Models and algorithms for fleet management of autonomous vehicles

Sahar Bsaybes

Models and algorithms for fleet management of autonomous vehicles

Modèles et algorithmes de gestion de flottes de véhicules autonomes

by

Sahar Bsaybes

A Thesis

submitted to the Graduate School of Engineering Sciences

of the

Université Clermont Auvergne

in fulfilment to the requirements for the degree of

Doctor of Philosophy
in Computer Science

Supervised by Alain Quillot and Annegret K. Wagler

at the LIMOS laboratory - UMR CNRS 6158

Defense date: 26 October 2017

Committee:

Reviewers:	Prof. Luis Miguel Torres	-	Ecole Politecnique de QUITO, Equateur
	Prof. Aziz Moukrim	-	Université de Technologie de Compiègne, France
Supervisors:	Prof. Alain Quillot	-	Université Clermont Auvergne, France
	Prof. Annegret K. Wagler	-	Université Clermont Auvergne, France
Chairman:	Christian TArtigues	-	Directeur de Recherche, CNRS-LAAS

I wish to dedicate this thesis to Almighty God and Mother Mary, for the countless blessings during the course of my studies and for being an ever-present pillar of strength throughout my life.
This work is also dedicated to my loving family, who have motivated me every step of the way . . .
to Hanane the voice of reality,
to Mary the voice of the heart,
to Nibal the voice of madness,
to Elie the voice of wittiness,
to the bandmasters my wonderful tender parents,
to the early birds Lynn, Ralph, Georges and Yvan!

"Searching for the optimal path to happiness is NP-hard!
Use the simplest heuristic:
Be happy whatever the path is! "

Acknowledgment

This work was founded by the Région Auvergne within the FUI projects. Without their generous financial support this thesis would never have been possible.

I would like to express my gratitude to my supervisor, Ms. Annegret K. WAGLER, whose expertise, understanding, motivation and encouragement, added considerably to my experience. I appreciate her vast knowledge and skills in many areas, and her assistance in writing. Without her efforts my thesis would have undoubtedly been more difficult.
I would like to thank prof. Alain QUILLIOT for the assistance and the advises he provided at all levels of the research project. My sincere thanks also goes to Jan-Thierry WEGENER and Benoit BERNAY for their discussions, help and support. We worked together on the first phases of the research project.

I would like to thank my fellow colleagues and friends at the university especially my dear friend Karima, Marina, Alexis and Maximes. Thanks to them the time spent at the laboratory was full of humor and great moments.

My friends at Clermont Ferrand made my time here a lot more fun. Thank you for the Choir LOUP ANGES for all the time we spent togehter singing and laughing.
Thanks to Guillaume, Matthias, Aurelien, Blandine, Béatrice, Adeline, Mira, Ghina, Fadi, Medou, Amina and all the gang, I discovered the beauty of the Auvergne. Thank you Remy, resp. Bertrand for adding a touch of madness resp. wisdom to my days.
I am lucky to have met my friend Nancy here, and I thank her for her friendship and unyielding support. I could not image to have finished this thesis without her.
I owe a debt of gratitude to all my dear friends in Lebanon for their support across the sea especially Youness, Zeina and Rashel.

Finally, a very special thanks goes out for my family for the support they provided me through my entire life. Thank you Mom, Dad, Hanane, Mary, Nibale and Elie for your constant love and support.

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). The goal is to develop and implement suitable algorithms for each mode in order to satisfy all the requests either under an economic point aspect or under a quality of service aspect, this by varying the studied objective functions.

We model the underlying online transportation system as a discrete event based system and propose a corresponding fleet management framework, to handle modes, demands and commands. We consider three modes of circulation, tram, elevator and taxi mode. We propose for each mode appropriate online algorithms and evaluate their performance, both in terms of competitive analysis and practical behavior by computational results. We treat in this work, the pickup and delivery problem related to the Tram mode and the Elevator mode the pickup and delivery problem with time windows related to the taxi mode by means of flows in time-expanded networks.

Keywords: fleet management; offline and online pickup and delivery problem; autonomous vehicles; online optimization; heuristic; network flows

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Transport systems are fundamental to modern societies, functioning as vital sections that are central to economic and social activities (Rodrigue et al., 2016 [111]). They enable and shape individuals' mobility through their daily activities, which makes up their social network geography and facilitates social and psychological needs deemed necessary for their social well-being, quality of life, independence and greater life participation (Delbosc, 2012 [51], Axhausen, 2008 [18]).

However, society faces a grand challenge as the current form for individual motorized transportation's issues reveal themselves: it is an unsustainable approach that contributes heavily to climate change, resulting in adverse environmental and health effects (Woodcock et al., 2007 [122]).

Despite technological advantages and its commitment to reduce all emissions by 20% below the 1990 level before 2020, the European union's transport sector is one of few sectors where emissions have increased over the last 20 years. For instance, the total CO2 emission from private cars continues to rise, making them the leading source of greenhouse gas emissions after power production (European Commission, 2016 [1]). Such problems has tarnished the traditional image of cars and made it synonymous with pollution, traffic jams, nuisance and high costs [Chevrier (2008)].

Several innovative mobility systems have emerged in response to the current system's issues.

The current trends in mobility management involve the use of flexible reactive systems, which meet mobility demands in a dynamic way by implementing vehicle sharing and by interacting with alternative transportation modes. Such systems strive in order to find their room between fully individual mobility and traditional collective transportation systems. They also aim at bridging the gap between goods and people mobility, and require the use of advanced technologies [100] such as Internet, web services, mobile communications, remote tracking and monitoring. Depending on the context, they work either as closed systems, whose access is restricted to users who accept rules related to mobility tracking, pricing and responsibility, or open systems, which work on the basis of a free access/free market principle. Among such systems Car-Sharing, Car-Pooling (e.g., AUTOLIB) and Ride Sharing systems (see [23, 34, 93]).

On-Demand Transportation (ODT) permits its users to define the spatio-temporal framework of their ride. Taxis are certainly the most popular type of ODT service in urban areas. I t is also used in rural areas where the demand for public transportation is less frequent and for disabled people for whom mobility is harder and more demanding. SuperShuffle is an example of a successful company that specializes in ODT services for airports. It started 30 years ago in California USA, has expanded since in 20 countries and has a 7 billion euros turnover. Another example is the research project Modulobus, financed by ANR (national agency of research) from 2008 to 2011, that studied a new ODT system optimizing the number of vehicles' detours and its response time to users demands. Many people associate the word transport to the use of an individual car and most of the time, this car has only one person on board. This mobility attitude inspired the Ride-Sharing system. The user may either borrow a vehicle exclusively for a period of time (Car-sharing systems) or he may share its capacity with other users in order to reach a common destination (car-pooling, uber). Such transport systems are economic, ecologic and convenient to flow management.

Road accidents cause 1.2 million deaths and 50 million injuries every year. However, none of the above mentioned transportation systems proposes a solution for this serious issue. Thereby the need for a new type of innovative solutions that address security and efficiency standards on the road. On another side, recent advances in artificial perception and remote control made appear new generations of autonomous (i.e., without any driver) individual or collective electrical vehicles, such as Cycab and VIPA (Individual Public Autonomous Vehicle) developed by Easymile and Ligier [79, 80]. This combination of the emergence of a new generation of autonomous vehicles and current trends about people/good mobility towards more shared use, flexibility and reactivity led, in the case of VIPA electric cars, to a large scale experimental project VIPAFLEET, whose main partners were LIGIER S.A (VIPA manufacturer), MICHELIN Manufacture, EXOTIC SYSTEM S.A, LIMOS CNRS and PASCAL INSTITUTE CNRS and which was carried on inside the MICHELIN/LADOUX industrial in Clermont-Ferrand (FRANCE). Other applications are currently considered for the future, involving hospitals and some pedestrian downtown areas.

This thesis was founded by the French National Research Agency, the European Commission (Feder funds) and the Reǵion Auvergne in the Framework of the LabEx IMobS3 (Innovative Mobility: Smart and Sustainable Solutions) where more than 300 researchers and engineers, and more than 150 PhD students and post-doctoral fellows work in 7 research laboratories on three main challenges:

(1) Intelligent vehicles and machines: the focus of this challenges is on the conception of ergonomic, safe and intelligent vehicles and machines (autonomous and partly autonomous driving, advanced driver assistance systems, agricultural robotics, ...).

(2) Services and systems for smart mobility: This challenge studies innovative systems for mobility and how they can be integrated into their economical and social environment. The study of the innovative systems also includes the development of

new management system supporting an optimized control of fleets of vehicles within these innovative systems for mobility.

(3) Energy production processes for mobility: this challenge focuses on the development of design and optimization of an innovative and efficient processes for the production of biofuel, the storage of biofuel, and the life cycle analysis linked to the production and use of the new forms of energy.

Figure 1.1: The VIPA by Ligier and its newer version Easy10 by Easymile

The project VIPAFLEET aims at contributing to sustainable mobility (2) 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 [81]. A VIPA is an autonomous vehicle that does not require a driver nor an infrastructure to operate, it is developed by Easymile and Ligier [79, 80] within Challenge (1) thanks to innovative computer vision guidance technologies [112, 113]. Figure 1.1 shows a picture of the VIPA developed by Ligier and its newer version Easy10 developed by Easymile. 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 circuits 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.
- **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) 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 Dial-a-Ride Problem with time windows [54, 58]. In our case, we are confronted with an online situation, where transportation requests are released over time [10, 25, 47].

The goal of this project is:

- To develop and implement suitable algorithms for each mode in order to satisfy all the requests and to reduce the waiting time of a customer.
- To develop and install a 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 customers (Dynamic Fleet Management).This is the innovative idea and challenging problem of the project.

The project consists of the following phases.

- P1. Develop the autonomous vehicle that is able to communicate with safe and reliable operation with the external environment
- P2. Develop an on-line fleet management system that can perform real-time optimization of the different movements of the vehicles
- P3. Develop a solution of communication and interactive associated borns in order to ensure the communication between vehicles and with the central server of Fleet Management.
- P4. Ensure the reliability of the VIPA
- P5. Perform the experimentation of VIPA in real conditions (CHU, Michelin)

Our work is mainly focused on the area of Discrete Optimization. It consists of developing suitable models and efficient methods for solving complex on-line transportation problems. Hereby, knowledge of the structure of the underlying networks is often required. This project is a collaboration between many partners (IMOBS3, Ligier, Exotic system, and Michelin) and the LIMOS (area of Models and algorithms for computer-aided decision). The LIMOS contributes in the phases 1,2 and 5 of the project. Phase 2 is the main objective of my PhD thesis, which is innovative from a theoretical point of view. Since this type of dynamic fleet management problems is novel and highly complex.

The sequel of this thesis is structured as follows. In Chapter 2, we present the State-of-the-Art on related transportation problems. In Chapter 3 we present all the technical details, the constraints, the features and the requirements of the VIPAFLEET management system, and we model it as a discrete event based system. Chapters 4-6 are devoted to the models and algorithms for each of the VIPA circulation modes. Finally, we end this thesis with some concluding remarks on our approaches and on the global fleet management system. We also give some future lines of research and open problems.

Chapter 2

State of Art

The VIPAfleet project consists of developing models and algorithms for managing the fleet of Individual Public Autonomous Vehicles (VIPA). In this system, 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). In this chapter, we first introduce the main classical transportation problems and give a brief survey about solution methods and applications. As in this project the requests are not known in advance but revealed over time, the classical transportation problems do not reflect such situations. Therefore, we introduce the online optimization and some online transportation problems as well as some results about competitiveness for online algorithms on specific metric spaces.

