RAIRO-Oper. Res. 53 (2019) 1007–1031
https://doi.org/10.1051/ro/2018042

RAIRO Operations Research
www.rairo-ro.org

FLEET MANAGEMENT FOR AUTONOMOUS VEHICLES: ONLINE PDP UNDER SPECIAL CONSTRAINTS*

Sahar Bsaybes[1,*], Alain Quilliot[2] and Annegret K. Wagler[2]

Abstract. The VIPAFLEET project consists in developing models and algorithms for managing a fleet of Individual Public Autonomous Vehicles (VIPA). Hereby, 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). One goal is to develop and implement suitable algorithms for each mode in order to satisfy all the requests under an economic point of view by minimizing the total tour length. The innovative idea and challenge of the project is to develop and install a dynamic fleet management system that allows the operator to switch between the different modes within the different periods of the day according to the dynamic transportation demands of the users. We model the underlying online transportation system and propose a corresponding fleet management framework, to handle modes, demands and commands. We consider two modes of circulation, tram and elevator mode, propose for each mode appropriate online algorithms and evaluate their performance, both in terms of competitive analysis and practical behavior.

Mathematics Subject Classification. 90C27, 05C21.

Received April 30, 2017. Accepted May 30, 2018.

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, business centers, big parkings, airports and train stations. A VIPA is an "autonomous vehicle" that does not require a driver nor an infrastructure to operate, it is developed by Easymile and Ligier [8,11] thanks to innovative computer vision guidance technologies developed by researchers at Institut Pascal [13,14]. This innovative project involves different partners in order to ensure the reliability of the transportation system [17]. A long-term experimentation [14] has been performed on the industrial site "Ladoux" of Michelin at Clermont-Ferrand as part of the VIPAFLEET project. A fleet of VIPAs shall be used in an industrial site to transport employees and visitors *e.g.* between

Keywords. Fleet management, Online transportation problem, autonomous vehicles, competitive analysis.

*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.

[1] Université Grenoble Alpes, CNRS, Grenoble INP (Institute of Engineering Univ. Grenoble Alpes), G-SCOP, F-38000 Grenoble, France.

[2] Université Clermont Auvergne (LIMOS UMR CNRS 6158), Clermont-Ferrand, France.

* Corresponding author: bsaybes@isima.fr

parkings, buildings and from or to the restaurant at lunch breaks. The fleet is distributed at specified stations to supply internal transportation, and a VIPA can operate in three different transportation 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 customer requests by moving to a station to let users enter or leave, thereby changing their driving direction if needed.
- *Taxi mode:* users book their transport requests (from any start to any destination station within the network with a start and an arrival time) in real time.

This leads to a Pickup-and-Delivery Problem (PDP) 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 (DARP). Many variants are studied including the DARP with time windows [7, 9] and Vehicle Routing Problems with time windows [10, 16]. In our case, we are confronted with an online situation, where transportation requests are released over time [2, 3, 6, 15].

In a practical context, different objective functions can be considered. We focus on the economic aspect where the objective is to minimize costs by minimizing the total tour length.

In a VIPAFLEET system, users can either call a VIPA directly from a station with the help of a call-box or can book their request in real time by mobile or web applications. Since the customer requests are released over time, the standard static PDP does not reflect the real situation of this project. Therefore we are interested in its online version. Our aim is to develop and install a Dynamic Fleet Management System that allows the operator to switch between different network designs, transportation modes and appropriate online algorithms within the different periods of the day in order to react to changing demands evolving during the day, with the objective to satisfy all demands in a best possible way. For that we model the underlying online transportation system and propose an according fleet management framework, to handle modes, demands and commands (see Sect. 2). In Section 3 we develop suitable online algorithms for tram and elevator mode and analyse them in terms of competitive analysis w.r.t. minimizing the total tour length. In Section 4, we provide computational results showing that the practical behavior of the algorithms is much better than their worst case behavior captured by their competitive factors. This enables us to cluster the demands into subproblems in such a way that, for each subproblem, a suitable subnetwork and a suitable algorithm can be proposed leading to a globally good solution (transportation schedule).

2. Problem description and model

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, arcs to their physical links in the closed site, and the distance d between two nodes $v_i, v_j \in V$ is the length of a shortest path from v_i to v_j. In V, we have a distinguished origin $v_o \in V$, the depot of the system where all k VIPAs are parked when the system is not running, *i.e.*, outside a certain time horizon $[0, T]$.

A Dynamic Fleet Management System shall allow the operator to switch between different transportation modes within the different periods of the day in order to react to changing customer demands evolving during the day. For that, for a certain period $[t, t'] \subseteq [0, T]$, we define a metric subspace $M' = (V', d')$ induced by a subnetwork $G' = (V', E')$ of G, where a subset of nodes and arcs of the network is active (*i.e.* where the VIPAs have the right to perform a move on this arc or pass by this node during $[t, t']$). An operator has to decide when and how to move the VIPAs in the subnetworks, and to assign customer requests to VIPAs.

Any customer request r_j is defined as a 4-tuple $r_j = (t_j, x_j, y_j, z_j)$ where

- $t_j \in [0, T]$ is the release date,
- $x_j \in V$ is the origin node,
- $y_j \in V$ is the destination node,

– z_j specifies the number of passengers.

The operator monitors the evolution of the requests and the movement of VIPAs over time and creates tasks to move the VIPAs to go to some station and pick up, transport and deliver users. More precisely, a *task* can be defined as $\tau_j = (t_j, x_j, y_j, z_j, G')$. This task is sent at time t_j to a VIPA operating in the subnetwork G' containing stations x_j and y_j, indicating that z_j passengers have to be picked up at x_j and delivered at y_j. In order to fulfill the tasks, we let a fleet of VIPAs (one or many, each having a capacity for Cap passengers) circulate in the network inducing the metric space.

More precisely we determine a feasible transportation schedule S for $(M, \mathcal{T})$ consisting of a collection of tours $\{\Gamma^1, \ldots, \Gamma^k\}$ where:

– each of the k VIPAs has exactly one tour,
– each request r_j is served not earlier than the time t_j it is released,
– each tour starts and ends in the depot.

If each user is transported from its start station to its final destination by only one VIPA, then S is called non-preemptive, otherwise preemptive. In particular, if start and destination station of a user do not lie on the same subnetwork G', the user has to change VIPAs on one or more intermediate stations. As this typically leads to inconveniences, the design of subnetworks has to be done in such a manner that preemption can be mostly avoided.

In addition, depending on the policy of the operator of such a system, different technical side constraints have to be obeyed. If two or many VIPAs circulate on the same (sub)network, the fleet management has to handle *e.g.* the

– meeting of two vehicles on a station or an arc,
– blocking the route of a VIPA by another one waiting at a station (if two VIPAs are not allowed to enter the same node or arc at the same time),

and has to take into account

– the events of breakdown or discharge of a vehicle,
– technical problems with the server, the data base or the communication network between the stations, VIPAs and the central server.

Our goal is to construct transportation schedules S for the VIPAs respecting all the above constraints w.r.t. minimizing the total tour length.

This will be addressed by dividing the time horizon $[0, T]$ in different periods according to the volume and kind of the requests, and by providing specific solutions within each period. The global goal is to provide a feasible transportation schedule over the whole time horizon that satisfies all requests and minimizes the total tour length (Dynamic Fleet Management Problem). For each period, we partition the network $G = (V, E)$ into a set of subnetworks $G' = (V', E')$. The aim of this partition is to solve some technical side constraints for autonomous vehicles, and to solve the online PDP using different algorithms at the same time on different subnetworks. To gain precision in solutions by applying a suitable algorithm to a certain subnetwork, the choice of the design of the network in the industrial site where the VIPAs will operate is dynamic and will change over time according to the technical features and properties.

In a metric space we partition the network into subnetworks G' that are either unidirected cycles, called *circuits* G'_c, or bidirected paths, called *lines* G'_ℓ. This partition is motivated by two of the operation modes for VIPAs, tram mode and elevator mode, that we will consider in the sequel of the paper.

3. Scenarios: combinations of modes and subnetworks

Based on all the above technical features and properties that have an impact on the feasibility of the transportation schedule, we can cluster the requests into subproblems, apply to each subproblem a certain algorithm, and check the results in terms of feasibility and performance.

The choice of the design of the network in the industrial site where the VIPAs will operate is dynamic and will change over time according to the technical features and properties. We consider four typical scenarios that occurred while operating a fleet in an industrial site[3] based on some preliminary studies of the transport requests within the site.

Morning/evening: The transport requests are between parkings and buildings. For this time period, we propose the following.

- Design a collection of subnetworks (lines and circuits) as shown in Figure 1 s.t.
  - all buildings and parkings are covered,
  - each subnetwork contains one parking p and all the buildings where p is the nearest parking (to ensure that for each request, during the morning origin (the parking) and destination (a building) lie in the same subnetwork, and during the evening destination (the parking) and origin (a building) lie in the same subnetwork).
- Depending on the number of employees in the served buildings, assign one VIPA (in elevator mode) to every line and one or several VIPAs (in tram mode) to each circuit.

Lunch time: The transport requests are between buildings and the restaurant of the industrial complex. For this time period, we propose the following.

- Design a collection of lines and circuits as shown in Figure 2 s.t.
  - all buildings are covered,
  - each subnetwork contains the station of the restaurant (to ensure that for each request, to or from the restaurant, origin and destination lie in the same subnetwork).
- Depending on the number of employees in the served buildings, assign one VIPA (in elevator mode) or one or several VIPAs (in tram mode) to the subnetworks.

Emergency case: In the case of a breakdown of the central servers, the database or the communication system, transports between all possible origin/destination pairs have to be ensured without any decision by the operator. For that we propose

- to use one Hamilton cycle through all the stations as subnetwork and
- to let half of the fleet of VIPAs operate in each direction on the cycle (all in tram mode).

Other periods: There are mainly unspecified requests without common origins or common destinations. The operator can use all VIPAs in his fleet in taxi mode on the complete network or design lines and circuits s.t. all stations are covered and the chosen subnetworks intersect (to ensure transports between all possible origin/destination pairs). *E.g.*, this can be done by

- using one Hamilton cycle through all stations where half of the fleet operates (in tram mode) in each direction,
- a spanning collection of lines and circuits meeting in a central station where one VIPA (in elevator mode) operates on each line, one or several VIPAs (in tram mode) on each circuit.

Remark 3.1. One of these combinations of subnetworks has been used in a long duration experimentation [14] on the industrial site "Ladoux" of Michelin at Clermont-Ferrand. VIPAs were operating using the tram mode on a unidirected circuit where some of the buildings and parkings of the industrial site were covered.

[3]A long-term experimentation [14] has been performed in the industrial site "Ladoux" of Michelin at Clermont-Ferrand where a fleet of VIPAs was operating for several months (October 2014–February 2015).

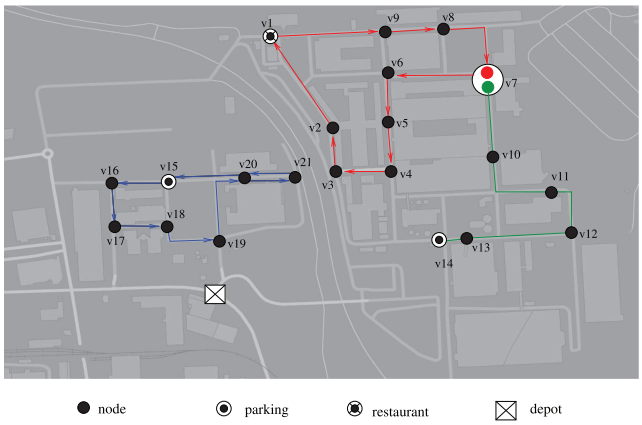


FIGURE 1. This figure illustrates an example of a collection of subnetworks (one line and two circuits) for the morning/evening scenario. All buildings and parkings are covered in this partition. Each subnetwork contains one parking (black dot within a white dot). The station v_7 is a common station for two subnetworks.

