each replanning step without loss of optimality. Such an approach has been applied by [7] to a service network design problem and by [21] to a multi-depot multi-vehicle bus scheduling problem for timetabled trips (by using time-space-based instead of connection-based networks which leads to a crucial reduction of the size of the mathematical models).

We also plan to study another variant of the TMP without the here required condition that, at each time, at most one request $r_j$ can be served by a VIPA and that this has to be done in a direct way along a shortest path from $x_j$ to $y_j$. Dropping this condition would open the possibility to serve several requests simultaneously by a same VIPA (as long as the capacity Cap is respected), but that while serving request $r_j$, sometimes detours are necessary to stations not lying on a shortest path from $x_j$ to $y_j$ in order to pickup or drop passengers from other requests. This problem variant will lead to a more complex model and also computing solutions is more involved, but may lead to a higher rate of accepted requests and, therefore, to a higher quality-of-service level for the fleet management.

## References


[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 2017.

[8] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[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. *arXiv preprint arXiv:1703.10565*, 2017.

[11] Sahar Bsaybes, Alain Quilliot, and Annegret K Wagler. Fleet management for autonomous vehicles using flows in time-expanded networks. *Electronic Notes in Discrete Mathematics (special issue of LAGOS 2017)*, 62:255–260, 2017.

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

[13] 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)*, number 2013, pages 185–205, 2013.

[14] 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.

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

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

[17] 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.

[18] http://www.easymile.com. Easymile, 2015.

[19] http://www.ligier.fr. Ligier group, 2015.

[20] http://www.viameca.fr. Viaméca, 2015.

[21] Natalia Kliewer, Taieb Mellouli, and Leena Suhl. A time–space network based exact optimization model for multi-depot bus scheduling. *European journal of operational research*, 175(3):1616–1627, 2006.

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

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

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

[25] 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.

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