2.1 Classical Transportation Problems

The first works on transportation problems are associated with the famous problem of the Traveling Salesman Problem (TSP). These works go back one or two centuries. Their origin is not clear, it appears to be in the 18th century. The TSP can be formulated as follows. A traveling salesman has to visit a number of given cities, starting and ending at the same city such that each of the cities must be visited once. The problem has been formulated several times by Sir William Rowam Hamilton, a mathematician from Ireland, and Thomas Penyngton Kirkman, a British mathematician. Detailed discussion about the work of Hamilton & Kirkman can be seen in the book titled Graph Theory (Biggs et al. 1976) [27]. It is believed that the general form of the TSP has been first studied by Karl Menger in Vienna and Harvard in 1932 [103]. The traveling salesman problem is a well known NP-hard problem [106]. The proof of this complexity was given by reducing the decision problem of TSP to the search for a Hamiltonian cycle for which Richard M. Karp showed in 1972 that it was NP-complete [89]. Thus, several heuristics [74], meta-heuristics [5, 88] and approximation algorithms [8, 35, 72, 73] have been applied in order to solve the problem within a reasonable time. The most successful methods are branch and bound techniques [78, 105] which can solve large instances i.e.

the Concorde TSP code [6] follows a branch-and-cut scheme and is able to obtain an optimal TSP tour through 85,900 cities

The vehicle routing problem (VRP) generalizes the traveling salesman problem. It consists of determining the optimal set of routes to be performed by a fleet of vehicles to serve a given set of customers and it is one of the most important, and studied, combinatorial optimization problems. It was first studied by George Dantzig and John Ramser in 1959 [50] where they described a mathematical formulation and an algorithmic approach applied for gasoline deliveries to service stations. In 1964, Clarke and Wright [41] improved Dantzig and Ramser's approach by proposing an effective greedy approach called the savings algorithm. The Vehicle Routing Problem VRP and its variants have been studied intensively in the last three decades and have been the object of several literature surveys. In particular, we refer to Magnanti 1981 [101], Assad et al. 1983 [11], Laporte & Nobert 1987 [97], Laporte 1992 [96] and Irnich et al. 2014 [83]. Specific examples are bank deliveries, postal deliveries, industrial garbage collection and routing and scheduling of school-buses. Furthermore, several variations of the vehicle routing problem have been studied, e.g., vehicle routing problems with time windows [98], capacitated vehicle routing problems [125], using a heterogeneous or homogeneous fleet of vehicles. Exact approaches for the vehicle routing problem and their variations are usually based on integer linear programming [38, 42, 56, 61, 65]. Typical applied metaheuristics range from tabu search [66, 68, 98, 117], to simulated annealing [117] and genetic algorithms [19, 118].

A generalization of the vehicle routing problem is the so-called pickup-&-delivery problem where a fleet of servers shall transport goods or persons between given origin and destination locations. If persons have to be transported, we usually talk about a Dial-a-Ride Problem (DARP). The usual objective functions considered are minimizing the operational costs, i.e., the costs of transferring the goods or persons. If persons are transported instead of goods, it is of interest to minimize the waiting or travel time for the persons. Typical applications of these pickup-&-delivery problems are less-than-truckload transportation and urban courier operations. A dial-a-ride problem typically occurs for every taxi company, the transportation of elderly or handicapped people.

Most variants of pickup and delivery problems have been studied intensively in the last three decades and have been the object of several literature surveys. In particular, we refer to Parragh, Doerner and Hartl [107, 108] for a general survey of PDPs and to Berbeglia et al. [24]. Many variants are also studied [45, 53] including the Dial-a-Ride Problems with time windows [54, 58, 59, 82, 124], capacitated Dial-a-Ride problems [46], using a heterogeneous [119] or homogeneous fleets [86] of transport vehicles. Besides tabu search [36, 46], and genetic algorithms [26, 37], also insertion techniques [52, 53] are applied to the dial-a-ride problem. An insertion algorithm initializes all tours for the drivers by the empty tours, and then, in each iteration, it tries to find a good position within a tour to insert and serve the requested transfers. Hereby, capacity and time-window constraints are respected.

Like other vehicle routing problems, these transportation problems are considered static or classical according to the availability of their information and their input data.

In static problems, all problem parameters are deterministic and known a priori before vehicle routes are constructed. Dynamic problems are characterized by the fact that some of the information required to make decisions is gradually revealed over time and requires the solution to be updated. Dynamic problems may also be stochastic when some information about the uncertain parameters are known in the form of probability distributions.

Complexity A (Combinatorial) Optimization Problem has as input a finite (implicitly given) set N, an objective $f : N \to \mathbb{Q}$ and the goal is to find among all feasible solutions N, one that maximizes respectively minimizes f, i.e. $\max_{x \in N} f(x)$ respectively $\min_{x \in N} f(x)$. The decision problem associated with an optimization problem P has an instance (a solution) $I \in P, z \in \mathbb{Q}$ and the goal is to answer whether there exists an objective f such that $I \geq z$ respectively $I \leq z$ or not. For example, the traveling salesman problem is an optimization problem, while the corresponding decision problem asks if there is a Hamiltonian cycle with a cost less than k in a given graph G. An optimization problem is NP-hard if its associated decision problem is NP-complete. The traveling salesman problem is a well known NP-hard problem [106]. The proof of this complexity was given by reducing the decision problem of TSP to the search for a Hamiltonian cycle for which Richard M. Karp showed in 1972 that it was NP-complete [89].

Conjecture 1. The decision problem, is there an optimal path in life for happiness, is NP-complete.

Flows in time-expanded network to model transportation problems Network flows over time have been first studied by Ford and Fulkerson [62, 63] in the 1960s, who developed a reduction of flow over time problems to static flow problems using time-expanded networks. These problems usually have a large number of variables and constraints and arise in great variety of applications

Two important characteristics of real-world transportation problems are the facts that flow along arcs varies over time and that flow does not arrive instantaneously at its destination but only after a certain delay. As none of these two characteristics is captured by classical network flows, the more powerful model of flows over time has been shifted into the focus of current research. Various interesting applications and examples can, for instance, be found in the surveys of Aronson [7] and Powell, Jailet and Odoni [110].

In our study, the two characteristics of the VIPAFLEET management system are the facts that no two VIPAs can traverse the same arc at the same time and that the operator needs to communicate to the VIPA the exact time to stop at certain station to pick up or deliver customers and he needs to communicate to the customer the time at which he will be served (picked up and then delivered).

In his introduction to network flows over time Skutella, 2009 [115] notes that the use of a discretization that includes each possibly relevant time point can be challenging computationally in many problem settings. There is widespread use of discretizations of time and time-expanded networks in service network design models ([4, 49, 57, 87, 95].

Modeling such problems is essentially based on the Multicommodity flow. Skutella et al. [75] show that the multicommodity flow over time problem is NP-hard, even on series-parallel graphs. Recently, for a service network design problem arising in less-than-truckload consolidation, Boland et al. 2017 [30] introduce a new approach, in which the strength of a time-expanded integer linear programming (IP) formulation is employed, without the penalty of an enormous model, they designed a partially time-expanded network without loss of optimality. In [31], the authors extend this dynamic approach solving the Traveling Salesman Problem with Time Windows.

Total unimodularity A matrix A is called totally unimodular if each square submatrix of A has determinant $0, +1$ or -1. In particular, each entry of A is $0, +1$ or -1. The interest of totally unimodular matrices for optimization was discovered by the following theorem of Hoffman and Kruskal 1956 [77]: If A is totally unimodular and b and w are integer vectors vectors, then both sides of the LP-duality equation

$$\max\{wx|Ax \leq b\} = \min\{yb|y \geq 0,\ yA = w\}$$

have integer optimum solutions.

The constraints matrix of maximum flow and minimum cost flow problems is totally unimodular. Thus, such network flow problems with bounded integer capacities have an integral optimal value.

2.2 Online Optimization

In our VIPAFLEET management system, users send their requests from smartphones, web application or ad-hoc communication devices and wait for the vehicles to serve them. As related movements are performed inside a closed and restricted area (smaller than a few square kilometers), user demands must be handled in a very reactive way. This means that standard classical transportation problem models are not useful here and that we really need to focus on the online situation. We must design a real time decision rule robust enough in order to be integrated into a complex wireless communication architecture. Hereby, this is the reason why the classical transportation problems presented above do not reflect the situation of the studied VIPAFLEET management system. In the next sections, we introduce the online optimization and some online transportation problems.

What is the common problem between a person who goes skying for the first time in his life, and a manager of a fleet of vehicles in a closed site where passengers arrive over time to benefit from a certain transportation service? **Uncertainty**.

They both need to take a decision that minimizes a certain cost or maximizes a certain profit without knowing the complete information about the upcoming future events.

Online optimization, a branch in operations research, provides the mathematical framework for dealing appropriately with such practical situations. In contrast to the

classical optimization problems, online optimization problems are characterized by the fact that not all their input data are known in advance. A solution strategy for online optimization problems (online algorithm) has, therefore, to make its decisions before the complete information about the data is available.

The arrival of the input data over time is most commonly modeled by the "sequence model" and the "time-stamp model", which differ in the way how the information becomes available to the online algorithm.

In both cases, an online algorithm *ALG* is confronted with a finite *request sequence* $\sigma = \{r_1, r_2, \ldots r_n\}$. Like in a classical optimization algorithm, ALG has to serve the requests r_j according to its specific rules and the action taken by ALG to serve r_j incurs a certain cost (respectively profit) and the overall goal is to minimize the total service cost (respectively maximize the profit).

For an online optimization problem in the *sequence model*, the requests must be served in the order of their occurrence. More precisely, that means:

- When serving a request r_j, the online algorithm ALG does not have any knowledge of the requests r_i with $i > j$; the decision taken by ALG how to serve r_j is irrevocable.

- Only after r_j has been served, the next request r_{j+1} becomes known to ALG; in some cases, the appearance of the last request r_n is announced, in other cases not.

As for an online optimization problem in the *time-stamp model*, requests are not ordered within a sequence but become available over time at their *release times*. The release time $t_j \geq 0$ is a non negative real number and specifies the time at which request r_j becomes known. More precisely, that means:

- an online algorithm ALG is confronted with a finite request sequence $\sigma = \{r_1, r_2, \ldots\}$ which is given in order of non-decreasing release times, and ALG cannot serve r_j before its release time t_j.

- ALG has to determine its behavior at time t, based on the requests r_j released up to time t, but ALG is allowed to wait and to revoke decisions (as long as they have not yet been executed or communicated to customers as (fixed) appointments).

- The action taken by ALG to serve r_j incurs a certain cost; waiting incurs additional costs typically depending on the elapsed time. In some cases, the end of the considered time horizon is known or announced, in other cases not.

Remark. Serving each single request r_j is like in classical optimization, but it might be only a locally good solution, not leading to a global optimum taken over the whole request sequence. It is also possible to define online profit-maximization algorithms for both the sequence model and the time-stamp model. For such problems, serving a request yields a profit and the goal is to maximize the total profit obtained.

The algorithms used in online situations are typically heuristics because no exact solution is possible due to the lack of information and because the requests must be processed immediately or within a short time horizon. Two main types of online algorithms for the sequence or time-stamp model are:

The general algorithmic scheme for the sequence model is relatively easy. The idea is to serve each newly released request with a certain RULE serve (see Algorithm 1), whereas the online algorithms for the time-stamp model are usually more complex (see Algorithm 2).

Algorithm 1 Algorithmic Scheme for the Sequence Model

Input: a sequence of requests σ (given one by one)
Output: costs for serving all requests in σ

```
for request r_j ∈ σ do
    serve r_j according to a RULE_serve
    update costs
return total costs
```

Algorithm 2 Algorithmic Scheme for the Time-Stamp Model

Input: a sequence of requests σ (given at their release times)
Output: costs for serving all requests in σ

```
Initialize list σ_t with released but unserved requests until time t
while σ ≠ ∅ do
    select one or several request(s) according to RULE_sel
    serve selected request(s) according to a RULE_serve
    update σ_t and costs
return total costs
```

We next discuss the evaluation of the applied solution strategies. The traditional theoretical analysis of algorithms is concerned with an offline problem where the complete input is given (but worst for the strategy of the algorithm) and is focused either on the efficiency (for an exact algorithm) or the quality of the solution compared to the optimal solution (for a heuristic).

To solve online problems, heuristic strategies are typically applied and so we are interested in rating the quality of a heuristic solution. So what should be called a "good" online algorithm? First systematic investigations to rate online algorithms were started by [Sleator and Tarjan 1985] [116] who suggested to compare an online algorithm with an optimal offline algorithm on the same request sequences. This so-called *competitive analysis* (introduced by [Karlin, Manasse, Rudolph, and Sleator 1988] [114]) became standard to rate the quality of online algorithms.

Given a request sequence σ, denote

- by ALG(σ) the cost incurred by an online algorithm ALG when serving σ and

- by OPT(σ) the optimal offline cost (the optimal offline algorithm OPT knows the entire request sequence σ in advance and can serve it with minimum cost, but also has to respect release times).

Definition 2.1. An online algorithm ALG is called **c-competitive** if ALG produces for some given $c \geq 1$ and for any request sequence σ a (primal) feasible solution ALG(σ) with

$$ALG(\sigma) \leq c \cdot OPT(\sigma).$$

The *competitive ratio* of ALG is the infimum over all c such that ALG is c-competitive.