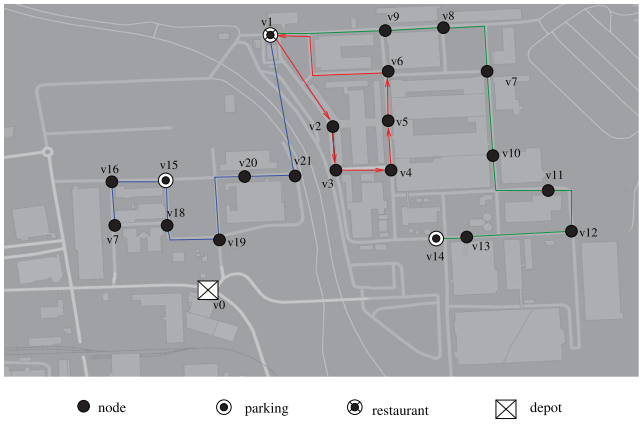


FIGURE 2. This figure illustrates an example of a collection of subnetworks (two lines and one circuit) for the lunch scenario. All stations (buildings and parkings) are covered in this partition, and each subnetwork contains the station of the restaurant (a cross within a circle).

4. Online algorithms and competitive analysis

Recall that the customer requests are released over time s.t. the studied PDP has to be considered in its online version. A detailed introduction to online optimization can be found *e.g.* in the book by Borodin and El-Yaniv [5]. 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 $\mathrm{ALG}(\sigma)$ and the optimal offline cost $\mathrm{OPT}(\sigma)$ knowing the entire request sequence σ in advance. For an online minimization problem, ALG is called *c-competitive* if ALG produces for any request sequence σ a feasible solution with

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

for some given $c \geq 1$. The competitive ratio of ALG is the infimum over all c such that ALG is c-competitive. Here, we are interested in designing and analyzing online algorithms for two possible operating modes of a VIPA, tram mode and elevator mode.

4.1. Tram mode

The tram mode is the most restricted operation mode where VIPAs run on predefined circuits in a predefined direction and stop at stations to let users enter or leave. We consider circuits C with one distinguished node, the origin of the circuit[4].

Optimal offline solution via colorings of interval graphs: An interval graph $\mathcal{G}(\mathcal{I})$ is obtained as intersection of a set $\mathcal{I}$ of intervals within a line segment, where

- the nodes of $\mathcal{G}(\mathcal{I})$ represent the intervals in $\mathcal{I}$,
- the edges of $\mathcal{G}(\mathcal{I})$ represent their conflicts in terms of overlaps (*i.e.* two nodes are adjacent if the corresponding intervals have a non-empty intersection).

The clique number $\omega(\mathcal{G}(\mathcal{I}))$ corresponds to the largest number of pairwise intersecting intervals in $\mathcal{I}$, a coloring corresponds to an assignment of colors to intervals such that no two intersecting intervals receive the same color. In all graphs, the clique number is a lower bound on the minimum number $\chi(\mathcal{G}(\mathcal{I}))$ of required colors. For interval graphs it was shown in [12] that the following Greedy coloring algorithm always produces an $\omega(\mathcal{G}(\mathcal{I}))$-coloring of $\mathcal{G}(\mathcal{I})$:

- sort all intervals in $\mathcal{I}$ according to their left end points.
- color the nodes of $\mathcal{G}(\mathcal{I})$ in this order: starting with the first node, assign to each node the smallest color that none of its already colored neighbors has.

We next interpret the offline solution for VIPAs operating in tram mode on a circuit in this context. We have given a circuit $C = \{v_o, v_1, \ldots, v_n\}$ and a sequence σ of m requests $r_j = (t_j, x_j, y_j, z_j)$ with origin/destination pairs $(x_j, y_j) \in C \times C$. W.l.o.g. we may assume that the origin v_0 of the circuit does not lie in the interior of (x_j, y_j) for any $r_j \in \sigma$. We transform C into a path $P = \{v_0, v_1, \ldots, v_n, v_0\}$ having the origin v_0 of C as start and end node (as the line segment), and we split each request r_j into z_j many uniform requests (resp. single passengers), interpreted as subpaths $(x_j, y_j) \subseteq P$ (to obtain the (multi) set $\mathcal{I}$ of intervals). By construction, we have for the resulting interval graph $G_\sigma = \mathcal{G}(\mathcal{I})$:

- the clique number $\omega(\mathcal{G}(\mathcal{I}))$ corresponds to the maximum number of requests r_j in σ (counted with their multiplicities z_j) traversing a same edge of P,
- a coloring of G_σ corresponds to an assignment of places in the VIPA(s) to passengers.

Clearly one VIPA can serve all (uniform) requests from up-to Cap color classes in a single subtour traversing C. We can, thus, turn any coloring of G_σ into a feasible transportation schedule by

- waiting until time t_m in the origin v_0 (to ensure that all requests are released before they are served)
- selecting up to Cap many color classes and assigning the corresponding uniform requests (*i.e.* single passengers) to one VIPA, to be served within the same subtour traversing C, until all requests are served.

This leads to the following algorithm to compute optimal offline solutions for the tram mode:

Opt-Tram
Input: $\sigma = \{r_1, r_2, \ldots, r_m\}$, $C = \{v_o, v_1, \ldots, v_n\}$, Cap and k
Output: transportation schedule

(1) for each $r_j = (t_j, x_j, y_j, z_j) \in \sigma$: create z_j many intervals (x_j, y_j) to obtain $\mathcal{I}$,
(2) sort all intervals in $\mathcal{I}$ according to their left end points,

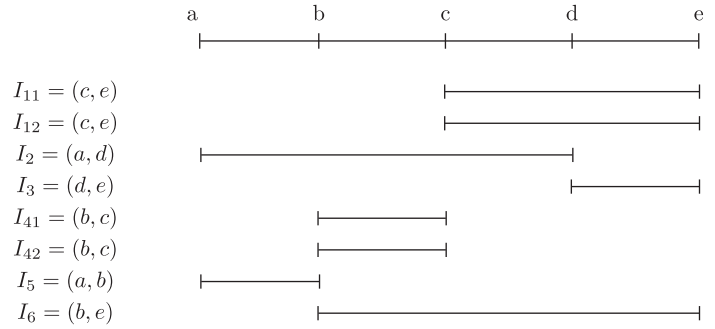
[4]Circuits can also be lines where one end is the origin and the VIPA can change its direction only in the two ends of the line.

(3) create the interval graph $\mathcal{G}(\mathcal{I})$ and apply the Greedy algorithm to color it,
(4) wait until t_m (the release date of the last request),
(5) as long as there are unserved requests:
select Cap many (or all remaining) color classes, assign the corresponding passengers to a VIPA and perform one subtour traversing C to serve them.

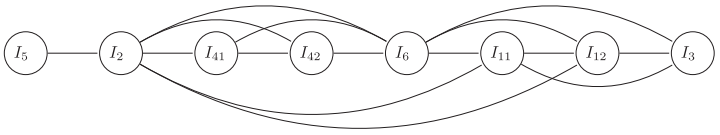
Example 4.1. Consider a circuit $C = (a, b, c, d, e)$ with origin a and one unit-speed server with capacity $\mathrm{Cap} = 2$ (*i.e.* a VIPA that travels 1 unit of length in 1 unit of time), and a sequence σ of 6 requests:

$$r_1 = (1, c, e, 2) \qquad r_4 = (4, b, c, 2)$$
$$r_2 = (2, a, d, 1) \qquad r_5 = (5, a, b, 1)$$
$$r_3 = (3, d, e, 1) \qquad r_6 = (6, b, e, 1)$$

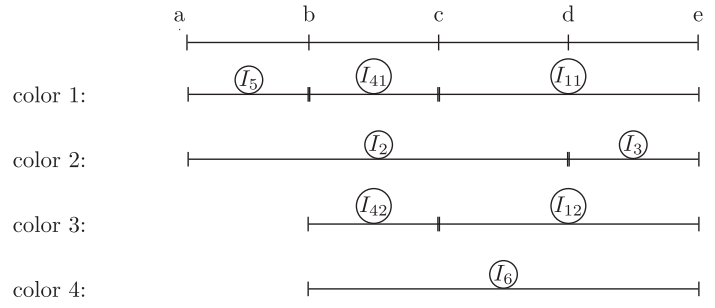
(1) For each $r_j \in \sigma$ we create z_j many intervals (x_j, y_j) to obtain $\mathcal{I}$:



(2) We sort all intervals in $\mathcal{I}$ according to their left end points: $I_5, I_2, I_{41}, I_{42}, I_6, I_{11}, I_{12}, I_3$.
(3) We create the interval graph $\mathcal{G}(\mathcal{I})$:



(4) We apply the Greedy algorithm to color it:



(5) As long as there are unserved requests, we randomly select 2 color classes, assign the corresponding passengers to a VIPA and perform one subtour traversing C to serve them, for instance:
- first round: color 1 and 2 $(r_5, r_2, r_{41}, r_{11}, r_3)$
- second round: color 3 and 4 (r_{42}, r_{12}, r_6).

Theorem 4.2. *Algorithm* Opt-Tram *provides a load-preemptive optimal offline solution w.r.t. minimizing the total tour length for VIPAs operating in tram mode on a circuit.*

Proof. By construction, splitting the requests r_j from σ according to their multiplicities z_j gives a (multi)set $\mathcal{I}$ of intervals where each of them stands for a uniform request of a single passenger. Accordingly, the clique number $w(\mathcal{G}(\mathcal{I}))$ of the resulting interval graph $\mathcal{G}(\mathcal{I})$ corresponds to the maximum number of passengers traversing a same edge e of the circuit C, that is:

$$\omega(\mathcal{G}(\mathcal{I})) = \max\left\{ \sum_{e\in(x_j,y_j),r_j\in\sigma} z_j; e \in C \right\}.$$

On the other hand, a coloring of $\mathcal{G}(\mathcal{I})$ corresponds to an assignment of places in the VIPA(s) to passengers, and subtours for the VIPA(s) can be easily created by repeatedly choosing Cap color classes and assigning the passengers colored that way to the Cap places of one VIPA. Clearly, at least $\left\lceil \frac{\omega(\mathcal{G}(\mathcal{I}))}{\text{Cap}} \right\rceil$ many such subtours are needed to serve all requests. The transportation schedule obtained is feasible because, by waiting until t_m to start any subtour, we ensure that all requests have been released before. As the Greedy coloring algorithm provides an optimal $\omega(\mathcal{G}(\mathcal{I}))$-coloring of $\mathcal{G}(\mathcal{I})$ by [12], we can guarantee to obtain a feasible transportation schedule performing the minimal number of subtours by always choosing Cap colors (except for the last subtour where we choose all remaining ones) so that the minimal total tour length equals

$$\text{OPT}(\sigma) = \left\lceil \frac{\omega(\mathcal{G}(\mathcal{I}))}{\text{Cap}} \right\rceil \cdot |C|.$$

The resulting solution is a load-preemptive transportation schedule because it cannot be ensured that all z_j passengers coming from the same request r_j are served by the same VIPA (even if $z_j \leq \text{Cap}$ holds). □

Remark 4.3.