Remarks:

- The definition of c-competitiveness varies in the literature. Sometimes an online algorithm ALG is called c-competitive, if there exists a constant b s.t.

  $$ALG(\sigma) \leq c \cdot OPT(\sigma) + b$$

  holds for any request sequence σ (and ALG is called strictly c-competitive if $b = 0$).

- 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 σ which maximizes the ratio between the online cost $ALG(\sigma)$ and the optimal offline cost $OPT(\sigma)$ knowing the entire request sequence σ in advance.

- In general, the theoretical results on analyzing deterministic online algorithms are weak. For randomized online algorithms, the adversary does not know the random decisions of the online player and cannot that easily construct a worst case instance.

- In practice, non-competitive algorithms typically behave worse than competitive ones. Whether competitive deterministic or randomized online algorithms behave better in practice depends on the studied instances (even if the randomized online algorithms often achieve a better competitive ratio). Typically, a *simulation* of the algorithm's behavior on some realistic test instances decides which algorithm is used in practice.

In the following, we define the types of adversaries used within this thesis. An **oblivious adversary** knows the complete behavior of a (deterministic) online algorithm ALG and chooses a worst-case sequence for ALG as well as the profits for serving the requests. He is allowed to move servers towards yet unreleased requests, but must not serve any request before it is released, i.e., before its release time.

In this thesis, we "weaken" the adversary by limiting the set of algorithms with which the adversary can solve the offline problem.

The **non-abusive adversary** (see, e.g., [92]) is limited by the algorithms he can choose from. He knows the complete behavior of ALG and can choose a worst case sequence but he is only allowed to move the servers towards origins (or destinations) of already released requests.

Remark 2.2. Competitive results, (ratios) against an oblivious adversary are the strongest. As for non-competitive results against a weaker adversary (e.g., non-abusive adversary) are the strongest. ♦

2.3 Online Transportation Problems in Metric Task Systems

In this section, we introduce a general model for transport problems, in which many (online) transportation problems can be captured. The definitions and the examples are taken from [120]. In a Metric Task System, servers have to process a sequence of tasks. The servers can be in one of a finite number of states and the cost of processing a task depends on the state of the server. Formally, a *Metric Task System* is a pair $(M, \mathcal{T})$, where $M = (V, d)$ is a (finite) metric space and $\mathcal{T}$ is a set of tasks, typically requests to visit certain points $v \in V$ or to transport objects from their start position $v_s \in V$ to their destination $v_d \in V$.

An example of an infinite metric space is the m-dimensional Euclidean space (R^m, d_E), where d_E is the standard Euclidean distance. Another example is a finite metric space induced by a weighted graph $G = (V, E)$ with weight function $w : E \to R_+$. The set of points is the set of nodes V and the distance between two points $v, w \in V$ is the length of the shortest path from v to w.

Many real-world problems can be modeled as transport problems in Metric Task Systems, where requests to visit certain points or to transport objects or persons from a start position to a destination have to be served by one or several servers (with a certain capacity in the case of transport requests). In this section we present online versions of the following optimization problems in the area of transportation.

- **Online Traveling Salesman Problem:** one server (the salesman) has to visit all points in V;
- **Online k-Server Problem:** k servers are available to visit points in V (which includes the problem to partition the requests appropriately and to plan tours for all k servers);
- **Online Pickup-and-Delivery Problem:** k servers are available to transport objects or persons from a start point to a destination in V; (which includes the problem to partition the requests appropriately and to plan tours for all k servers, taking into consideration that a server typically has a certain capacity C.
- **Online Pickup-and-Delivery Problem with time windows:** k servers are available to transport objects or persons from a start point to a destination in V; (which includes the problem to partition the requests appropriately and to plan tours for all k servers, taking into consideration that a server typically has a certain capacity C, respecting a time window for each of the requests. A server must pick up the good or person not earlier then the pickup time and deliver it not later then the latest deliver time.

2.3.1 Online Traveling Salesman Problem

The "Online Traveling Salesman Problem" (see, e.g., [29, 84]) is given in the time-stamp model. An instance of the Online TSP consists of a metric space $M = (X, d)$ with a distinguished origin $x_0 \in X$ and a sequence $\sigma = \{r_1, \cdots, r_n\}$ of requests.

- Each request is a pair $r_i = (t_i, x_i)$ where t_i is the release date and $x_i \in X$ the point in the metric space requested to be visited.
- a server is located in the origin x_0 at time $t = 0$ and can move at unit speed.
- A feasible solution is a tour for the server visiting all request points (not earlier than their release dates) which starts and ends in the origin.
- The cost of a tour is the time when the server has served the last request and has returned back to the origin (i.e. the makespan).
- The online algorithm does neither have information about the time when the last request is released nor about the total number of requests.

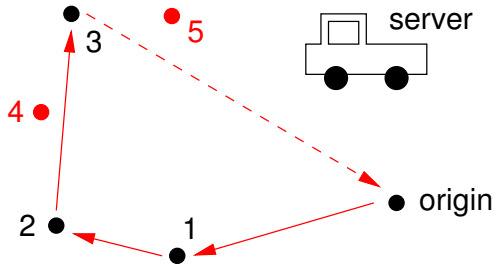


Figure 2.1: A partial solution of the O-TSP.

Figure 2.1 illustrates the setting for O-TSP. More formally, we obtain the following problem formulation:

Problem 2.3 (The Online Traveling Salesman Problem (O-TSP))**.**

Given: *a metric space $\mathcal{M} = (X, d)$ with an origin $x_o \in X$ and a sequence $\sigma = \{r_1, \ldots, r_n\}$ of requests, where each request is a pair $r_i = (t_i, x_i)$ specifying release time t_i and the point $x_i \in X$ to be visited.*

Task: *create a tour for the server, visiting all points x_i not earlier than t_i, which starts and ends in the origin.*

Goal: *Minimize the travel time of the server.*

Remark: The O-TSP differs from its famous relative, the classical TSP in certain aspects:

- The server might visit a certain point in the metric space more than once.

- The cost of a tour is not the length of the tour, but the *total travel time* needed by the server (obtained from the tour length plus waiting times).
- There are two versions of the online problem. In the first one, open online TSP, the server is not required to return to the departure point after all presented requests have been served. For the other version of the problem, closed online TSP, returning to the departure point is required. Both, returning immediately to the origin and waiting for new requests could cause unnecessary costs, depending on whether or not a further request becomes known.

An offline algorithm for this problem knows the complete sequence $\sigma = r_1, r_2, \dots r_n$ in advance and can produce an optimal tour. As the problem is also hard in the offline version, producing an optimal solution is not always real-time compatible, but there are good approximation algorithms (especially in the metric case).
In principle, every online algorithm for the closed O-TSP is described as in Algorithm 3.

Algorithm 3 Algorithmic Scheme for the O-TSP

```
1: Wait in the origin for the first released request and visit x_1.
2: if there are known unserved requests then
3: |   determine the next point(s) to be visited due to a certain RULE
4: |   visit the point(s)
5: else
6: |   return to the origin.
7: return makespan
```

Thus, the algorithms for O-TSP differ only in the RULE how to determine the next point(s) to be visited (based only on the sequence $r_1, \dots r_j$ with $t_j \leq t$). Examples for RULEs are:

- **FIFO:** select the longest released point next
  (i.e. visit all requested points x_i in the order of their release dates t_i)
- **GREEDY:** select the requested point x_j closest to the current server position
  (i.e. visit always the currently nearest point)
- **IGNORE:** determine an optimal tour for all currently known unserved requests and completely serve this tour
- **REPLAN:** determine an optimal tour for all currently known unserved requests and serve this tour until the next request becomes known

Note that FIFO behaves like in the sequence model. GREEDY and IGNORE make some use of the possibility to postpone decisions. Only REPLAN uses the possibility to revoke decisions about yet unserved requests. On the other hand, IGNORE can be seen to obey (already fixed) appointments with customers.

Example 2.4. Given the metric space $\mathcal{M} = (R_+^2, d)$ with d Manhattan metric, the origin $x_o = (0, 0)$, a unit speed server (1 unit way per 1 unit time) and the following request sequence σ:

i	1	2	3	4	5
t_i	0	2	3	6	10
x_i	$(3, 1)$	$(0, 2)$	$(3, 2)$	$(2, 1)$	$(1, 3)$

The considered strategies produce the solutions shown in Figure 2.2, including the optimal offline solution OFF(σ) and the optimal solution OPT(σ) of the underlying TSP without respecting release dates.

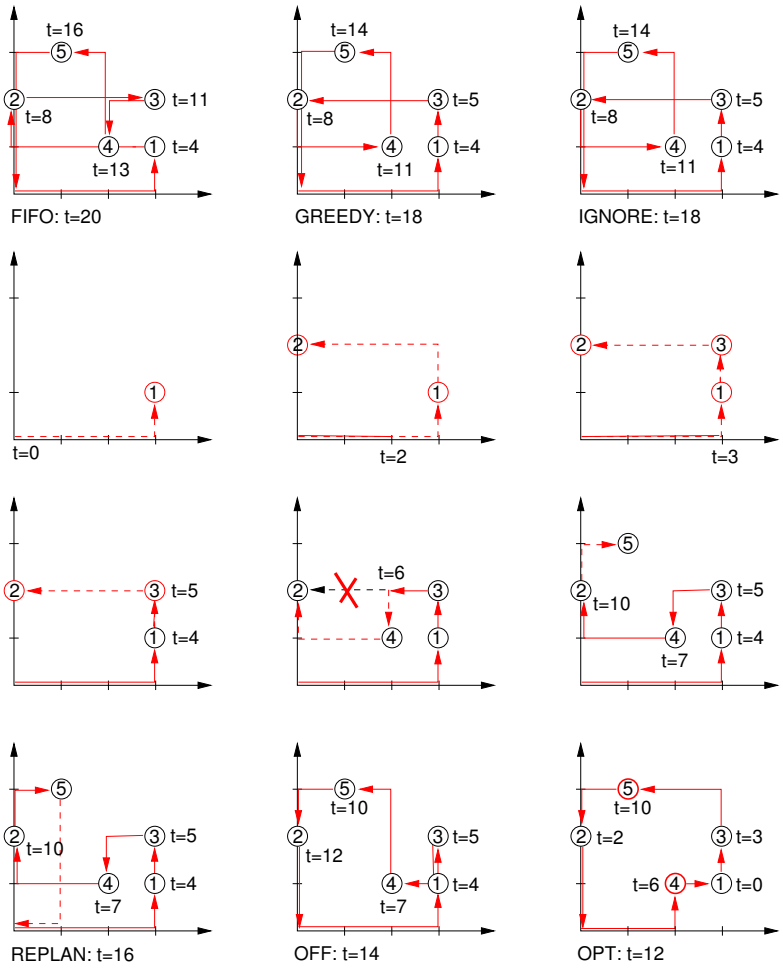


Figure 2.2: The solutions of the O-TSP from Example 2.4.

◊

Research concerning online versions of the TSP have been introduced relatively recently. Kalyanasundaram and Pruhs [28] have treated a unique version of an online

traveling salesman problem where new cities are revealed locally during the traversal of a tour (i.e., an arrival at a city reveals any adjacent cities that must also be visited). Angelelli, Savelsbergh, Speranza [3, 4] study related online routing problems in a multi-period setting. Ausiello, Feuerstein, Leonardi, Stougie, Talamo [15] studied the online TSP. They analyzed the problem on the real line and on general metric spaces, developing online algorithms for both cases and achieving a best-possible online algorithm for general metric spaces, with a competitive ratio of 2. These authors also provide a polynomial-time online algorithm, for general metric spaces, which is 3-competitive. The paper by Ascheuer, Krumke, Rambau [10] implies the existence of a polynomial-time algorithm, for general metric spaces, which is 2.65-competitive as well as a $(2+\epsilon)$-competitive ($\epsilon > 0$) algorithm for Euclidean spaces. Lipmann [99] developed a best-possible online algorithm for the real line, with a competitive ratio of approximately 1.64. Blom, Krumke, de Paepe, Stougie [15][29] gave a best-possible online algorithm for the non-negative real line, with a competitive ratio of $\frac{3}{2}$. Other groups of researchers have studied the online TSP in other variants: Ausiello, Bonifaci and Laura [7] [14] have studied the online Asymmetric TSP, Ausiello, Demange, Laura, Paschos [8] [15] have studied the online Quota TSP, and Allulli et al. [3], define the notion of a lookahead. A lookahead δ allows an online algorithm to foresee all requests that arrive during the next δ time units. The authors investigate the effect of the lookahead on many different online vehicle routing problems.

Ausiello et al.[17] consider in their paper two versions of the problem open online TSP, and closed online TSP. For the open online TSP they derive a lower bound on the competitive ratio of 2 on the real line. Besides, a 2.5-competitive algorithm for a wide class of metric spaces, and a $\frac{7}{3}$-competitive algorithm for the real line are provided. For the other version of the problem, closed online TSP, they present an optimal 2-competitive algorithm for the general class of metric spaces. If in this case the metric space is the real line they present a 1.75-competitive algorithm that is compared with a 1.64 lower bound.

Bjelde et al. 2017 [28] provided recently for closed online TSP, a 1.64-competitive algorithm, thus matching the known lower bound. For open online, they give a new upper bound as well as a matching lower bound that establish the remarkable competitive ratio of 2.04.

Lipmann [36] [99] considers the online TSP where all cities are on the real line. Lipmann designs a rather complicated online algorithm that is best-possible with a competitive ratio of $(9 + \sqrt{17})/8 \approx 1.64$

Blom et al. [29] consider the online TSP when all cities are on the non-negative real line. They also consider different types of adversaries. In other words, instead of comparing the cost of an online algorithm to that of an optimal offline algorithm, the online cost is compared to the cost of another weaker algorithm. In this way, the competitive ratio results are not as pessimistic and more realistic since many times the competitive ratio is induced by rather contrived problem instances. We conclude our summary of these authors' work with a presentation of their best-possible online algorithm Move-Right-If-Necessary (MRIN) for the online TSP on the real line, where

Table 2.1: Overview of results of competitive analysis of the Closed Online TSP on the line w.r.t. minimizing the makespan

References	Algorithm	Competitive ratio c	lower bound LB
Ausiello et al. 1994 [16]	Algorithm	$c = \frac{7}{4} = 1.75$	$LB = 1.64$
Ascheuer et al. 1998 [9]			$LB = 1 + \frac{\sqrt{2}}{2} = 1.70$
Lipmann 1999 [99]	(complicated)	$c = 1.64$	$LB = 1.64$
Blom et al. 2001 [29]	MRIN (non-negative real line)	$c = 3/2$	
Bjelde et al. 2017 [28]	(simple)	$c = (9 + \sqrt{17})/8 \approx 1.64$	

the competitive ratio is 3/2.

Algorithm 4 Algorithm : MRIN for the O-TSP on the non-negative real line

1: If there is an unserved city to the right of the salesman, he moves towards it at unit speed.
2: If there are no unserved cities to the right of the salesman, he moves back towards the origin at unit speed
3: Upon reaching the origin, the salesman becomes idle.

Regarding the performance of the Online TSP, we have

Theorem 2.5. *(Ausiello et al. 1994) [16].*
No (deterministic) online algorithm for Online TSP can achieve a competitive ratio $c < 2$.

The online algorithms IGNORE and REPLAN have been developed and analyzed w.r.t. minimizing the makespan by [Ascheuer, Krumke and Rambau 1998].

Theorem 2.6 (Ascheuer, Krumke and Rambau 1998 [9])**.**
The online algorithms IGNORE and REPLAN are $\frac{5}{2}$*-competitive for O-TSP w.r.t. minimizing the makespan on general metric spaces.*

On the other hand, we have the following lower bounds:

Theorem 2.7 (Ascheuer, Krumke and Rambau 1998 [9])**.**
No (deterministic) online algorithm for O-TSP can achieve a competitive ratio

- $k < 1 + \frac{\sqrt{2}}{2} = 1.70$ *for the metric space* $X = R$*,*
- $k < \frac{5}{3} = 1.66$ *in general.*

Table 2.2: Overview of results of competitive analysis of the Closed Online TSP on the general metric space w.r.t. minimizing the makespan

References	Algorithm	Competitive ratio c	lower bound LB
Ausiello et al. 1994 [16]			$LB = 2$
Ascheuer et al. 1998 [9]	IGNORE and REPLAN	$c = \frac{5}{2}$	$LB = \frac{5}{3} = 1.66$

2.3.2 The Online k-Server Problem

In a k-Server Problem, the objective is to find the best way to visit a given set of points exactly once using k servers (starting from and returning to a depot). The offline version of the k-Server Problem is already NP-hard because it contains the TSP as a special case (the TSP is a k-Server Problem with $k = 1$). In addition finding a tour for each server includes the problem to partition the set of given points into k appropriate subsets.

The Online k-Server Problem can be understood as a metric task system where an instance consists of

- a metric space (X, d) with a distinguished origin $x_0 \in X$ (the depot),
- a sequence $\{r_1, \cdots, r_n\}$ requests $r_i = (t_i, x_i)$ where t_i specifies the release date and $x_i \in X$ is the point requested to be visited.

A task is served by moving one server to the requested point $x_i \in X$ (not earlier than the time t_i).

A feasible solution consists of k tours for the k servers such that each requested point is visited exactly once; all k tours start and end in the origin x_0. The goal is to minimize the total distance traveled by the servers. Hereby the Online k-Server Problem may occur in different variants:

- **Time-stamp model:** there is "locally" an offline situation on the subset of already released but not yet served requests: the requests in the waiting list have to be assigned to servers and integrated in their tours.
- **Sequence model:** the requests become known one by one and for each of them, it has to be decided which of the k servers shall serve this request. (by moving from its current position to the requested one).

The "k-server problem" is a widely analyzed problem (see, e.g., [20–22, 39, 40, 43, 48, 55, 91, 102]).

This problem is a natural online version of the well-known transportation problem and has been extensively discussed in previous work (see for example [33, 76]). This section will present a brief summary of the main ideas and results.

Bonifaci and Stougie [32] study the online k-server problem. For the case where all cities are on the real line, these authors give an asymptotically (as $m \to \infty$) optimal

online algorithm. They also focus on resource augmentation with respect to the number of vehicles: the online algorithm has m salesmen and the offline algorithm has $m^* \leq m$ salesmen. They give an online algorithm that is $(1 + \sqrt{1 + 1/2^{\lceil m/m^* \rceil - 1}})$ -competitive. Ausiello, Allulli, Bonifaci, Laura [13] consider the behavior of online routing algorithms as a function of the number of servers.

2.3.3 The Online Pickup-&-Delivery Problem

The classical version of the Pickup-&-Delivery Problem PDP is NP-hard because it contains the TSP as a special case. In the Pickup-&-Delivery Problem, k servers have to transport loads from given origins or pickup node to given destinations or delivery node (starting from and returning to a depot). In fact the Pickup-&-Delivery Problem PDP generalizes the k-server problem, where the origin and the destination of a transportation request are equal.

The Online PDP can be understood as metric task system where an instance of the problem consists of

- a metric space (X, d) with a distinguished origin $x_0 \in X$ (the depot),
- a sequence $\{r_1, \cdots, r_n\}$ of transportation requests $r_j = (t_j, o_j, d_j, z_j)$ where t_j specifies the release date, $(o_j, d_j) \in X \times X$ the origin/destination pair, and z_j the load size.
- k servers s_i of capacity Cap and their initial position $pos(s_i) \in X$.

Serving a task r_j means to transport a load from the origin $o_i \in X$ of the request to its destination $d_j \in X$ by a server of capacity Cap (not earlier than t_j).

A feasible solution, called transportation schedule, consists of tours for the k servers such that each requested transport is performed, the server capacity is respected and all k tours start and end in the depot x_0. The goal is to minimize the total distance traveled by the k servers.

Hereby tours are composed by transportation moves.

Definition 2.8. A **transportation move** for a server s is a quadruple

$$m(s) = (t, x, y, R)$$

where t is the starting time, x the starting point, y the end point, and R the (possibly empty) set of requests loaded during the move. The load $\sum_{r_i \in \sigma} z_j$ of move $m(s)$ must not exceed the capacity Cap.

A **tour** for server s is a sequence $m_1(s), m_2(s), \cdots$ of transportation moves such that the

- first move starts in the origin x_0,
- end point of $m_i(s)$ is the starting point of $m_{i+1}(s)$,

- starting time of $m_i(s)$ respects the release dates of all loaded requests,
- arrival time of a move is the sum of its starting time t and $d(x, y)$,
- last move ends in the origin x_0.

A transportation schedule consists of tours for all k servers such that each requested transport is performed.

Preemptive or non preemptive In the preemptive Pickup-&-Delivery Problem or Dial-A-Ride problem, the server can drop off any request it is carrying at its current location at any time. In the non-preemptive Dial-A-Ride problem, the server may only drop off a request at its target location.

In [60], Feuerstein and Stougie consider the online Dial-a-Ride problem, where each city is replaced by an origin-destination pair. The authors consider both the uncapacitated case, giving a best-possible 2-competitive algorithm, and the capacitated case, giving a 2.5-competitive algorithm. They also show that this is best possible, as no algorithm can have competitive ratio better than 2 independent of the capacity of the server.

As the Online TSP is a special case of the Online PDP, we infer from Theorem 2.5:

Theorem 2.9. *No (deterministic) online algorithm for Online PDP can achieve a competitive ratio $c < 2$.*

Ascheuer et al. [10] give a 2-competitive online algorithm for the online Dial-a-Ride problem with multiple servers and capacity constraints.

A survey on some competitive ratios for the Online TSP and other variants and Online DARP is found in [85]. In [28], authors narrow the gaps for online Dial-A-Ride on the line by giving improved bounds.

Additionally, they provide a simple preemptive 2.41-competitive algorithm, which improves a (non-preemptive) 3.41-competitive algorithm by Krumke [94]. For the closed Dial-A-Ride variant, the lower bound of 1.64 by Ausiello et al. [17] was improved for one server with unit capacity without preemption to 1.71 by Ascheuer et al. [10]. They improve this bound further to 1.75 for any finite capacity $c \geq 1$. The best known algorithm for closed Dial-A-Ride on the line for finite capacity $c \geq 1$ is 2-competitive and was given by Ascheuer et al. [10].

For the closed online Dial-A-Ride problem without preemption, Feuerstein and Stougie [60] show a lower bound of 2 for the competitive ratio in general, and present an algorithm with a best-possible competitive ratio of 2 for the case that the server has infinite capacity. Ascheuer et al. [10] analyze different algorithms for the same setting and present a 2-competitive algorithm for any finite capacity $c \geq 1$.

Complexity For the non-preemptive offline Dial-A-Ride problem on the line, results have previously been obtained for the closed variant without release times. For capacity c = 1 Gilmore and Gomory [67] and Atallah and Kosaraju [12] gave polynomial time

Table 2.3: Overview of results of competitive analysis of the Online PDP on the line w.r.t. minimizing the makespan

References	Algorithm	Competitive ratio c
Ascheuer et al. 2000 [10]	multiple servers and capacity constraints	2
Krumke 2006 [94]	non-preemptive algorithm	3.41
Bjelde et al. 2017 [28]	preemptive algorithm	2.41

Table 2.4: Overview of results of competitive analysis of the Closed Online Dial-A-Ride on the general metric space w.r.t. minimizing the makespan

References	Algorithm	Competitive ratio c	lower bound LB
Ausiello et al. 1994 [16]	one server, unit capacity		$LB = 1.64$
Ascheuer et al. 2000 [10]	finite capacity $c \geq 1$	2	$LB = 1.71$
Feuerstein and Stougie [60]	infinite capacity	Alg 2-competitive	$LB = 2$
Krumkeet al. 2012 [93]	k servers (real line and trees)	$c = k$	
Bjelde et al. 2017 [28]	finite capacity $c \geq 1$		$LB = 1.75$

algorithms, and Guan [71] proved hardness for the case $c = 2$. In [28], authors show that both the open and closed variant of the problem are NP-hard for any capacity $c \geq 2$. Additionally, they show that the case with release times and any $c \geq 1$ is NP-hard. The complexity of offline Dial-A-Ride on the line with unbounded capacity remains open.

There are many offline variants of the Dial-A-Ride problem, differing in capacities, the underlying metric space, release times and deadlines, open versus closed tours, and in whether preemption is allowed (e.g., see [121]). The special case without release times and unit capacity is known as the stacker crane problem. Attalah and Kosaraju [12] present a polynomial algorithm for the closed, non-preemptive stacker crane problem on the real line. Frederickson and Guan [64] show that this problem is NP-complete on trees. Guan [71] shows that the Dial-A-Ride problem remains easy on the line with capacities larger than one if preemption is allowed, and that it remains hard on trees.

For the non-preemptive Dial-A-Ride problem on the line authors of [28] show that the open and closed variant with release times are NP-hard. Without release times, they prove they are NP-hard for capacity $c \geq 2$. their reductions are from the circular arc coloring problem, which is also used in a reduction for minimizing the sum of completion times of Dial-A-Ride on the line with capacity $c = 1$ [121].