- The minimal total tour length does not depend on the number of VIPAs used to serve all requests.
- Algorithm Opt-Tram is clearly polynomial because all the steps of the algorithm can be computed in polynomial time.
- By not selecting Cap color classes randomly to create subtours, it is possible to:
  - reduce load-preemption,
  - minimize the number of stops performed to let passengers leave/enter a VIPA,
  - handle the case of more than one VIPA,

  but the so modified algorithm is not necessarily polynomial anymore.

Online algorithms: For the online situation, we propose the following simple algorithm for VIPAs operating in tram mode on a circuit C:

SIR ("Stop If Requested")

- each VIPA waits in the origin of C; as soon as a request is released, a VIPA starts a full subtour in a given direction, thereby it stops at a station when a user requests to enter/leave.

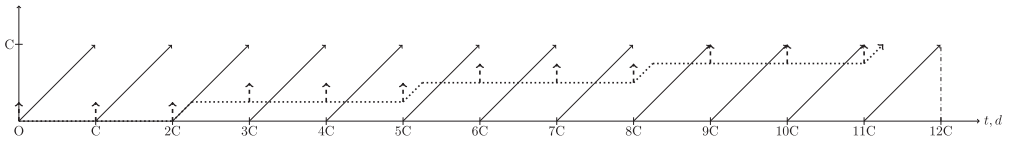


FIGURE 3. This figure illustrates the tour performed by SIR (solid arcs) and the adversary (dotted arcs) in order to serve the requests (dashed arcs) from Example 4.4 for $\mathrm{Cap} = 3, n = 3$ and $|C| = 4$.

In fact, in tram mode, the possible decisions of the VIPA are either to continue its tour or to wait at its current position for newly released requests. This can be used by the adversary to "cheat" an online algorithm, in order to maximize the ratio between the online and the optimal costs. Here, the strategy of the adversary is to force SIR to serve only one uniform request per subtour, whereas the adversary only needs a single subtour traversing C to serve all requests.

Example 4.4. Consider a circuit $C = (v_0, v_1, \ldots, v_n)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, and one unit-speed server with capacity Cap. The adversary releases a sequence σ of $\mathrm{Cap} \cdot n$ uniform requests that force SIR to perform one full round (subtour) of length $|C| = n + 1$ for each uniform request, whereas the adversary is able to serve all requests in a single subtour (fully loaded on each edge):

- Cap requests $r_j = ((j-1)|C|, v_0, v_1, 1)$ for $1 \leq j \leq \mathrm{Cap}$
- Cap requests $r_j = ((j-1)|C|, v_1, v_2, 1)$ for $\mathrm{Cap} + 1 \leq j \leq 2\mathrm{Cap}$

  $\vdots$
- Cap requests $r_j = ((j-1)|C|, v_{n-1}, v_n, 1)$ for $(n-1)\mathrm{Cap} + 1 \leq j \leq n\mathrm{Cap}$
- Cap requests $r_j = ((j-1)|C|, v_n, v_0, 1)$ for $n\mathrm{Cap} + 1 \leq j \leq (n+1)\mathrm{Cap}$

SIR starts its VIPA at time $t = 0$ to serve $r_1 = (0, v_0, v_1, 1)$ and finishes the first subtour of length $|C|$ without serving any further request. When the VIPA operated by SIR is back to the origin v_0, the second request $r_2 = (|C|, v_0, v_1, 1)$ is released and SIR starts at $t = |C| = n+1$ a second subtour of length $|C|$ to serve r_2, without serving any further request in this subtour. This is repeated for each request yielding $\mathrm{SIR}(\sigma) = \mathrm{Cap} \cdot |C| \cdot |C|$.

The adversary waits at the origin v_1 until $t = (\mathrm{Cap} - 1)|C|$ and serves all requests $r_1, \ldots, r_{\mathrm{Cap}}$ from v_0 to v_1. Then he waits until $t = (2\mathrm{Cap} - 1)|C|$ at v_1 and serves all requests $r_{\mathrm{Cap}+1}, \ldots, r_{2\mathrm{Cap}}$ from v_1 to v_2. This is repeated for all Cap requests from v_i to v_{i+1}, yielding $\mathrm{OPT}(\sigma) = |C|$. The tours performed by SIR and OPT are illustrated in Figure 3. Therefore, we obtain

$$\frac{\mathrm{SIR}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{\mathrm{Cap} \cdot |C| \cdot |C|}{|C|} = \mathrm{Cap} \cdot |C|$$

as a lower bound for the competitive ratio of SIR.

In the special case of the lunch scenario, we may consider VIPAs operating in tram mode on circuits, where each circuit has the restaurant as its distinguished origin. A sequence σ' containing the first Cap and the last Cap requests from the sequence presented in Example 4.4 shows that 2Cap is a lower bound on the competitive ratio of SIR, see Figure 4 for an illustration.

As for the morning resp. evening scenario, we consider VIPAs operating in tram mode on a circuit C where the parking is the distinguished origin of C. A sequence σ'' containing the first Cap resp. last Cap requests from the sequence presented in Example 4.4 shows that Cap is a lower bound on the competitive ratio of SIR, see Figures 5 and 6, respectively.

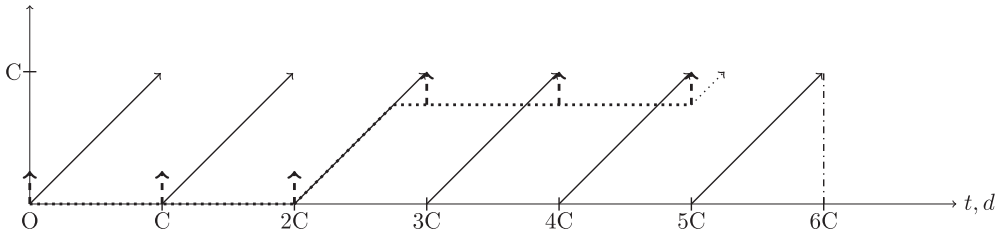


FIGURE 4. This figure illustrates the tour performed by SIR (solid arcs) and the adversary (dotted arcs) in order to serve the first Cap and the last Cap requests (dashed arcs) from the sequence presented in Example 4.4 for $\text{Cap} = 3, n = 3$ and $|C| = 4$. These requests satisfy the criterias of the lunch scenario.

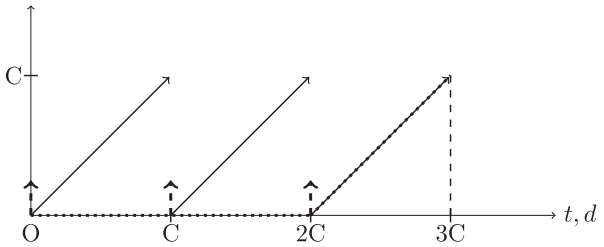


FIGURE 5. This figure illustrates the tour performed by SIR (solid arcs) and the adversary (dotted arcs) in order to serve the first Cap requests (dashed arcs) from the sequence presented in Example 4.4 for $\text{Cap} = 3, n = 3$ and $|C| = 4$. These requests satisfy the criterias of the morning scenario.

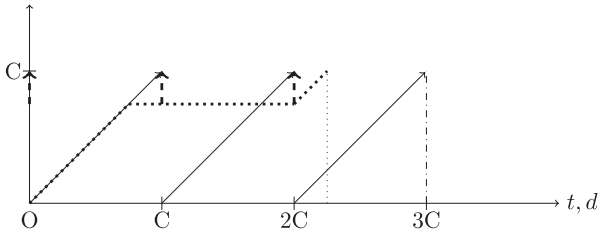


FIGURE 6. This figure illustrates the tour performed by SIR (solid arcs) and the adversary (dotted arcs) in order to serve the last Cap requests (dashed arcs) from the sequence presented in Example 4.4 for $\text{Cap} = 3, n = 3$ and $|C| = 4$. These requests satisfy the criterias of the evening scenario.

We can prove that the previously presented examples are indeed worst cases for SIR:

Theorem 4.5. *For one or several VIPAs with capacity* Cap *operating in tram mode on a circuit C with length $|C|$, SIR is w.r.t. the objective of minimizing the total tour length*

- $\text{Cap} \cdot |C|$*-competitive in general,*
- $2 \cdot \text{Cap}$*-competitive during the lunch scenario,*
- Cap*-competitive during the morning scenario resp. the evening.*

Proof. Recall that a transportation schedule is based on a coloring of the interval graph G_σ, whose nodes stand for passengers from σ, *i.e.* to the requests $r_j \in \sigma$ counted with their multiplicities z_j. The worst coloring of

G_σ is to assign different colors to all nodes, *i.e.* using $|G_\sigma| = \sum_{r_j \in \sigma} z_j$ many colors. The worst transportation schedule results if, in addition, each VIPA performs a separate subtour of length $|C|$ for each color (*i.e.* serving a single uniform request only per subtour), yielding $|G_\sigma| \cdot |C|$ as total tour length.

SIR can indeed be forced to show this behavior by releasing the requests accordingly (*i.e.* by using uniform requests with $z_j = 1$ each and with sufficiently large delay between t_j and t_{j+1}),

- in general: using the sequence σ from Example 4.4,
- during lunch: using the sequence σ' restricted to the first Cap and the last Cap requests $(t_j, v_0, v_1, 1)$ and $(t_j, v_n, v_0, 1)$ from the sequence σ presented in Example 4.4 as in Figure 4,
- during morning/evening: using the sequence σ'' restricted to the first Cap requests $(t_j, v_0, v_1, 1)$ (resp. the last Cap requests $(t_j, v_n, v_0, 1)$) from the sequence σ presented in Example 4.4, as Figure 5 (resp. Fig. 6) shows.

Furthermore, to maximize the ratio between this total tour length obtained by SIR and the optimal offline solution, we need to ensure that all requests in σ can be served with as few subtours of length $|C|$ as possible. This is clearly the case if all requests have length 1 and there are Cap many requests traversing the same edge of C s.t. a single subtour suffices to serve all of them (see again Example 4.4). This leads to

- $|G_\sigma| = |\sigma| = \text{Cap} \cdot |C|$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

$$\frac{\text{SIR}(\sigma)}{\text{OPT}(\sigma)} = \frac{\text{Cap} \cdot |C| \cdot |C|}{1 \cdot |C|} = \text{Cap} \cdot |C|$$

  is the maximum possible ratio between $\text{SIR}(\sigma)$ and $\text{OPT}(\sigma)$ taken over all possible sequences in general.
- $|G_{\sigma'}| = |\sigma'| = 2\text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

$$\frac{\text{SIR}(\sigma')}{\text{OPT}(\sigma')} = \frac{2 \cdot \text{Cap} \cdot |C|}{1 \cdot |C|} = 2 \cdot \text{Cap}$$

  is the maximum possible ratio between $\text{SIR}(\sigma)$ and $\text{OPT}(\sigma)$ taken over all possible sequences during the lunch.
- $|G_{\sigma''}| = |\sigma''| = \text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

$$\frac{\text{SIR}(\sigma'')}{\text{OPT}(\sigma'')} = \frac{\text{Cap} \cdot |C|}{1 \cdot |C|} = \text{Cap}$$

  is the maximum possible ratio between $\text{SIR}(\sigma)$ and $\text{OPT}(\sigma)$ taken over all possible sequences during the morning or evening.