Theorem 2.10. *Bjelde et al. 2017 [28]*
No algorithm for the non-preemptive closed Dial-A-Ride problem on the line with fixed capacity Cap ≥ 1 *has competitive ratio lower than* $c = 1.75$

2.4 The Online Pickup-&-Delivery Problem with Time Windows

The Online Pickup-&-Delivery Problem with Time Windows can be understood as metric task system where an instance of the problem consists of

- a metric space (V, d) with a distinguished origin $v_0 \in V$ (the depot),
- k servers s_i of capacity Cap and their initial position $pos(s_i) \in V$.
- a sequence $\{r_1, \cdots, r_n\}$ of transportation requests $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ where
  - $t_j \in [0, T]$ is the release time (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 or goods,

  and $t_j \leq p_j$, $p_j + d(x_j, y_j) \leq q_j$, as well as $z_j \leq$ Cap needs to be satisfied

Serving a task r_j means to transport a load from the origin $o_i \in X$ of the request by a server of capacity Cap (not earlier than p_j) to its destination $d_j \in X$ (not later than q_j. 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 server 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.

A feasible solution, called transportation schedule, consists of tours for the k servers such that each accepted request (in case there are rejected requests) is performed, the server capacity is respected and all k tours start and end in the depot x_0.

Hereby tours are composed by transportation moves.

Definition 2.11. A **transportation move** for a server s is a quadruple

$$m(s) = (t, x, y, R)$$

where t is the starting time, x the starting point, y the end point, and R the (possibly empty) set of requests loaded during the move. The load $\sum_{r_i \in \sigma} z_j$ of move $m(s)$ must not exceed the capacity Cap.

A **tour** for server s is a sequence $m_1(s), m_2(s), \cdots$ of transportation moves such that the

- first move starts in the origin x_0,
- end point of $m_i(s)$ is the starting point of $m_{i+1}(s)$,
- starting time of $m_i(s)$ respects the time windows of all loaded requests,
- arrival time of a move is the sum of its starting time t and $d(x, y)$,
- last move ends in the origin x_0.

A transportation schedule consists of tours for all k servers such that each requested transport is performed.

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 servers has exactly one tour that starts and ends in the depot,
- each (accepted) request r_j is served within time window $[p_j, q_j]$.

The goal is to construct transportation schedules S for the servers with the objective to minimize the total distance traveled by the k servers. Solution approaches for problems with dynamic requests must follow the online routing process where, at the beginning of the planning horizon, initial tours are constructed for the k vehicles based on the already released requests. These tours can be followed without any modifications, until a new customer request is released. In this case, there is always a chance that the new customer can be inserted into the existing planned tours without affecting the order of subsequent customers and with minimal delay. However, it is more likely that the insertion of new requests into the existing tour will require either partial or full rescheduling of the vehicle tour.

The common practice for generating a base routing plan is to use exact, metaheuristic or heuristic algorithms already developed for the corresponding static problem. These algorithms can be applied in a rolling time horizon basis to reoptimize the existing solution when there is a new released request. For example, the method tabu developed by [Cordeau and Laporte (2003)] for the static case is also used for the Online PDP, see the works of [Mitrović-Minić et al. (2004)], [Attanasio et al. (2004)], [Berbeglia et al. (2012)] et [Kergosiena et al. (2011)]. Note that this process does not take into account of a possible appointment with the user. These insertions are done w.r.t. minimizing the cost. [Attanasio et al. (2004)] integrate the implementation of [Cordeau and Laporte (2003)] in the dynamic problem with an infinite penality validating the different solutions obtained on each unit. [Berbeglia et al. (2012)] used also the method of [Cordeau and Laporte (2003)] for the dynamic case, but the Tabu search is accompanied by an exact constraint programming procedure. [Kergosiena et al. (2011)] exploit the Tabu research in a real context: the transport of patients. Reoptimization approaches have the drawback of repeatedly solving difficult optimization problems, which may require

excessive computational times. Note also that exact approaches can provide optimal solutions for the current state only (unless information is available over the entire planning horizon in advance). In this case, any solution at hand may be suboptimal once new data arrives. We refer to Psaraftis [123] for an example where insertion of a new customer into an existing optimal tour renders the tour suboptimal. The use of exact approaches often results in better overall solutions compared to using heuristics for the same purpose (see Yang, Jaillet, and Mahmassani [158] and Chen and Xu [26]). As mentioned above, heavy computational requirements might hinder the use of reoptimization procedures, especially in highly dynamic environments where the problem quickly becomes more complex with arrivals of new information (see Ichoua, Gendreau, and Potvin [77]). In this case, an alternative approach is to locally update the existing solution. For this purpose, a wide variety of local update and instant reaction heuristic methods (e.g., insertion heuristics) have been proposed. Ordinary insertion procedures and well-known construction heuristics have often been employed in the literature for various problem settings under several operational constraints. In this case, planned routes are constructed for all known requests. Besides flexibility, one other advantage of using insertion procedures to react to incoming immediate requests is that the planned routes can be also used for later decisions. Furthermore, insertion procedures are sufficiently fast and can also be used in real time to accept or reject a request or to specify a time window for a customer visit (see Ichoua, Gendreau, and Potvin [77]). [Madsen et al. (1995)] used the heuristic of resolution based on insertions of [Jaw et al. (1986)] and adapt it to the dynamic context. Their approach inserts the new request to the existing tour in less then one second. Their works had an application in the real word that was a transportation service for the elderly and or disabled people in Copenhagen. Users provide a single time window for the origin or for the destination. Different types of vehicles are used. They are not available at the same time and they may break down.

Whenever an immediate request is received (or at regular time intervals) the effort is initially to find feasible insertion positions, or to dispatch a new vehicle, for the new requests in the existing plan and handle the insertion of pairs of pickup and delivery locations. At this point, accept or reject requests may occur (see Gendreau et al. [48] and Ichoua, Gendreau, and Potvin [74] Berbeglia, Cordeau, and Laporte [13], Cordeau and Laporte [32], Cordeau et al. [34], and Madsen, Ravn, and Rygaard [98].

Subsequently, depending on the objective(s), the best feasible insertion position(s) is selected and the new requests are incorporated into the routing plan. Although various fitness criteria and metrics have been proposed (e.g., related to the geographical proximity, the temporal closeness, the latency, the response times), the most used methods adopt an insertion position that results in the shortest detour over a subset of vehicle routes. Yi and Tian (2005) maximize the number of requests for which service starts within a fixed time period after their release. They provide lower bounds for the single-vehicle case with either unit capacity or infinite capacity. Yi et al. (2006) add restricted information and a finite capacity to the work of Yi and Tian (2005).

Applications Some approaches for resolving the online DARP with time windows or online PDP with time windows are already used in real world transportation. For example, Hanne et al. (2009), Beaudry et al. (2010) study transportation systems in a hospital context, where emergency requests should be serviced within a very limited time frame. Coslovich et al. (2006) focus on unexpected users asking for service during the stop of a vehicle. Cremers et al. (2009) consider subcontracting requests to taxi services during peak moments. The taxis are cheaper when booked one day in advance, but some requests are only revealed at the beginning of the operation day. Apart from additional requests, several unexpected events related to users or vehicles may be taken into account, including user no-shows, cancelations of requests, changes of requests, vehicle breakdowns and traffic jams (Donoso et al. 2009; Häme 2011). Particularly the latter two may have a considerable operational impact (Xiang et al. 2008). We refer to [Beaudry et al. (2012)] for relevant details and to [Cordeau and Laporte (2007)] and [Pillac et al. (2013)] for surveys on online PDPs and online VRPs.

In the VIPAFLEET management system, we are dealing with autonomous vehicles. Therefore, we need to carefully model the problem to trace the route of the VIPAs over time and ensure that there is no VIPA blocking another and we need to obey as well other technical requirements detailed in Chapter 3. Therefore we will use some simple heuristics for solving the online PDP on specific metric spaces. In order to solve the online PDPTW, we will use flows in time-expanded networks.

Chapter 3

Modeling Framework for the VIPAFLEET Management System

We embed the VIPAFLEET management problem in the framework of a metric task system as proposed for the (online) transportation problems detailed in Chapter 2. 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 $[0, T]$. Figure 3.1 shows the industrial site "Ladoux" of Michelin at Clermont-Ferrand where a long-term experimentation has been performed for several months.

3.1 Discrete Event-based System

The studied VIPAFLEET management system can be modeled as a discrete event-based system where

- the system components are the set of **VIPAs** $VH = \{vh_1, \ldots, vh_k\}$ having Cap as Capacity;
- a **system state** $\boldsymbol{w}^t \in \mathbb{Z}^n$ specifies, for each VIPA vh, the load that is the number of customers w^t_{vh} in the VIPA at a time $t \leq T$ within a time horizon $[0, T]$;
- an **attribute** $\text{att}(vh, t)$ of VIPA vh at time t specifies the location of the VIPA at time t, at which station, or on which arc;
- states can be changed by vehicle events (breakdown, meeting of two vehicles ...), or customer requests (pick up and deliver users).

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

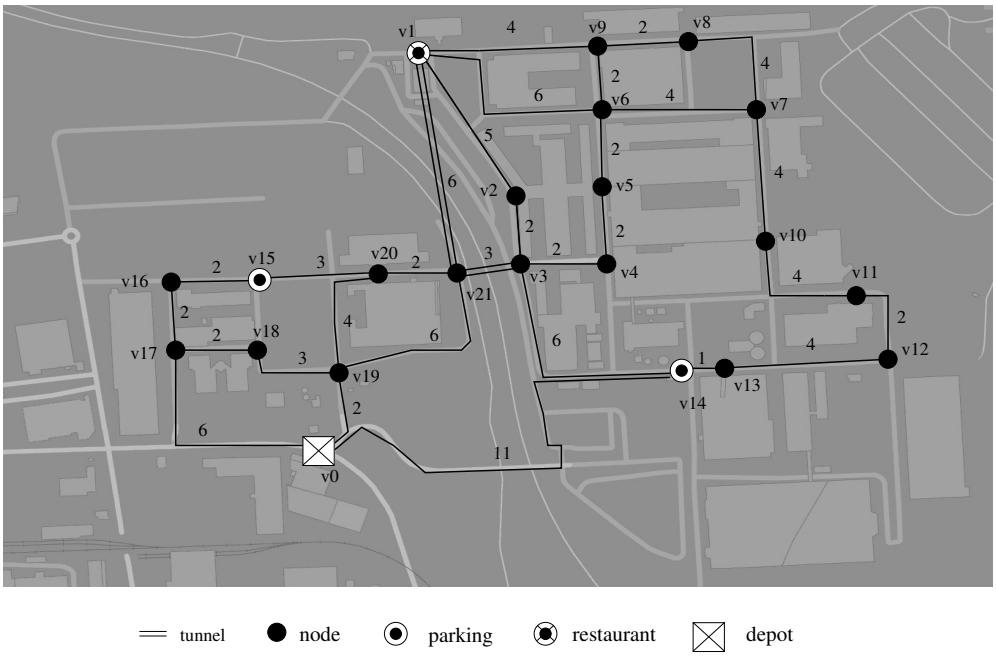


Figure 3.1: This figure illustrates the network G of the industrial site "Ladoux" of Michelin at Clermont-Ferrand. Each station is highlighted by a dot on the map; the different parkings are illustrated by a black dot within a white dot. The depot of the network is illustrated by a cross within a square and the restaurant by a cross within a circle. The edges illustrated by 2 parallel lines are tunnels (the road cannot be shown in the figure).

- $t_j \in [0, T]$ 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,

and $t_j \leq p_j$, $p_j + d(x_j, y_j) \leq q_j$, as well as $z_j \leq$ Cap needs to be satisfied.

Note that a request r_j may have missing information according to the mode wherein a VIPA is operating (in a tram, elevator, or a taxi mode) and to the source of the request. 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. Accordingly, different types of requests can be distinguished:

- **pdp-request:** a request coming from an evolved call-box specifying release date, origin, destination and load, $r_j^{pd} = (t_j, x_j, y_j, null, null, z_j)$, or simply $r_j^{pd} = (t_j, x_j, y_j, z_j)$, where t_j, x_j, y_j and z_j are known.

- **full-request:** a request coming from a web application $r_j^f = (t_j, x_j, y_j, p_j, q_j, z_j)$, where all the parameters t_j, x_j , y_j, p_j, q_j, and z_j are known.

In particular, the source of the request has an impact on the request type and therefore on the online transportation problem being considered.

- Having pdp-requests from call-boxes leads to an *Online Dial-a-Ride Problem* or *Online Pickup-&-Delivery Problem* (VIPAs have to transport users from an origin station to a destination).

- Having full-requests from a mobile or web application leads to an *Online Dial-a-Ride Problem with Time Window*, or *Online Pickup-&-Delivery Problem with Time Windows* (VIPAs have to transport users from an origin station to a destination within a certain time window).

An operator manages a fleet of k VIPAs each with a capacity of Cap passengers. The fleet management allows the operator to decide when and how to move the VIPAs in the network, and to assign requests to VIPAs. The operator monitors the evolution of the requests over time and

- decides which requests can be accepted (in case some requests have to be rejected), and
- creates tasks to serve accepted requests by moving the VIPAs to some stations to pickup, transport and deliver users.

More precisely, a **task** can be defined in different ways according to the type of requests that we are dealing with.

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

- **full-task** $\tau_j^f = (t_j, x_j, t_j^{pick}, y_j, t_j^{drop}, z_j)$**:** a task created by the operator in order to serve an accepted full-request $r_j^f = (t_j, x_j, y_j, p_j, q_j, z_j)$, this task 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. The system states $\boldsymbol{w}^t$ are influenced by customer requests. For that, we define, for every time t, an update vector $\mathbf{u}^t \in \mathbb{Z}^n$, where each index corresponds to

the number of customers leaving the vehicle vh ($u^t_{vh} < 0$) or entering to vh ($u^t_{vh} > 0$). Then, we can define for a system state $\boldsymbol{w}^t$ at time t the successor state $\boldsymbol{w}^{t+1}$ by

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \mathbf{u}^t.$$

A system state $\boldsymbol{w}^t$ is **feasible** if $0 \leq w^t_v \leq \text{Cap}$ and the equality $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \mathbf{u}^t$ holds for every VIPA $vh \in VH$ and **infeasible** otherwise.

An operator manages a fleet of k VIPAs each with a capacity for Cap passengers. The fleet management allows the operator to decide when and how to move the VIPAs in the network, and to assign tasks to VIPAs. In order to fulfill the tasks in $\mathcal{T}_A$, 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 (accepted) request r_j is served not earlier than the time t_j it is released (or not earlier than p_j if it is precised),
- each tour starts and ends in the depot.

Vehicle preemption Next, we explain the notion of **preemption** between tours, where users are not transported from their pickup station to their delivery station by the same VIPA, so they change VIPAs at a certain station v. Hereby, it is reasonable to assume that the users can only change VIPAs at stations and not on the street where changing vehicles can possibly cause a traffic jam, and is not desirable for a matter of security since we are dealing with autonomous vehicles. In other words, there is a preemption in v if a VIPA drops users at v and these users are picked up by another VIPA in order to arrive to their final destination. 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.

Load preemption Splitting transportation requests having a load $z_j > 1$, i.e. involving several passengers is called load preemption. Therefore, if load preemption is allowed, a request may be split if it is beneficial to do so. Then, a transportation request consisting of more than one user may be served by more than one VIPA or consecutively by the same VIPA.

Technical constraints 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 database or the communication network between the stations, VIPAs and the central server.

Depending on the policy of the operator in the studied VIPAFLEET management problem, different questions may arise:

- Do customers book their requests in advance?
- Can we reject customer requests, or do we have to serve them all?
  - If we have to serve them all: is there a feasible solution?
  - If we can reject requests: when does the algorithm have to make decision?
  - When do we communicate the decision to the customers to accept or reject the corresponding request?
- What is the objective function?
  - Maximize the number of accepted requests/the profit?
  - Minimize the costs/the tour length?
  - Minimize the maximal/total waiting time?
  - Is the riding time in the vehicle limited?
- Which strategy is best to plan tours?
- How do different algorithms perform?
- Do we allow load preemption?
- Which type of call-boxes do we use?
- Does a VIPA block the other or can one overtake the other in case of meeting of two VIPAs?
- Can two VIPAs use a same arc in both directions?

The answer to these questions depends on the circulation mode used for the VIPAs.

3.1.1 The Objective Functions

Depending on the policy of the operator, different objective functions may be considered. Therefore, the goal is to construct a transportation schedule $S = \{\Gamma^1, \ldots, \Gamma^k\}$ w.r.t minimizing one of the following objective functions.

- **Total tour length**: the length of a tour corresponds to the distance traveled by the VIPA(s). In this case, the total tour length of S is the sum of the lengths of the tours without considering the waiting time of the VIPA(s).
- **Makespan**: the time when the last VIPA returned to the depot v_0 after all tasks are served.
- **Total waiting time**: the sum of waiting times of the requests defined as the difference between the pickup time of a request, the time when the VIPA performed the pickup, and its release time.
- **Total number of stops**: the total number of stops where the VIPA stops at a station to pick up or deliver users. This objective function is interesting to study in case of autonomous vehicles as the time required by a VIPA to move between two stations clearly takes into account the decelerations and the accelerations required before and after the stop. This is a normal case while using any vehicle. But when using autonomous vehicles, this characteristic may slow enormously the VIPAS, as their speed is 15km/h and this will decrease if they have to stop at each station and restart.
- **Total number of rejected requests:** by first maximizing the number of accepted requests and second by serving the accepted ones with a minimum total tour length.

The global goal is to provide a feasible transportation schedule over the whole time horizon (from the morning till the evening) that satisfies all accepted requests and minimizes one of these objective functions (Global Fleet Management Problem).

3.1.2 Requirements of each Circulation Mode

Recall that a VIPA can operate in three different **circulation modes** in order to supply internal transportation:

- **Tram mode:** VIPAs continuously run on predefined lines or circuits 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.

Based on the circulation modes of the VIPA (tram, elevator, taxi) and the types of requests (pdp-requests and full requests), we may notice the following.

- Running the VIPAs in different circulation modes on the whole network at the same time is not possible, we may face security issues due to meeting of two VIPAs, and the whole network is not suitable for all circulation modes. The VIPA can run only on predefined lines or circuits when operating in tram mode, and only on predefined lines when operating in elevator mode.
- Precising a time window by a customer for his pick up and delivery station is of no interest if the VIPA is operating in tram mode or elevator mode.

Based on these technical precisions, we need to respect the following.

- Operating the VIPA in tram mode requires a subnetwork $G' = (V', E')$ of G, that is either a unidirected cycle, called *circuit* C or a bidirected path, called *line* L, and pdp-requests $(r_j = t_j, x_j, y_j, z_j)$. If more than one VIPA operates on C or L, the meeting of two VIPAs has to be addressed (avoided).
- Operating the VIPA in elevator mode, requires a subnetwork $G' = (V', E')$ of G, that is a bidirected path, called *line* L, and pdp-requests $(r_j = t_j, x_j, y_j, z_j)$. Only one VIPA can operate on a line.
- Operating the VIPA in taxi mode, requires a connected network $G = (V, E)$ with full-requests $(r_j = t_j, x_j, y_j, p_j, q_j, z_j)$.

Hereby, the VIPAFLEET management system integrates three main online problems: Tram Mode Problem, Elevator Mode Problem and Taxi Mode Problem. In our context, we are confronted with an online situation for the three problems, as the transportation requests are released over time. In the online situation, the customers interact with the VIPAFLEET system during the routing and scheduling process. Thus, the system state changes in general during the routing process in the online situation and the operator has to react dynamically to the changes.

3.2 Tram Mode Problem ($TramMP$)

The tram mode is the most restricted operation mode where VIPAs run on predefined circuits in a predefined direction at a constant speed equal to about 15 km/h and stop at stations to let users enter or leave. The behavior of the VIPAs is even independent of the source of requests (call-boxes, web or mobile application) and thus, the type of generated requests and tasks. We consider circuits C with one distinguished node, the origin of the circuit. Circuits can also be lines where one end is the origin and the VIPA can change its direction only at the two ends of the line. In this mode, we consider that a VIPA cannot overtake another. Running two or many VIPAs on the same circuit may cause problems, thus while operating several VIPAs on the same circuit, we make use of waiting strategies, which consider delaying the VIPA from performing its tour in an

attempt to prevent its meeting with another VIPA. For example, it might be beneficial for a VIPA to wait at its current location if it is empty and to be prepared to pick up new requests on its way in case new requests are released. Moreover, sometimes a VIPA may continue its tour without stopping at a station that is a pickup station of an already released but unserved request (there is a user waiting at this station to be picked up) to avoid blocking the next VIPA that will arrive shortly at this station.

Problem 3.1 (Tram Mode Problem $(M, \sigma, T, k, \text{Cap})$ $(TramMP)$)**.** *Given a metric space $M = (V, d)$ induced by a connected network $C = (V, E)$ (circuit), a sequence σ of pdp-requests $r_j = (t_j, x_j, y_j, z_j)$, a time horizon $[0, T]$ and k VIPAs of capacity Cap, determine a feasible transportation schedule $S = \{\Gamma^1, \ldots, \Gamma^k\}$ to serve all requests in σ with minimizing one of the following objective functions:*

- *total tour length,*
- *makespan,*
- *total waiting time,*
- *total number of stops.*

3.3 The Elevator Mode Problem (EMP)

The elevator mode is a less restricted operation mode where one VIPA runs on a predefined line (bi-directional elementary path) 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 (called, the "left" end). In this mode, we consider that a VIPA cannot overtake another one, running two or many VIPAs on the same circuit may cause problems. Thus, we concentrate on fully independent lines (they do not share any arc with each other).

Problem 3.2 (Elevator Mode Problem $(M, \sigma, p, T, 1, \text{Cap})$ (EMP))**.** *Given a metric space $M = (V, d)$ induced by a connected network $L = (V, E)$ (line), a sequence of pdp-requests σ, a time horizon $[0, T]$ and one VIPA of capacity Cap, determine a feasible transportation schedule $S = \{\Gamma^1\}$ to serve all requests in σ with respect to minimizing one of the following objective functions:*

- *total tour length,*
- *makespan,*
- *total waiting time,*
- *total number of stops.*

3.4 The Taxi Mode Problem ($TaxiMP$)

The taxi mode, which characterizes standard DARP or PDP systems and makes vehicles free to move from any origin to any destination along a network through a certain path is an elaborated mode. Actually, current technology does not make it realistic in the case of the VIPAs. Yet, this thesis involves the autonomous vehicles, the VIPAs, in the operational research issues. Modeling and studying this system makes it possible to create the schedules of these new vehicles. The future purpose would be to study and model such systems without worrying about the fact that they are autonomous. This work attempts to provide all the possible scenarios and analysis that may occur for the Taxi mode within a closed site to provide a solution that can be used today with technical constraints and that can be adaptable in the future when these constraints and issues are solved.

Problem 3.3 (Taxi Mode Problem $(M, \sigma, p, T, k, \text{Cap})$ $(TaxiMP)$)**.** *Given a metric space $M = (V, d)$ induced by a connected network $G = (V, E)$, a sequence of full-requests σ, profits p for accepted requests, a time horizon $[0, T]$ and k VIPAs of capacity Cap, determine a maximum subset σ_A of accepted requests (by maximizing the profit) and find a feasible transportation schedule $S = \{\Gamma^1, \ldots, \Gamma^k\}$ of minimum total tour length to serve all requests in σ_A.*

3.5 Design of the Network

In the following, we discuss how the customer behavior and their requests are represented within this model. This implies the representation of information about the behavior of the customers of the VIPAFLEET system (e.g., gained from statistics and preliminary studies) and their usage within a metric task system in order to solve the online transportation problems behind.

Defining three different transportation problems ($TramMP$, EMP, $TaxiMP$) motivated by the circulation modes of the VIPA and the type of the requests solves the VIPAFLEET management problem if we assume that we chose one problem among the three and apply it over the whole time horizon $[0, T]$, or if we partition the time horizon into different periods $[t, t'] \subseteq [0, T]$, and apply to it one of the three defined problems. However, running all the VIPAs in one circulation mode on the whole network G over the whole time horizon $[0, T]$ may not always be the best solution, i.e., If we consider one Hamilton circuit where VIPAs operate in tram mode, the passengers may wait for long time to be picked up or delivered. Considering a line that covers all the nodes of the network where one VIPA operates in elevator mode is not enough, it will restrict the quality of the solutions and limit the number of requests that can be accepted. In case we use the taxi mode on the whole network G over the whole time horizon and solve the Taxi Mode Problem, no partition of the network is needed. However, as we are working with autonomous vehicles, the taxi mode is a very difficult mode to operate. The VIPAs should apprehend the whole graph, arc by arc and they should handle all the degrees of

road deviation and angles. Thereby, the taxi mode should be used in necessary cases, when other modes fail to satisfy the requests.