□

Moreover, SIR can be adapted to follow the strategy of the adversary. For that, we propose two other algorithms for VIPAs operating in tram mode in the morning resp. evening:

SIF_M ("Start if fully loaded") for the morning scenario

- each VIPA waits in the parking until Cap passengers have entered,
- it starts a full round (as soon as it is fully loaded) and stops at stations where passengers request to leave.

SIF_E ("Start if fully loaded") for the evening scenario

- each VIPA waits in the parking until enough requests are released to reach Cap,
- it starts a full round and stops at stations where passengers request to enter and returns (fully loaded) to the parking.

We can ensure optimality for these two strategies:

Theorem 4.6. $\mathrm{SIF_M}$ (*resp.* $\mathrm{SIF_E}$) *is* 1*-competitive w.r.t. minimizing the total tour length for one or several VIPAs operating in tram mode on a circuit during the morning (resp. evening).*

Proof. Both variants of SIF are optimal, because due to the special request structure during the morning resp. evening, all requests traverse the first (resp. last) edge of P in the morning (resp. evening) s.t. G_σ becomes a clique. In other words, no two passengers can share a same place in a VIPA s.t. starting fully loaded from the origin (resp. returning fully loaded to the origin) indeed provides the optimal solution w.r.t. minimizing the total tour length. □

The two previous algorithms $\mathrm{SIF_E}$ and $\mathrm{SIF_M}$ can be merged together to obtain a version for the lunch scenario:

$\mathrm{SIF_L}$ ("Start if fully loaded on at least one arc")

- each VIPA waits in the restaurant until enough requests are released s.t. by serving these requests using one VIPA, the VIPA is fully loaded at least on one arc in the circuit.
- it starts a full round and stops at stations where passengers request to enter or leave and returns to the restaurant.

Theorem 4.7. $\mathrm{SIF_L}$ *is* 2*-competitive w.r.t. minimizing the total tour length for one or several VIPAs operating in tram mode on a circuit during the lunch.*

Proof. Due to the special request structure during the lunch, all requests start and end in the restaurant and, thus, traverse the first and the last edge of P s.t. G_σ consists of two cliques Q_1 and Q_2 resulting from all uniform requests traversing the first and the last edge, respectively.
The worst transportation schedule of $\mathrm{SIF_L}$ results if the requests are released in a way that $\mathrm{SIF_L}$ never serves a request from Q_1 with one from Q_2 together, therefore by each subtour of length $|C|$ performed by $\mathrm{SIF_L}$ Cap requests are served either from the clique Q_1 or from Q_2, yielding $\mathrm{SIF_L}(\sigma) = \lceil \frac{|G_\sigma|}{\mathrm{Cap}} \rceil \cdot |C|$ as total tour length.
In order to maximize the ratio, OPT needs to serve as many requests as possible using the least total tour length possible. OPT always combines Cap requests from Q_1 with Cap requests from Q_2 and serves them together by performing a subtour of length $|C|$. In addition, to avoid not fully loaded moves for OPT, the adversary chooses $|Q_1| = |Q_2|$ and as a multiple of Cap which leads to $\mathrm{OPT}(\sigma) = \frac{|G_\sigma|}{2\mathrm{Cap}} \cdot |C|$, therefore

$$\frac{\mathrm{SIF_L}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{\frac{|G_\sigma|}{\mathrm{Cap}} \cdot |C|}{\frac{|G_\sigma|}{2\mathrm{Cap}} \cdot |C|} = 2$$

is the maximum possible ratio between $\mathrm{SIF_L}(\sigma)$ and $\mathrm{OPT}(\sigma)$ taken over all possible sequences on a circuit of length $|C|$ during the lunch. □

4.2. Elevator mode

The elevator mode is a less restricted operation mode where one VIPA runs on a predefined line and can change its direction at any station of this line to move towards a requested station. One end of this line is distinguished as origin O (say, the "left" end).
Optimal offline solution: In order to obtain the optimal offline solution $\mathrm{OPT}(\sigma)$ w.r.t. minimizing the total tour length, we compute a min cost flow in a suitable network. Given a line $L = (v_0, \ldots, v_n)$ with origin v_0 as a subnetwork, capacity Cap of a VIPA, and a request sequence σ with requests $r_j = (t_j, x_j, y_j, z_j)$.
In the sequel, we distinguish in which direction an edge $v_i v_{i+1}$ of L is traversed and speak of the up arc (v_i, v_{i+1}) and the down arc (v_{i+1}, v_i). In order to construct the network, we proceed as follows:

- Neglect the release dates t_j and only consider the loads of the requests z_j, their origins x_j and their destinations y_j.

– Partition the requests into two subsets:
  • U of "up-requests" $r_j \in \sigma$ with $x_j < y_j$,
  • D of "down-requests" $r_j \in \sigma$ with $x_j > y_j$.
– Determine the loads of all up arcs (v_i, v_{i+1})) or down arcs (v_{i+1}, v_i) of the line L as a weighted sum of the load of all request-paths (x_j, y_j) containing this arc:
  • $\text{load}(v_i, v_{i+1}) = \sum_{(v_i,v_{i+1})\in(x_j,y_j),x_j<y_j} z_j \ \forall i \in \{0, n-1\}, \forall r_j \in U$
  • $\text{load}(v_{i+1}, v_i) = \sum_{(v_{i+1},v_i)\in(x_j,y_j),x_j>y_j} z_j \ \forall i \in \{0, n-1\}, \forall r_j \in D$
– Determine the "multiplicities" m of all up/down arcs: in order to serve all the requests in σ, each arc (v_i, v_{i+1}) must be visited $m_{(i,i+1)} = \lceil \frac{\text{load}(v_i,v_{i+1})}{\text{Cap}} \rceil$ times and each arc (v_{i+1}, v_i) must be visited $m_{(i+1,i)} = \lceil \frac{\text{load}(v_{i+1},v_i))}{\text{Cap}} \rceil$ times. In case the multiplicity $m_{(i,i+1)}$ resp. $m_{(i+1,i)}$ is equal to zero, then the corresponding up resp. down arc is removed.

Now we build a network $G_E = (V_E, A_E)$, where

– The node set $V_E = V^{\text{up}(o)} \cup V^{\text{up}(d)} \cup V^{\text{down}(o)} \cup V^{\text{down}(d)}$ contains
  • the origin nodes of all up arcs where the multiplicity is different from zero in $V^{\text{up}(o)}$,
  • the destination nodes of all up arcs where the multiplicity is different from zero in $V^{\text{up}(d)}$,
  • the origin nodes of all down arcs where the multiplicity is different from zero in $V^{\text{down}(o)}$,
  • the destination nodes of all down arcs where the multiplicity is different from zero in $V^{\text{down}(d)}$,
  • the origin v_0 of the line L as source s and as sink t.
– The arc set $A_E = A_s \cup A_U \cup A_D \cup A_L \cup A_t$ is composed of:
  • source arcs from the source s to all $v_i^{\text{up}(o)} \in V^{\text{up}(o)}$ and all $v_i^{\text{down}(o)} \in V^{\text{down}(o)}$ in A_s,
  • up arcs $(v_i^{\text{up}(o)}, v_{i+1}^{\text{up}(d)})$ whenever $m_{(i,i+1)} \neq 0$ in A_U,
  • down arcs $(v_{i+1}^{\text{down}(o)}, v_i^{\text{down}(d)})$ whenever $m_{(i+1,i)} \neq 0$ in A_D,
  • link arcs in A_L going from all $v_i^{\text{up}(d)} \in V^{\text{up}(d)}$ to all $v_i^{\text{down}(o)} \in V^{\text{down}(o)}$, and from all $v_i^{\text{down}(d)} \in V^{\text{down}(d)}$ to all $v_i^{\text{up}(o)} \in V^{\text{up}(o)}$,
  • sink arcs from all $v_i^{\text{up}(d)} \in V^{\text{up}(d)}$ and from all $v_i^{\text{down}(d)} \in V^{\text{down}(d)}$ to the sink t in A_t.

Accordingly, the objective function considers costs $d(a) = d(u, v)$ for the flow f on all arcs $a = (u, v)$ in A_E, where $d(u, v)$ is the length of a shortest path from u to v in the line L. To correctly initialize the system, we use the source node s as source for the flow and the sink node t as its destination. For all internal nodes, we use normal flow conservation constraints. We require a flow on all up/down arcs $f(a) = m(a)$ for all $a \in \{A_U \cup A_D\}$, see constraint (4.1e). We finally add the constraint (4.1d) to eliminate all possible isolated cycles that the flow may contain (since the network contains directed cycles).

This leads to a Min-Cost Flow Problem, whose output is a subset of arcs needed to form a transportation schedule for a metric task system, whose tasks are induced by the requests. The corresponding integer linear program is as follows:

$$\min \sum_{a\in A_E} d(a)f(a) \tag{4.1a}$$

$$\text{s.t.} \sum_{a\in\delta^-(s)} f(a) = 1 \tag{4.1b}$$

$$\sum_{a\in\delta^-(v)} f(a) = \sum_{a\in\delta^+(v)} f(a) \qquad \forall v \neq \{s, t\} \tag{4.1c}$$

$$\sum_{a\in\delta(W)} f(a) \geq 2 \qquad \forall W \subset V_E \setminus \{s, t\}, 2 \leq |W| \leq |V_E| - 3 \tag{4.1d}$$

$$f(a) = m(a) \qquad \forall a \in \{A_U \cup A_D\} \tag{4.1e}$$

$$f(a) \in \mathbb{Z}_+ \tag{4.1f}$$

where $\delta^-(v)$ denotes the set of outgoing arcs of v, $\delta^+(v)$ denotes the set of incoming arcs of v and $\delta(W)$ denotes the set of incoming or outgoing arcs (u,v) of W s.t. $u \in W$ and $v \notin W$ or $u \notin W$ and $v \in W$. The time required to compute the integer linear program grows in proportion to $2^{|V|}$ due to the constraint (4.1d) that eliminates all possible isolated cycles, and, hence, it may grow exponentially. However, this integer linear program can be computed in reasonable time provided that the number V of nodes in the original network (the line) is small.

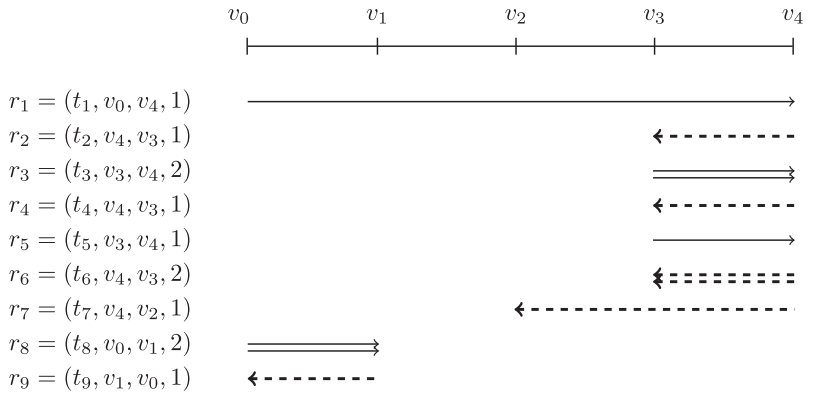
Remark 4.8. The family of constraints (4.1d) can be generated and then each inequality is separated to verify if it is violated or not. However, due to their exponential number, the process of separation in order to verify if a solution satisfies all constraints is exponential. Since the number of subtour elimination constraints is exponential, we may firstly compute the integer linear program without the constraints (4.1d). Then we check if there is an isolated cycle in the solution obtained, if yes, we add only the constraint (4.1d) using the nodes of this isolated cycle. This procedure is repeated until a solution without isolated cycles is found.