Therefore we aim at proposing a suitable design for the network. This design might be a collection of subnetworks (circuits and lines) such that all stations of the network are covered, and the chosen subnetworks intersect (to ensure transports between all possible origin/destination pairs), such collection is called a partition. Partitioning of the network into subnetworks is mandatory for the reliability of the system, but the question is how to partition the network? For that we especially rely on the customer behavior. In the following we detail some preliminary studies of the transport requests within the industrial site "Ladoux" of Michelin at Clermont-Ferrand where a long-term experimentation [112] has been performed for several months (October 2014 - February 2015) that enabled us to suggest several partitioning solutions depending on the different periods of the time horizon.

3.5.1 Patterns of Requests

Based on some preliminary studies, we notice some particularities that characterize the requests in the VIPAFLEET management system on different periods of the day. We notice the following changing patterns of requests over time.

Morning respectively evening pattern of requests The transport requests are between parkings and buildings. Thus the requests have the same pickup stations, the different parkings of the site, during the morning and respectively the same delivery stations during the evening. Figure 3.2 shows the morning respectively evening pattern of requests in the industrial site "Ladoux" of Michelin at Clermont-Ferrand.

Lunch pattern of requests: The transport requests are between buildings and the restaurant of the industrial complex. Thus, the requests have either the same pickup station or the same delivery station (the restaurant). Figure 3.3 shows an example of the lunch pattern of requests in the industrial site "Ladoux" of Michelin at Clermont-Ferrand.

General pattern of requests: There are mainly unspecified requests without common origins or common destinations.

3.5.2 Scenarios: Combinations of Modes and Subnetworks

A Global Fleet Management System allows the operator to switch between different circulation modes within the different periods of the day in order to react to changing patterns of customer demands evolving during the day. When the system is prepared before the beginning of the service within a certain period, it is essential to use the information of the preliminary studies about the customer behavior to predict a good design of the network for the next period. For that, for a certain period $[t, t'] \subseteq [0, T]$, we define a metric subspace $M' = (V', d')$ induced by a set of subnetworks of G, where

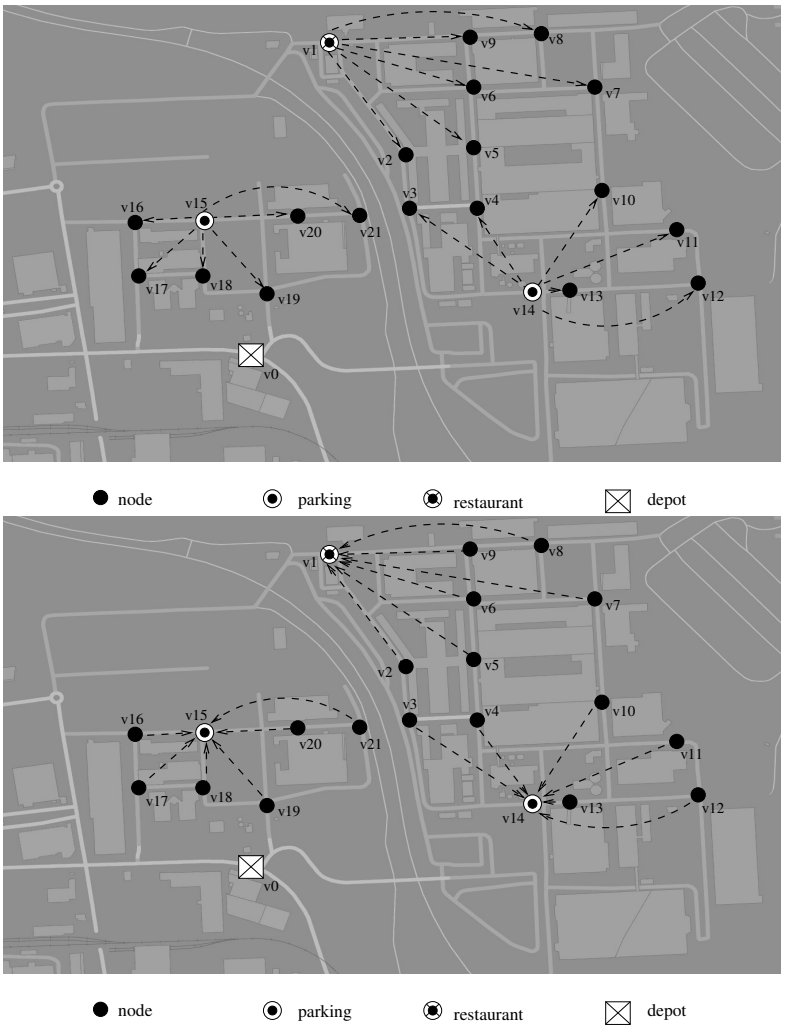


Figure 3.2: This figure illustrates an example of the morning respectively evening pattern of requests on the network G of the industrial site "Ladoux" of Michelin at Clermont-Ferrand. Each station is highlighted by a dot on the map; the different parkings are illustrated by a black dot within a white dot. The depot of the network is illustrated by a cross within a square. The transportation requests of the customers are indicated by arcs from the origin to the destination.

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 $[t, t']$).

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 operate, will change over time according to the technical features, properties

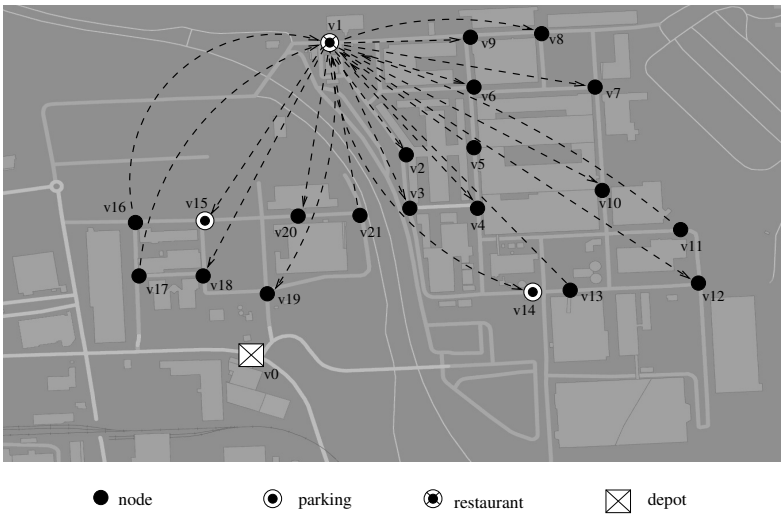


Figure 3.3: This figure illustrates an example of the lunch pattern of requests of the network G of the industrial site "Ladoux" of Michelin at Clermont-Ferrand. The requests are between the buildings (dot) and the restaurant (a cross within a circle).

and the request patterns. We consider four typical scenarios (periods $[t, t'] \subseteq [0, T]$) that occurred while operating a fleet in an industrial site 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 3.4 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 3.5 s.t.

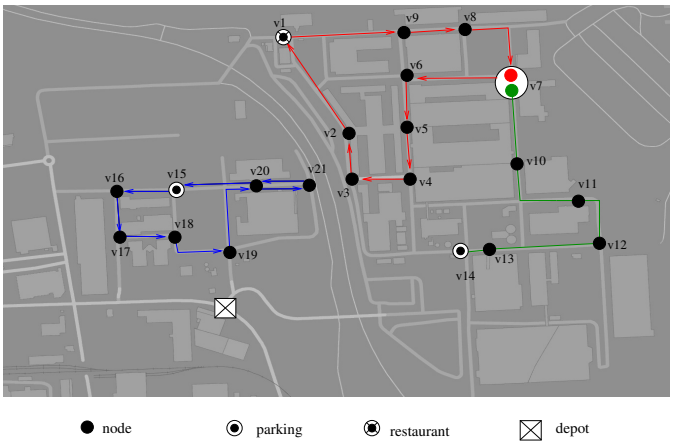


Figure 3.4: This figure illustrates an example of a collection of subnetworks (one line and two circuits) for the morning/evening scenario, each subnetwork is shown in a different color. All buildings and parkings are covered in this partition. Each subnetwork contains one parking (black dot within a white dot). The station v_7 is a common station for two subnetworks.

  - 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 as shown in Figure 3.6, 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). For example, this can be done by

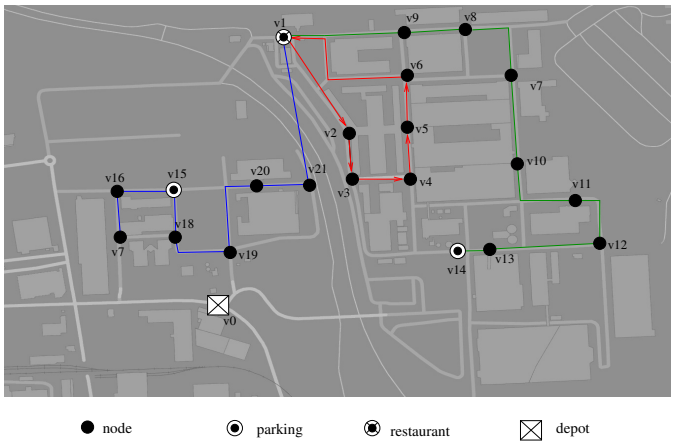


Figure 3.5: This figure illustrates an example of a collection of subnetworks (two lines and one circuit) for the lunch scenario, each subnetwork is shown in a different color. All stations (buildings and parkings) are covered in this partition, and each subnetwork contains the station of the restaurant (a cross within a circle).

- using one Hamilton cycle through all stations where half of the fleet operates (in tram mode) in each direction, (leading to tour non-preemptive schedules)
- a spanning collection of lines and circuits meeting in a central station, as shown in Figure 3.3, where one VIPA (in elevator mode) operates on each line, one or several VIPAs (in tram mode) on each circuit (leading to tour preemptive schedules).

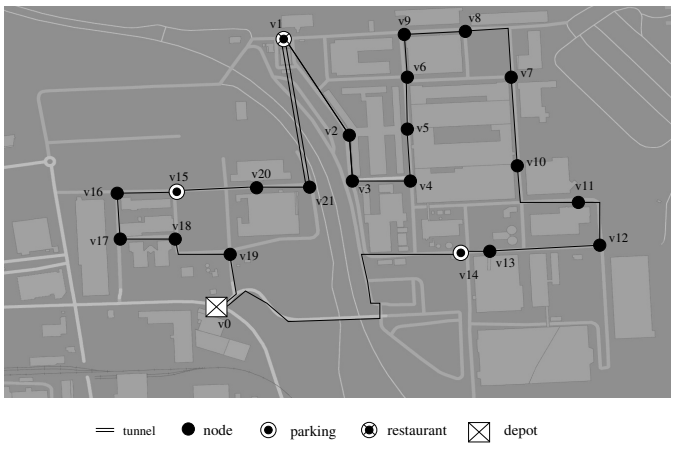


Figure 3.6: This figure illustrates an example of a Hamilton cycle for emergency cases, i.e. all stations (buildings, parkings, restaurant and depot) are covered in this cycle.

Outline

The sequel of this thesis is structured as follows. We study each of the three online transportation problems (Tram Mode Problem, Elevator Mode Problem, and Taxi Mode Problem) separately in Chapters 4, 5 and 6.

In Chapter 4 we study the Tram Mode Problem. We propose different online algorithms. We present a model for computing the optimal offline solution for the $TramMP$ w.r.t. each of the objective functions (total tour length, makespan, total waiting time and total number of stops). Then we evaluate the performance of the proposed algorithms in comparison with the optimal offline solutions in theory (with the help of competitive results), and in practice as well (with the help of some computational results).

In Chapter 5 we study the Elevator Mode Problem. We propose an online algorithm (see Section 5.1). We present a model for computing the optimal offline solution for the EMP w.r.t. each of the objective functions (total tour length, makespan, total waiting time and total number of stops). Then we evaluate the performance of the proposed algorithm in comparison with the optimal offline solutions in theory (with the help of competitive results), and in practice as well (with the help of some computational results).