Finally, the flow in the time-expanded network is interpreted as a transportation schedule. The tracks of the VIPA over time can be recovered from the flow $f(a)$ on the arcs by standard flow decomposition, see [1]. Hereby, a flow $f(a)$ on an arc $a=(u,v)$ corresponds to a move of a VIPA on this arc. Based on the flow values, one can construct a unique path from source s to sink t traversing all arcs a with positive flow exactly $f(a)$ times. This shows that the optimal solution of system (4.1) corresponds to a transportation schedule with minimal total tour length for the offline problem behind the elevator mode.

Example 4.9. Consider a line $L=(v_0,\ldots,v_n)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, with a set σ of 9 requests shown in Figure 7, and a VIPA with capacity $\mathrm{Cap}=2$. The resulting network $G_E=(V_E,A_E)$ of the line presented in this example is illustrated in Figure 8. The optimal offline solution, the transportation schedule of the VIPA with a minimal total tour length, is obtained by computing the presented integer linear program for a min-cost flow problem in the constructed network $G_E=(V_E,A_E)$.

Online algorithm: An algorithm MRIN ("Move Right If Necessary") has been proposed for the Online-Traveling Salesman Problem (where no transports have to be performed, but only points to be visited) on a line and analyzed w.r.t. minimizing the makespan [4], giving a competitive ratio of 3/2. We generalize MRIN to the Pickup and Delivery Problem and analyze it w.r.t. minimizing the total tour length. In elevator mode, the server (VIPA) has the choice to continue its tour in the current direction, to wait at its current position or to change its driving direction. Accordingly, we propose the algorithm MAIN. The adversary can again "cheat" the strategy of MAIN as the following example shows.

MAIN ("Move Away If Necessary")
Input : request sequence σ, line L with origin v_O, Cap
Output: tour on L to serve all requests in σ
initial server position $s := v_O$;
initial set of currently waiting requests (already released but not yet served requests) $\sigma' := \{r_j \in \sigma : t_j = 0\}$;
while $\sigma' \neq \varnothing$ **do**
  determine the subset σ'_{up} of requests $r_j=(t_j,x_j,y_j,z_j) \in \sigma'$ with $s \le x_j \le y_j$;
  if $\sigma'_{\mathrm{up}} \neq \varnothing$ **then**
    Serve all requests (or up to Cap passengers) in σ'_{up} (moving away from v_O to furthest destination y_k among all $r_j \in \sigma'_{\mathrm{up}}$);
  end
  else
    determine subset σ'_{down} of requests $r_j=(t_j,x_j,y_j,z_j) \in \sigma'$ with $x_j > y_j$;
    serve all requests (or up to Cap passengers) in σ'_{down} while moving to the origin;
  end
  update s and σ' (remove all served requests, add all newly released requests);
end



$load(uparcs)$	3	1	1	4
$load(downarcs)$	1	0	1	5
$m(uparcs) = \lceil \frac{load}{\text{Cap}} \rceil$	2	1	1	2
$m(downarcs) = \lceil \frac{load}{\text{Cap}} \rceil$	1	0	1	3

FIGURE 7. This figure illustrates the line $L = (v_0, \ldots, v_n)$ with origin v_0, and a set σ of 9 requests and a VIPA with capacity $\text{Cap} = 2$. The requests are partitioned into two subsets "up-requests" (solid arcs) and "down-requests" (dashed arcs). Each arc represents a load of 1, for example r_3 is represented by 2 arcs going from v_3 to v_4. The loads of all up arcs (all arcs (v_i, v_{i+1})) or down arcs (all arcs (v_{i+1}, v_i)) of the line L are shown in the first and second row of the table. Then the third and the forth rows contain the "multiplicities" m of all up/down arcs.

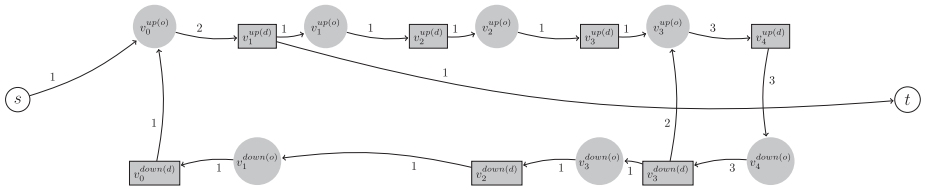


FIGURE 8. This figure illustrates the flow computed by the presented integer linear program for a min-cost flow problem in the network $G_E = (V_E, A_E)$ of Example 4.9. The values above the arcs correspond to the value of the flow $f(a)$ or the number of times the VIPA traverses the arc a in the transportation schedule.

Example 4.10. Consider a line $L = (v_0, \ldots, v_n)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, and one unit-speed server with capacity Cap. The adversary releases a sequence σ of uniform requests that force MAIN to leave the origin of the line and perform a subtour of a certain distance for each request, whereas the adversary is able to serve all requests in a single subtour of length $2|L|$:

The first block σ_1 of $n \cdot \text{Cap}$ requests:

$r_1 = (0, v_0, v_1, 1)$
$r_j = (t_{j-1} + 2d(v_0, v_1), v_0, v_1, 1)$ for $2 \leq j \leq \text{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_1), v_1, v_2, 1)$ for $j = \text{Cap} + 1$
$r_j = (t_{j-1} + 2d(v_0, v_2), v_1, v_2, 1)$ for $\text{Cap} + 2 \leq j \leq 2\text{Cap}$

$$r_j = (t_{j-1} + 2d(v_0, v_{n-1}), v_{n-1}, v_n, 1) \text{ for } j = (n-1)\text{Cap} + 1$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_{n-1}, v_n, 1) \text{ for } (n-1)\text{Cap} + 2 \leq j \leq n\text{Cap}$$

The second block σ_2 of $2\ell \cdot \text{Cap}$ requests

$$r_j = (t_{j-1} + 2d(v_0, v_n), v_n, v_{n-1}, 1) \text{ for } n\text{Cap} + 1 \leq j \leq (n+1)\text{Cap}$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_{n-1}, v_n, 1) \text{ for } (n+1)\text{Cap} + 1 \leq j \leq (n+2)\text{Cap}$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_n, v_{n-1}, 1) \text{ for } (n+2\ell-2)\text{Cap} + 1 \leq j \leq (n+2\ell-1)\text{Cap}$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_{n-1}, v_n, 1) \text{ for } (n+2\ell-1)\text{Cap} + 1 \leq j \leq (n+2\ell)\text{Cap}$$

The third block σ_3 of $n \cdot \text{Cap}$ requests:

$$r_j = (t_{j-1} + 2d(v_0, v_n), v_n, v_{n-1}, 1) \text{ for } (n+2\ell+1)\text{Cap} + 1 \leq j \leq (n+2\ell+2)\text{Cap}$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_{n-1}, v_{n-2}, 1) \text{ for } j = (n+2\ell+2)\text{Cap} + 1$$
$$r_j = (t_{j-1} + 2d(v_0, v_n), v_n, v_{n-1}, 1) \text{ for } (n+2\ell+2)\text{Cap} + 2 \leq j \leq (n+2\ell+3)\text{Cap}$$
$$r_j = (t_{j-1} + 2d(v_0, v_2), v_1, v_0, 1) \text{ for } j = (2n+2\ell-1)\text{Cap} + 1$$
$$r_j = (t_{j-1} + 2d(v_0, v_1), v_1, v_0, 1) \text{ for } (2n+2\ell-1)\text{Cap} + 2 \leq j \leq (2n+2\ell)\text{Cap}$$

MAIN starts its VIPA at time $t = 0$ to serve $r_1 = (0, v_0, v_1, 1)$ and finishes the first subtour of length $2d(v_0, v_1) = 2$ without serving any further request. When the VIPA operated by MAIN is back to the origin v_0, the second request $r_2 = (2, v_0, v_1, 1)$ is released and MAIN starts at $t = 2$ a second subtour of length 2 to serve r_2, without serving any further request in this subtour. This is repeated for each request until serving the first block of $n \cdot \text{Cap}$ requests yielding

$$\text{MAIN}(\sigma_1) = 2 \cdot \text{Cap} \sum_{1 \leq i \leq n} i = \text{Cap} \cdot |L| \cdot (|L| + 1)$$

Then at $t = t_{n\text{Cap}+1}$ MAIN starts to serve $r_{n\text{Cap}+1}$ from v_n to v_{n-1} and performs a subtour of length $2d(v_0, v_n) = 2|L|$. When the VIPA operated by MAIN is back to the origin v_0, the request $r_{n\text{Cap}+2}$ is released and MAIN performs a new subtour of length $2|L|$ to serve it. This is repeated for each request until serving the second block of $\ell \cdot 2\text{Cap}$ requests yielding

$$\text{MAIN}(\sigma_2) = 2\ell \cdot 2\text{Cap}|L|.$$

Finally in order to serve the third block MAIN has the same behavior as to serve the first block of requests yielding

$$\text{MAIN}(\sigma_3) = 2 \cdot \text{Cap} \sum_{1 \leq i \leq n} i = \text{Cap} \cdot |L| \cdot (|L| + 1).$$

Therefore

$$\text{MAIN}(\sigma) = \text{Cap} \cdot |L| \cdot (|L| + 1) + 2\ell \cdot 2\text{Cap}|L| = (|L| + 1 + 2\ell) \cdot 2|L| \cdot \text{Cap}.$$

The adversary waits at the origin v_0 until $t = t_{\text{Cap}}$ and serves all requests $r_1, \ldots, r_{\text{Cap}}$ from v_0 to v_1. Then he waits until $t = t_{2\text{Cap}}$ at v_1 and serves all requests $r_{\text{Cap}+1}, \ldots, r_{2\text{Cap}}$ from v_1 to v_2. This is repeated for all Cap requests from v_i to v_{i+1} until the adversary arrives to v_n. OPT served the first block of $n \cdot \text{Cap}$ requests with a total tour length equal to $|L|$. Then the adversary begins to oscillate his VIPA between v_n and v_{n-1} and serves each time Cap requests, this is repeated 2ℓ times leading to a total tour length for σ_2 equal to 2ℓ. Finally the

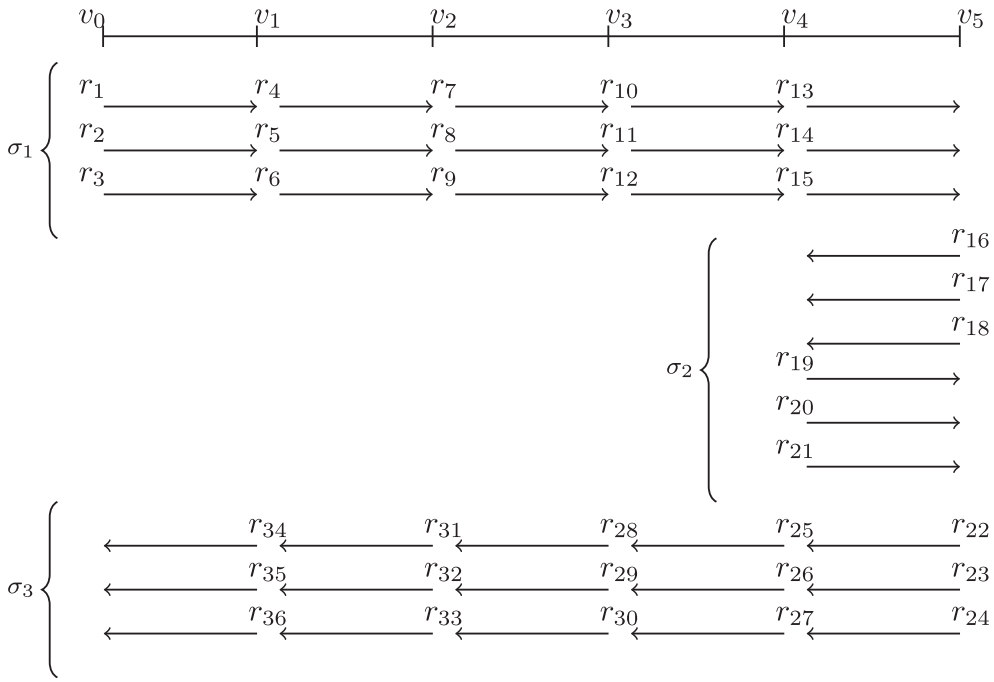


FIGURE 9. This figure illustrates the set $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3$ of requests (arcs under the line $L = (v_0, \ldots, v_5)$ with origin v_0) from Example 4.10 for $\text{Cap} = 3, n = 5$ and $\ell = 1$.

adversary follows the other direction and waits each time until Cap requests are released to serve them, for all Cap requests from v_i to v_{i-1} until reaching v_0, yielding $\text{OPT}(\sigma) = 2|L| + 2\ell$. Therefore, we obtain

$$\frac{\text{MAIN}(\sigma)}{\text{OPT}(\sigma)} = \frac{2 \cdot \text{Cap} \cdot |L| \cdot (|L| + 1) + (\ell \cdot 2 \cdot \text{Cap} \cdot 2|L|)}{2(|L| + \ell)} = \frac{2 \cdot \text{Cap} \cdot |L| \cdot (|L| + 1 + 2\ell)}{2(|L| + \ell)}$$

$$= \text{Cap} \cdot |L| + \frac{(1+\ell)}{|L| + \ell} \text{Cap} \cdot |L| \underset{\ell \to +\infty}{\longrightarrow} 2\text{Cap} \cdot |L|$$

as a lower bound for the competitive ratio of MAIN.

We can determine an upper bound for the competitive ratio of MAIN close to the ratio obtained by the previous example:

Theorem 4.11. *MAIN is* $2\text{Cap} \cdot |L|$*-competitive w.r.t. minimizing the total tour length for one VIPA operating in elevator mode on a line* L *with length* $|L|$.

Proof. The worst transportation schedule results if all requests are uniform and the VIPA operated by MAIN performs a separate subtour serving a single request $r_j = (t_j, x_j, y_j, 1)$ each time the VIPA leaves the origin v_0 of the line, yielding $\sum_{r_j \in \sigma} 2\max(d(v_0, x_j), d(v_0, y_j))$ as total tour length.
To maximize the ratio between the total tour length obtained by MAIN and the optimal offline solution, we need to ensure that

- we do not have a move with a load less than the capacity Cap of the VIPA in the transportation schedule of OPT;
- all requests in σ can be served with as few and as short subtours as possible in OPT.

The worst ratio of subtours can be obtained when

- OPT oscillates fully loaded between two neighbored nodes of L,
- MAIN is forced to traverse the whole line twice per passenger, *i.e.* oscillates between v_0 and v_n.

For that, v_n needs to be either origin or destination of each request, and the delay between the release dates needs to be sufficiently large. This can be achieved with subsequence σ_2 from Example 4.10, with ℓ blocks each consisting of

- Cap consecutive uniform requests from v_n to v_{n-1}, alternated by
- Cap consecutive uniform requests from v_{n-1} to v_n,

always with a delay $2|L|$ between the release dates of any two requests r_j and r_{j+1}. We obtain $\mathrm{OPT}(\sigma_2) = 2\ell$ and $\mathrm{MAIN}(\sigma_2) = \ell \cdot 2 \cdot \mathrm{Cap} \cdot 2 \cdot |L|$ which leads to the studied subtour ratio of

$$\frac{\mathrm{MAIN}(\sigma_2)}{\mathrm{OPT}(\sigma_2)} = \frac{\ell \cdot 2 \cdot \mathrm{Cap} \cdot 2 \cdot |L|}{2\ell} = 2 \cdot \mathrm{Cap} \cdot |L|.$$

However, this ratio so far neglects the initial and final server position v_0 for the VIPA operated by OPT. The requirement of starting and ending the tour in v_0 leads to a total tour length for OPT of

$$\mathrm{OPT}(\sigma) = |L| \cdot 2\ell \cdot |L|.$$

In order to maximize the ratio of the complete tours, the adversary releases more requests to ensure that the VIPA operated by

- OPT can arrive at v_n (resp. return from v_n to v_0) fully loaded on each arc,
- MAIN is forced to oscillate between v_0 and the destination y_j (resp. the origin x_j) of each uniform request r_j.

This can be achieved with the subsequences σ_1 and σ_3 from Example 4.10 with

- Cap consecutive uniform requests from v_i to v_{i+1} for each $0 \leq i < n$ and
- Cap consecutive uniform requests from v_i to v_{i-1} for each $n \geq i \geq 1$,

always with delay $2 \cdot d(v_0, y_j)$ resp. $2 \cdot d(x_j, v_0)$ between the release dates of any two requests r_j and r_{j+1} within these subsequences. We obtain (as in Example 4.10) that

$$\mathrm{MAIN}(\sigma_1) = \mathrm{MAIN}(\sigma_3) = 2 \cdot \mathrm{Cap} \sum_{1 \leq i \leq n} i = \mathrm{Cap} \cdot |L| \cdot (|L| + 1).$$

This finally leads to

$$\frac{\mathrm{MAIN}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{2 \cdot \mathrm{Cap} \cdot |L| \cdot (|L| + 1) + (\ell \cdot 2 \cdot \mathrm{Cap} \cdot 2|L|)}{2(|L| + \ell)} = \frac{2 \cdot \mathrm{Cap} \cdot |L| \cdot (|L| + 1 + 2\ell)}{2(|L| + \ell)}$$

$$= \mathrm{Cap} \cdot |L| + \frac{(1+\ell)}{|L| + \ell} \mathrm{Cap} \cdot |L| \underset{\ell \to +\infty}{\longrightarrow} 2\mathrm{Cap} \cdot |L|$$

as the maximum possible ratio between $\mathrm{MAIN}(\sigma)$ and $\mathrm{OPT}(\sigma)$ taken over all possible sequences on a line L. □

Concerning the lunch scenario, we may consider VIPAs operating in elevator mode on lines, where each line has the restaurant as its distinguished origin. A sequence σ' containing the first Cap requests of the first block σ_1 and the last Cap requests from the third block σ_3 from the sequence presented in Example 4.10 shows that $2 \cdot \mathrm{Cap}$ is a lower bound on the competitive ratio of MAIN. As for the morning resp. evening scenario, we may consider VIPAs operating in elevator mode on lines, where each line has a parking as its distinguished origin. A sequence σ' containing the first Cap requests of the first block σ_1 resp. the last Cap requests from the third block σ_3 from the sequence presented in Example 4.10 shows that Cap is a lower bound on the competitive ratio of MAIN. We can show that these examples are the worst cases for MAIN during lunch, morning and evening:

Theorem 4.12. *For one VIPA with capacity* Cap *operating in elevator mode on a line, MAIN is w.r.t. the objective of minimizing the total tour length*

- $2 \cdot$ Cap-*competitive during the lunch scenario,*
- Cap-*competitive during the morning resp. the evening scenario.*

Proof. The worst transportation schedule results if the VIPA operated by MAIN performs a separate subtour serving a single uniform request $r_j = (t_j, v_0, v_1, 1)$ or $r_j = (t_j, v_1, v_0, 1)$ each time the VIPA leaves the origin v_0 of the line, yielding $\sum_{r_j \in \sigma} 2d(v_0, v_1)$ as total tour length. MAIN can indeed be forced to show this behavior by releasing the requests accordingly (*i.e.* by using requests with $z_j = 1$ each and with sufficiently large delay between t_j and t_{j+1}). In order to maximize the ratio between the total tour length obtained by MAIN and the optimal offline solution, we need to ensure that

- we do not have a move from or to the origin with a load less than the capacity Cap of the VIPA in the transportation schedule of OPT. For that, the adversary releases
  - during the lunch Cap many requests traversing the same arc. Whereas MAIN traverses $d(v_0, v_1)$ twice to serve a request $r_j = (t_j, v_0, v_1, 1)$ or $r_j = (t_j, v_1, v_0, 1)$, OPT travels $d(v_0, v_1)$ once to serve the request and can share it with Cap $- 1$ others.
  - during the morning/evening Cap many requests traversing the same arc. Whereas MAIN traverses $d(v_0, v_1)$ twice to serve a request $r_j = (t_j, v_0, v_1, z_j)$ resp. $r_j = (t_j, v_1, v_0, z_i)$, OPT travels the same distance to serve the request but can share it with Cap $- 1$ others.
- all requests in σ can be served with as few and as short subtours as possible in OPT. For that, the adversary releases
  - during the lunch a sequence σ of 2Cap requests: Cap many requests $v_0 \to v_1$ followed by Cap many requests $v_1 \to v_0$. Therefore we obtain

$$\mathrm{MAIN}(\sigma) = \sum_{r_j \in \sigma} 2d(v_0, v_1) = 2 \cdot \mathrm{Cap} \cdot 2d(v_0, v_1) \text{ and } \mathrm{OPT}(\sigma) = 2d(v_0, v_1)$$

$$\text{s.t. } \frac{\mathrm{MAIN}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{2 \cdot \mathrm{Cap} \cdot 2d(v_0, v_1)}{2d(v_0, v_1)} = 2\mathrm{Cap}$$

    is the maximum possible ratio between $\mathrm{MAIN}(\sigma)$ and $\mathrm{OPT}(\sigma)$ taken over all possible sequences on a line during the lunch.
  - during the morning/evening a sequence σ of Cap requests: Cap many requests $v_0 \to v_1$ resp. Cap many requests $v_1 \to v_0$. Therefore we obtain

$$\mathrm{MAIN}(\sigma) = \sum_{r_j \in \sigma} 2d(v_0, v_1) = \mathrm{Cap} \cdot 2d(v_0, v_1) \text{ and } \mathrm{OPT}(\sigma) = 2d(v_0, v_1)$$

$$\text{s.t. } \frac{\mathrm{MAIN}(\sigma)}{\mathrm{OPT}(\sigma)} = \frac{\mathrm{Cap} \cdot 2d(v_0, v_1)}{2d(v_0, v_1)} = \mathrm{Cap}$$

    is the maximum possible ratio between $\mathrm{MAIN}(\sigma)$ and $\mathrm{OPT}(\sigma)$ taken over all possible sequences on a line during the morning resp. evening.

□

4.3. Computational results

This section deals with computational experiments for the proposed online algorithms. In fact, due to the very special request structures of the previously presented worst case instances, we can expect a better behavior of the proposed online algorithms in average. The computational results presented in Tables 1 and 2 support

TABLE 1. This table shows the computational results for several test instances of the algorithms SIR and SIF_L in comparison to the value of the optimal solution.

		SIR (morning) $c = \text{Cap}$			SIR (Evening) $c = \text{Cap}$			SIR (Lunch) $c = 2\text{Cap}$			SIF_L (Lunch) $c = 2$			SIR (General) $c = \text{Cap} \cdot \lvert C \rvert$			
m	Cap	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	c	TTL	$\frac{\text{TTL}}{\text{OPT}}$	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	c
5	1	125	125	1,00	125	125	1,00	112,5	93,75	1,20	2	100	1,07	143,75	106,25	1,35	25
5	5	86,67	25	3,47	100	25	4,00	75	25	3,00	10	25	1,00	131,25	25	5,25	125
5	10	86,67	25	3,47	81,25	25	3,25	68,75	25	2,75	20	25	1,00	87,5	25	3,50	250
20	1	500	500	1,00	500	500	1,00	418,75	337,5	1,24	2	400	1,19	593,75	337,5	1,76	25
20	5	125	100	1,25	113,3	100	1,13	206,25	75	2,75	10	81,25	1,08	287,5	75	3,83	125
20	10	106,25	50	2,13	110,4	50	2,21	206,25	43,75	4,71	20	56,25	1,29	268,75	50	5,38	250
200	1	5000	5000	1,00	5000	5000	1,00	4300	2818,75	1,53	2	2968,75	1,05	5400	2706,25	2,00	25
200	5	1270,8	1000	1,27	2881,25	1000	2,88	2125	562,5	3,78	10	600	1,07	2600	550	4,73	125
200	10	585,5	500	1,17	575,45	401,5	1,43	1512,5	275	5,50	20	468,75	1,70	2168,75	275	7,89	250

TABLE 2. This table shows the computational results for several test instances of the algorithm MAIN in comparison to the value of the optimal solution.

		MAIN (Morning) $c = \text{Cap}$			MAIN (Evening) $c = \text{Cap}$			MAIN (Lunch) $c = 2\text{Cap}$				MAIN (General) $c = 2\text{Cap} \cdot \lvert L \rvert$			
m	Cap	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	c	TTL	OPT	$\frac{\text{TTL}}{\text{OPT}}$	c
5	1	97,3	97,3	1	85,5	57,3	1,49	61,15	29,5	2,07	2	56,3	36	1,56	64
5	5	55,5	25,78	2,15	55,5	25,78	2,15	74,76	37	2,02	10	42,45	27	1,57	320
5	10	55,5	25,78	2,15	55,5	25,78	2,15	74,76	37	2,02	20	42,45	27	1,57	640
20	1	327,2	327,2	1	365,2	327,2	1,12	222,35	177	1,26	2	140,8	120,5	1,17	64
20	5	135,8	135,8	1	135,8	73	1,86	78,4	67	1,17	10	95,54	43	2,22	320
20	10	78,3	47,5	1,65	108,3	47,5	2,28	75,58	34	2,22	20	80,15	33,5	2,39	640
200	1	3394,8	3394,8	1	3434,8	3394,8	1,01	4918,75	1855,5	2,65	2	5023,25	1435,5	3,5	64
200	5	1050,5	694,8	1,51	1050,5	694,8	1,51	2696,42	368,5	7,32	10	1662,4	292	5,69	320
200	10	760,8	362,34	2,1	860,8	362,34	2,38	1166,25	185	6,3	20	180,8	120,5	1,5	640

this expectation, they compare the total tour length TTL computed by the online algorithms with the optimal offline solution. The computations use instances based on the network from the industrial site of Michelin and randomly generated request sequences resembling typical instances that occurred during the experimentation [14]. The computations are performed with the help of a simulation tool developed by Yan Zhao [18]. The instances use subnetworks as a circuit or a line depending on the mode and the algorithm with 1–5 VIPAs, 5–200 requests, 1–12 as the maximum load z_j of a request. For every parameter set we created 6 test instances. In these tables, the instances are grouped by the number of requests and the capacity of the VIPA. The average results of the instances are shown. The operating system for all tests is Linux (CentOS with kernel version 2.6.32). The algorithms SIR, SIF_L, MAIN and OPT-TRAM have been implemented in Java. For solving the integer linear program to get the optimal solution for the elevator mode, we use Gurobi 8.21. Note that the average runtime of the heuristics is not shown as it never exceeds one second. Computing the offline solutions is not relevant in practice, therefore the time needed to compute the optimal offline solutions is not shown.

In Table 1, the instances are grouped by the number of requests (1st column) and the capacity (2nd column). Furthermore the values of the total tour length TTL found by SIR with morning, evening and lunch instances are shown in comparison with the optimal solution OPT and the ratio between them. Then the values of the total tour length TTL found by SIF_L with lunch instances are shown in comparison with the optimal solution OPT and the ratio between them. Finally, the results for instances respecting the general scenario are shown: the total tour length TTL found by SIR, the optimal solution OPT and the ratio between them. The competitive ratio c is shown for each of the scenarios, it is always greater than $\frac{\text{TTL}}{\text{OPT}}$ unless $c = \frac{\text{TTL}}{\text{OPT}} = 1$. In these test instances the length of the circuit used is equal to $\lvert C \rvert = 25$.

In Table 2, the instances are grouped by number of requests (1st column) and the capacity (2nd column). First the values of the total tour length TTL found by MAIN for the instances respecting the morning scenario are shown, the optimal solution OPT and the ratio between them. Then the results for instances respecting the evening scenario are shown: the total tour length TTL found by MAIN and, the optimal solution OPT and the ratio between them. Finally, the values of the total tour length TTL found by MAIN with general instances are shown in comparison with the optimal solution OPT and the ratio between them. The competitive ratio c is shown for each of the scenarios, it is always greater than $\frac{\text{TTL}}{\text{OPT}}$ unless $c = \frac{\text{TTL}}{\text{OPT}} = 1$. In these test instances the length of the line used is equal to $|L| = 32$

5. Concluding remarks

Vehicle routing problems integrating constraints on autonomy are new in the field of operational research but are important for the future mobility. Autonomous vehicles, which are intended to be used as a fleet in order to provide a transport service, need to be effective also considering to their management. We summarize the results of competitive analysis presented in this paper in Table 3. Competitive analysis has been one of the main tools for deriving worst-case bounds on the performance of algorithms but an online algorithm having the best competitive ratio in theory may reach the worst case more frequently in practice with a certain topology. That is the reason why we are not only interested in the "worst-case" but also in the "best-case" performance of the algorithms, thus we need to determine properties which govern the behavior of each chosen algorithm and define the cases where it can be applied and give the best results in terms of performance. So far, we can suggest the following:

- Morning/evening: partition the network into disjoint circuits as subnetworks such that each subnetwork contains one parking p, assign one or several VIPA to every circuit operating in tram mode using SIF_M resp. SIF_E.
- Lunch time: consider a collection of circuits all meeting in a central station (the restaurant), one or several VIPAs on each circuit operating in tram mode using SIF_L.
- in general: consider a spanning collection of lines and circuits meeting in a central station where one VIPA (in elevator mode) operates on each line using MAIN, one or several VIPAs (in tram mode) on each circuit using SIR.

However, more research is required to give more precise suggestions for operating the fleet management in a globally good way.

In addition, minimizing the total tour length is not the only possible objective function to rate the quality of the studied online algorithms. In view of the fact that all requests for a time period $[t, t'] \subseteq [0, T]$ shall be served before time t', also minimizing the makespan (*i.e.* the time when the last request is served and the last VIPA returned to the depot) is an interesting objective. Moreover, in view of the quality-of-service aspect of the fleet management, also minimizing (total, maximal or average) waiting times of passengers is important. All these aspects have to be jointly taken into account in order to provide globally good solutions for the fleet management. However, this causes several difficulties: Consider one combination of a subnetwork G_1 and an online algorithm ALG_1 and another combination G_2 and ALG_2 for some time period $[t, t'] \subseteq [0, T]$. Even if ALG_1 operated on G_1 yields in general a better competitive ratio than ALG_2 operated on G_2 w.r.t. some objective function,

- there might be a request sequence σ (*e.g.* a best case for ALG_2, but a worst case for ALG_1) s.t. $\text{ALG}_2(\sigma) < \text{ALG}_1(\sigma)$ w.r.t. this objective function;
- ALG_2 may achieve a better competitive ratio than ALG_1 w.r.t. another objective function;
- an online algorithm in elevator mode can only operate one VIPA on a line, whereas an online algorithm in tram mode can operate several VIPAs on the same subnetwork and, thus, transport more passengers.

We illustrate these difficulties with the help of an example.

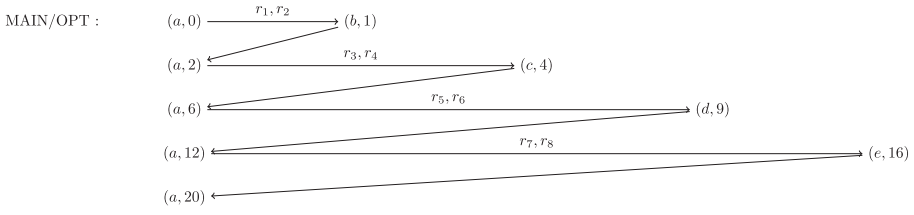
TABLE 3. This table shows the competitive ratios obtained in this paper. For each case we show the competitive ratio for the algorithm on which type of graphs, using which mode.

Mode (type of graph)	Algorithm	General	Lunch	Morning	Evening
Tram mode (Circuit)	SIR	$\text{Cap} \cdot \lvert C \rvert$	2Cap	Cap	Cap
Tram mode (Circuit)	SIF_M	–	–	1	–
Tram mode (Circuit)	SIF_E	–	–	–	1
Tram mode (Circuit)	SIF_L	–	2	–	–
Elevator mode (Line)	MAIN	$2\text{Cap} \cdot \lvert L \rvert$	2Cap	Cap	Cap

Example 5.1. Consider a subnetwork $G = (a, b, c, d, e)$ with origin a, all edge weights equal to 1, VIPAs of capacity 2and the following request sequence σ for the morning scenario:

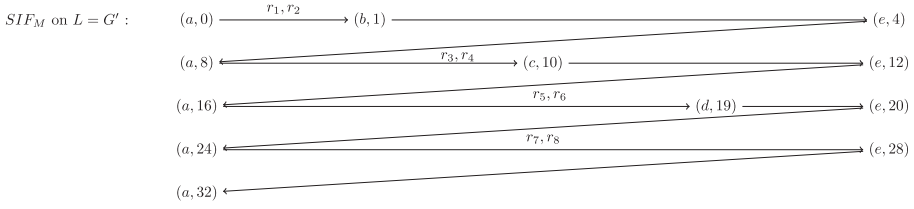
$$r_1 = r_2 = (0, a, b, 1)$$
$$r_3 = r_4 = (2, a, c, 1)$$
$$r_5 = r_6 = (6, a, d, 1)$$
$$r_7 = r_8 = (12, a, e, 1)$$

MAIN can operate one VIPA on bidirected lines and is Cap-competitive w.r.t. minimizing the total tour length. However, σ is a best case sequence for MAIN s.t. the tour constructed by MAIN for σ equals the optimal offline solution for the elevator mode on a line:

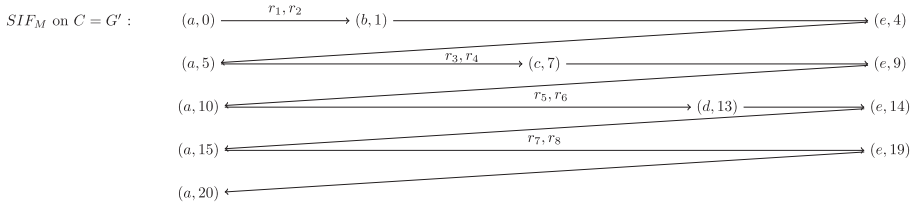


We have $\text{MAIN}(\sigma) = 20$ w.r.t. minimizing the total tour length and makespan, no waiting time occurred.

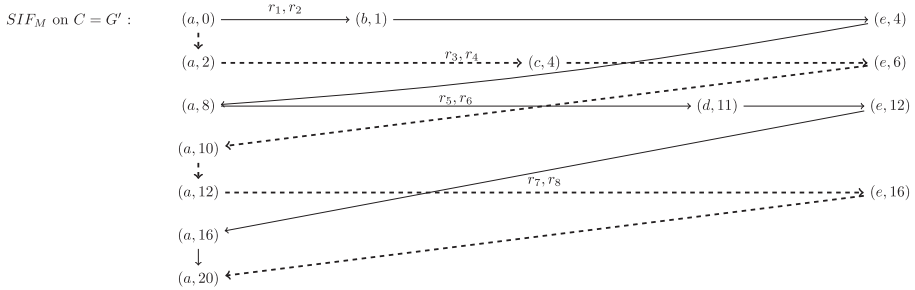
SIF_M can operate one or more VIPAs on a bidirected line $L = G'$ (*i.e.* with the change of the driving direction in a and e only) or on a unidirected circuit $C = G'$ (with an arc coming back from e to a). SIF_M produces the optimal solution w.r.t. minimizing the total tour length against an adversary on the same subnetwork. However, σ is a bad sequence for a bidirected line (as the VIPA cannot return to a after serving the request but has to traverse the whole distance): one VIPA operated by SIF_M yields:



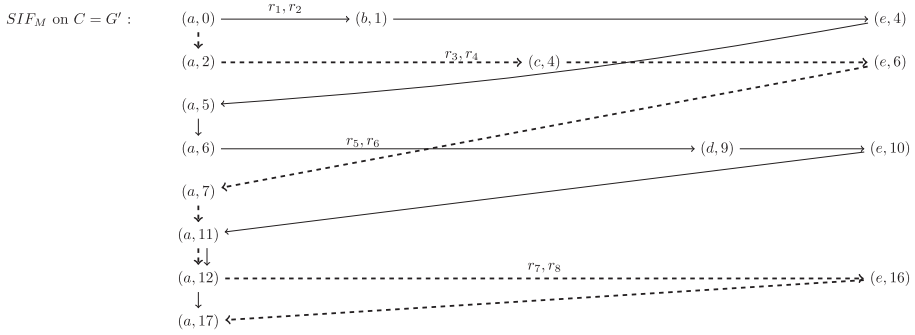
We have $\mathrm{SIF_M}(\sigma) = 32$ w.r.t. total tour length and makespan; a total waiting time of 56, a maximal waiting time of 12 and an average waiting time of 7. In contrary, if $\mathrm{SIF_M}$ operates one VIPA on a circuit, we obtain:



We have $\mathrm{SIF_M}(\sigma) = 20$ w.r.t. total tour length and makespan; a total waiting time of 20, a maximal waiting time of 4 and an average waiting time of 2.5. Hence, by operating one VIPA on a circuit, $\mathrm{SIF_M}$ produces on the best case sequence σ for MAIN the same objective function values as MAIN w.r.t. total tour length and makespan, but has longer waiting times. However, $\mathrm{SIF_M}$ can operate more than one VIPA on the same subnetwork. Using two VIPAs on a bidirected line, $\mathrm{SIF_M}$ constructs the following tours:



Again, $\mathrm{SIF_M}(\sigma) = 32$ w.r.t. total tour length (recall that this value is independent from the number of VIPAs used), but the makespan is only 20, and total waiting time $\mathrm{TWT} = 4$, Maximum waiting time $\mathrm{MWT} = 2$ and average waiting time $\mathrm{AWT} = 0.5$. Finally, using two VIPAs on a circuit, $\mathrm{SIF_M}$ constructs the following tours:



Hence, the total tour length remains 20, but the makespan reduces to 17, and there are no waiting times for passengers in this setting.

Hence, for the studied sequence σ the best choice is to let $\mathrm{SIF_M}$ operate 2 VIPAs on a circuit.

The future works are to analyze the proposed scenarios and algorithms further, *e.g.*, by

- providing competitive ratios w.r.t. the objective of minimizing the makespan,
- studying also the quality of service aspect by minimizing the waiting time for customers,
- providing solution approaches for the taxi mode,
- discussing the pro's and con's w.r.t. all objective functions, different combinations of modes, algorithms and subnetworks towards a globally optimal fleet management.

Appendix A.

We here provide details on the computational results.

Computational results

Detailed computational results are summarized in Tables A.1 and A.2. In Tables A.1 and A.2, a hyphen '-' indicates that no solution has been found (due to the restricted time horizon). In case the solution is preceded by a percentage, it means that only this percentage of instances give a solution within the time horizon.

Table A.1. This table shows the computational results for several test instances of the algorithms SIR and SIF_L w.r.t. the total tour length TTL and Makespan MS as objective functions.

		Morning				Evening				Lunch				MAIN (General)	
		SIR		SIF_M		SIR		SIF_E		SIR		SIF_L		SIR	
n	Cap	TTL	MS	TTL	MS	TTL	MS	TTL	MS	TTL	MS	TTL	MS	TTL	MS
5	1	125	125	125	105	125	125	125	125	112,5	112,5	100	112,5	143,75	95.7
5	5	86,67	119	25	105	100	90,2	25	95,6	75	74,5	25	75,5	131,25	91.6
5	10	86,67	119	25	105	81,25	90,2	25	95,6	68,75	74,5	25	75,5	87,5	91.6
20	1	500	–	500	–	500	–	500	–	418,75	–	400	–	593,75	123.5
20	5	125	115,5	100	95,4	113,3	92,6	100	87,6	206,25	106,7	81,25	104,5	287,5	103.4
20	10	106,25	106,25	50	95,4	110,4	92,6	50	87,6	206,25	106,7	56,25	104,5	268,75	92.6
200	1	5000	–	5000	–	5000	–	5000	–	4300	–	2968,75	–	5400	–
200	5	1270,8	–	1000	–	2881,25	–	1000	–	2125	–	600	–	2600	(29%)124.8
200	10	585,5	124,1	500	118,6	575,45	118,26	401,5	102,46	1512,5	120,5	468,75	113,5	2168,75	121.6

Notes. In this table, the instances are grouped by the number of requests (1st column) and the capacity (2nd column). Furthermore the values of the total tour length TTL and Makespan MS found by SIR with morning, evening, lunch and general instances are shown. For the morning (respectively evening and lunch) instances, the values of the total tour length TTL and Makespan MS found by SIF_M (respectively SIF_E and SIF_L) are also shown. In these test instances, 5 VIPAs are operating on a circuit that have a length equal to $|C| = 25$ and the time horizon is $[0, 160]$.

Table A.2. This table shows the computational results for several test instances of the algorithm MAIN w.r.t. the objective functions total tour length TTL and Makespan MS.

		MAIN (Morning)		MAIN (Evening)		MAIN (Lunch)		MAIN (General)	
n	Cap	$MAIN^{TTL}$	$MAIN^{MS}$	$MAIN^{TTL}$	$MAIN^{MS}$	$MAIN^{TTL}$	$MAIN^{MS}$	$MAIN^{TTL}$	$MAIN^{MS}$
5	1	97,3	110,8	85,5	117,67	61,15	89,5	56,3	72,4
5	5	55,5	84,5	55,5	68,6	74,76	84,5	42,45	50,78
5	10	55,5	84,5	55,5	68,6	74,76	84,5	42,45	50,78
20	1	327,2	–	365,2	–	222,35	–	140,8	185,5
20	5	135,8	135,8	135,8	145,6	78,4	90,8	95,54	120,3
20	10	78,3	92,35	108,3	124,6	75,58	84,6	80,15	100,4
200	1	3394,8	–	3434,8	–	4918,75	–	5023,25	–
200	5	1050,5	–	1050,5	–	2696,42	–	1662,4	–
200	10	760,8	–	860,8	–	1166,25	–	180,8	(15%)154,5

Notes. In this table, the instances are grouped by number of requests (1st column) and the capacity (2nd column). First the values of the total tour length TTL and Makespan MS found by MAIN for the instances respecting the morning scenario are shown. Then the results for instances respecting the evening scenario, and then the lunch scenario. Finally, the values of TTL and MS found by MAIN with general instances are shown. In these test instances, the length of the line used is equal to $|L| = 32$ and the time horizon is $[0, 160]$.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network flows: theory, algorithms, and applications. In: Prentice-Hall, Englewood Cliffs, New Jersey, USA Arrow, KJ (1963). Social Choice and Individual Values, Wiley, New York. Gibbard, A.(1973). Manipulation of Voting Schemes: A general result, Econometrica. **41** (1993) 587–602.

[2] N. Ascheuer, S.O. Krumke and J. Rambau, Online dial-a-ride problems: minimizing the completion time. In: *STACS 2000*. Springer (2000) 639–650.

[3] G. Berbeglia, J.-F. Cordeau and G. Laporte, Dynamic pickup and delivery problems. *Eur. J. Oper. Res.* **202** (2010) 8–15.

[4] M. Blom, S.O. Krumke, W.E. de Paepe and L. Stougie, The online TSP against fair adversaries. *INFORMS J. Comput.* **13** (2001) 138–148.

[5] A. Borodin and R. El-Yaniv, Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (2005).

[6] J.-F. Cordeau and G. Laporte, The dial-a-ride problem: models and algorithms. *Ann. Oper. Res.* **153** (2007) 29–46.

[7] S. Deleplanque and A. Quilliot, Transfers in the on-demand transportation: the DARPT Dial-a-Ride Problem with transfers allowed. *Multi. Int. Scheduling Conf. Theory Appl. (MISTA)* **2013** (2013) 185–205.

[8] Easymile, available at: http://www.easymile.com (2015).

[9] A. Fabri and P. Recht, Online dial-a-ride problem with time windows: an exact algorithm using status vectors. In: *Operations Research Proceedings 2006*. Springer (2007) 445–450.

[10] N. Labadi, C. Prins and M. Reghioui, A memetic algorithm for the vehicle routing problem with time windows. *RAIRO: OR* **42** (2008) 415–431.

[11] Ligier group, available at: http://www.ligier.fr (2015).

[12] S. Olariu, An optimal greedy heuristic to color interval graphs. *Info. Process. Lett.* **37** (1991) 21–25.

[13] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier and F. Marmoiton, Outdoor autonomous navigation using monocular vision. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005* (*IROS 2005*). IEEE (2005) 1253–1258.

[14] 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. *RFIA2016* (2016).

[15] E.L. Solano-Charris, C. Prins and A. Cynthia Santos, Solving the bi-objective robust vehicle routing problem with uncertain costs and demands. *RAIRO: OR* **50** (2016) 689–714.

[16] É.D. Taillard, A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO: OR* **33** (1999) 1–14.

[17] Viaméca, available at: http://www.viameca.fr (2015).

[18] Z. Yan, *Simulator for vipafleet management*, Master thesis, ISIMA Institut Supérieur d'Informatique de Modélisation et de leurs Applications, Clermont Ferrand, France (2016).