In Chapter 6, we consider the Taxi Mode Problem, where the objective is a hierarchical one; to maximize the number of accepted customer requests (primary) and to serve them at minimum costs (secondary). This means that, the operator decides which request can be accepted and which has to be rejected. We distinguish two main problems that arise from the Taxi Mode Problem, the Non-Preemptive and the Preemptive Taxi Mode Problems, and provide a solution approach for each of them by means of flows in time-expanded networks. In Section 6.1.1, we present a way to compute optimal offline solutions for the Non-Preemptive Taxi Mode Problem. In Section 6.1.2, we propose a replan strategy for the Online Non-Preemptive Taxi Mode Problem that, in fact, solves the online problem by computing a sequence of offline subproblems on certain subsequences of requests. In Section 6.2.1, we propose two ways to solve the Offline Preemptive Taxi Mode Problem optimally first using a path formulation and second using multicommodity flow. Due to the long computation time of finding a good or an optimal solution in the Preemptive Taxi Mode Problem, we propose a flow-based heuristic. In Section 6.2.2,

we propose a modified replan strategy for the Online Preemptive Taxi Mode Problem that, in fact, solves the online problem by computing a sequence of "heuristic offline subproblems" on certain subsequences of requests. Then, we evaluate the performance of the proposed replan strategies in comparison with the optimal offline solutions in theory (with the help of competitive analysis, in Section 6.3), and in practice as well (with the help of some computational results, in Section 6.4).

Finally, we end this thesis with some concluding remarks on our approaches and on the global fleet management system. We also give some future lines of research and open problems.

Chapter 4

Tram Mode Problem

In this chapter, we treat the PDP related to the tram mode. It is the most restricted operation mode in which VIPAs run on predefined circuits in a predefined direction at a constant speed of about 15km/h and stop at stations to pick up or deliver users. We consider circuits C with one distinguished node, v_0, the origin of the circuit. Circuits can also be lines where one end is the origin and the VIPA can change its direction only at the two ends of the line.

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

- a circuit or a bi-directed path $C = (V, E, d)$, where the nodes in V correspond to stations, edges in E to their links, and edge weights $d : E \to \mathbb{R}_+$ determine the driving times between two nodes $v_i, v_j \in V$ with respect to the distance d corresponding to the length of a shortest path from v_i to v_j.
- a sequence $\sigma = \{r_1, \ldots, r_h\}$ of pdp-requests[1] $r_j = (t_j, x_j, y_j, z_j)$ with $z_j \leq \text{Cap}$,
- 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 Tram Mode Problem is a feasible transportation schedule S serving all requests in σ.

The goal is to construct a transportation schedule $S = \{\Gamma^1, \ldots, \Gamma^k\}$ w.r.t minimizing one of the following objective functions: total tour length, makespan, total waiting time and total number of stops.

[1] In this chapter, the term "request" means pdp-request.

4.1 Online Algorithms

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. Starting from the origin of C, a tour of a VIPA operating in tram mode consists of one or many full rounds called "subtour(s)". Each subtour starts and ends in the origin of the circuit and has a length of $|C|$, the length of the circuit C. For the online situation, we propose the following 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.

SIR starts its tour as soon as a request is released. During the morning respectively the evening, all requests have the same pickup respectively delivery node. Therefore, SIR can be adapted to accumulate the requests up to the capacity Cap of the VIPA before starting its tour. For that, we propose two other algorithms for VIPAs operating in tram mode in the morning respectively 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.

The algorithms SIF_E and SIF_M can be merged together to obtain a version for the lunch scenario, where requests have either the same pickup node or the same delivery node, which is the origin v_0 of the circuit.

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.

4.2 Minimizing the Total Tour Length

4.2.1 Optimal Offline Solution for the TramMP w.r.t. Minimizing the Total Tour Length

To compute an optimal offline solution $OPT(\sigma)$ w.r.t. minimizing the TTL, we will provide two possibilities:

- one as a coloring problem in an interval graph.
- another by means of flows in a suitable network.

In both cases, the optimal solution can be computed in polynomial time. It will turn out that the interpretation as coloring problem is helpful for the argumentation in subsequent proofs, whereas the flow formulation is more practical to perform the computations.

Optimal offline solution via colorings of interval graphs: An interval graph $\mathcal{G}(\mathcal{I})$ is obtained as intersection graph 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 $w(\mathcal{G}(\mathcal{I}))$ 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 $\mathcal{X}(\mathcal{G}(\mathcal{I}))$ of required colors. For interval graphs it was shown in [104] that the following Greedy coloring algorithm always produces an $w(\mathcal{G}(\mathcal{I}))$-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 $C = \{v_o, v_1, \ldots, v_\ell\}$ 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 $P = \{v_0, v_1, \ldots, v_\ell, v_0\}$ 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 $w(G_\sigma)$ 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 w.r.t. TTL
Input: $\sigma = \{r_1, r_2, \ldots, r_m\}$, $C = \{v_o, v_1, \ldots, v_\ell\}$, Cap and k
Output: transportation schedule of minimum total tour length

(i) 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 $\mathcal{I}$,

(ii) sort all intervals in $\mathcal{I}$ in increasing left end points (and in case left end points are equal, in increasing right end points),

(iii) create the interval graph $\mathcal{G}(\mathcal{I})$ and apply the Greedy algorithm to color it,

(iv) wait until t_m (the release time of the last request),

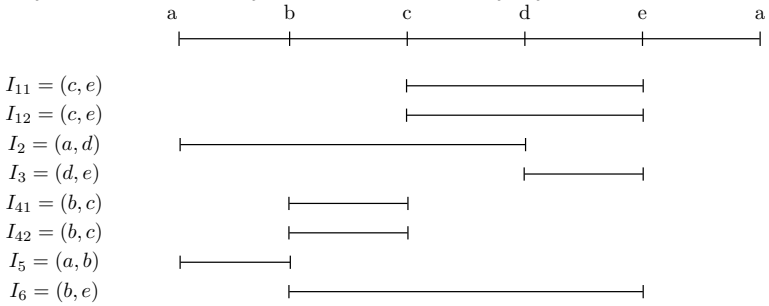
(v) as long as there are unserved requests:

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 (i.e. a VIPA that travels 1 unit of length in 1 unit of time) with capacity Cap = 2, and a sequence σ of 6 requests:

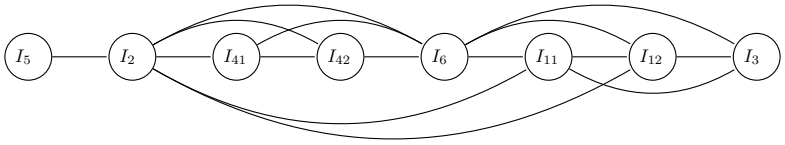
$$r_1 = (1, c, e, 2) \quad r_4 = (4, b, c, 2)$$
$$r_2 = (2, a, d, 1) \quad r_5 = (5, a, b, 1)$$
$$r_3 = (3, d, e, 1) \quad r_6 = (6, b, e, 1)$$

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

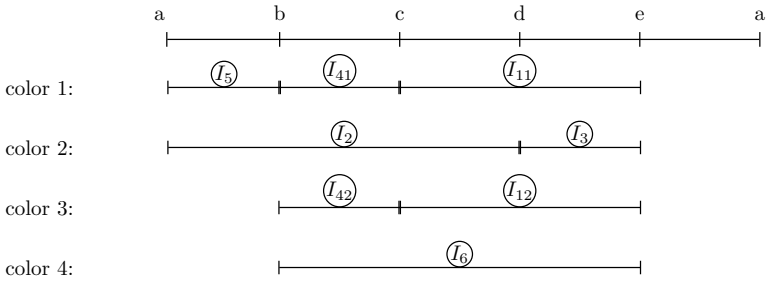


(ii) We sort all intervals in $\mathcal{I}$ according to their left end points: $I_5, I_2, I_{41}, I_{42}, I_6, I_{11}, I_{12}, I_3$.

(iii) We create the interval graph $\mathcal{G}(\mathcal{I})$:



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



(v) As long as there are unserved requests, we select 2 random color classes, assign the corresponding passengers to a VIPA and perform one subtour traversing C to serve them, for instance:

- first VIPA, first round: color 1 and 2 $(r_5, r_2, r_{41}, r_{11}, r_3)$,
- second VIPA, or second round of first VIPA: color 3 and 4 (r_{42}, r_{12}, r_6).

$\Diamond$

Theorem 4.2. *Algorithm* Opt-Tram *provides a load-preemptive optimal solution for the Offline Tram Mode Problem w.r.t. minimizing the total tour length.*

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

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

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 $\left\lceil \frac{w(\mathcal{G}(\mathcal{I}))}{\text{Cap}} \right\rceil$ 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 $w(\mathcal{G}(\mathcal{I}))$-coloring of $\mathcal{G}(\mathcal{I})$, 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

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

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 or enter a VIPA,

  but the so modified algorithm is not necessarily polynomial anymore.

♦

In Example 4.1, if we do not select the color classes randomly, but

- first VIPA, first round contains color 1 and 3 thus, the intervals I_{41}, I_{42} and I_{11}, I_{12} are together, and r_5, r_4 and r_1 are served.

- second VIPA, or second round of first VIPA contains color 2 and 4 s.t. r_2, r_3, and r_6 are served

Then, this selection avoids load-preemption (as I_{11}, I_{12} and I_{41}, I_{42} are served together) and yields the minimum possible number of stops (8) from this coloring. Adding this procedure to the algorithm Opt-Tram changes its complexity. Given the set $Color = \{1 \cdots colors\}$, finding the partition P^* of colors that

- minimizes the total number of stops, and or
- avoids or reduce load-preemption

is a set partitioning problem that is NP-hard. Hereby, this procedure consists of solving a set partitioning problem and Opt-Tram will have an exponential running time if the color classes are not selected randomly.

Optimal Offline Solution w.r.t. minimizing the Total Tour Length via a min-cost flow In order to solve the Optimal Offline Solution for the Tram Mode Problem w.r.t. minimizing the Total Tour Length, we build a demand network $C_D = (V_D, A_D)$ based on σ and the original circuit C.
The node set $V_D = s \cup V_x \cup V_y \cup t$ is composed of

- all origins x_j of all requests r_j in σ, in V_x,
- all destinations y_j of all requests r_j in σ, in V_y,
- the nodes s as source that correspond to the origin of the circuit where all VIPAs are initially placed and t as sink.

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

- source arcs from s to all origins $x_j \in V_x$, in A_+,
- request arcs from each $x_j \in V_x$ to $y_j \in V_y$ in A_R,
- link arcs from all destinations $y_j \in V_y$ to all reachable origins $x_i \in V_x$ such that $y_j \leq x_j$ in driving directions of the VIPAs, i.e. the origin v_0 of the circuit does not lie in the interior of the path (y_j, x_j), in A_L,
- sink arcs from all destinations $y_j \in V_y$ to t in A_-.

This demand network $C_D = (V_D, A_D)$ reflects the requests $r_j \in \sigma$. We require a flow $f(a) = z_j$ on all request arcs $a \in A_R$ to ensure that all requests are served, see constraint (4.1b).

The demand network C_D is acyclic by construction. A flow f through this demand network C_D corresponds to the places occupied by the passengers in the VIPAs. For all internal nodes $v \in V_x \cup V_y$, we use normal flow conservation constraints, (see 4.1c). Flow conservation constraints (4.1c) give rise to a totally unimodular matrix (the node-arc incidence matrix of the digraph underlying the network C_D), therefore integrality constraints are not required. We consider a min-cost flow problem, where the costs

correspond to the traveled distances $c(a) = d(u, v)$ on all arcs $a = (u, v) \in A_D$. The corresponding linear program is detailed in (4.1).

$$\min \sum_{a \in A_D} c(a) f(a) \tag{4.1a}$$

$$\text{s.t. } f(a) = z_j \qquad \forall a \in A_R \tag{4.1b}$$

$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a) \qquad \forall v \in V_x \cup V_y \tag{4.1c}$$

$$f(a) \geq 0 \qquad \forall a \in A_D \tag{4.1d}$$

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). There are no capacity constraints on arcs needed. In fact, for each unit of flow sent from the source s to the destination t, a cost is incurred equal to the sum of costs of all the arcs traversed. As we consider a min-cost flow problem consisting in finding a solution that minimizes the total cost (4.1a) while meeting the demand of all request arcs in the network (4.1b), then capacities on the arcs are taken implicitly.

The linear program (4.1) solves the offline version of the Tram Mode Problem, where the whole sequence σ of requests is known at time $t = 0$, w.r.t. minimizing the total tour length to optimality. The resulting demand network $C_D = (V_D, A_D)$ of the circuit presented in Example 4.1 is shown in Figure 4.1, and the solution computed by the LP (4.1) is shown in Figure 4.2.

Theorem 4.4. *The linear program (4.1) provides an optimal load-preemptive solution of the Offline Tram Mode Problem w.r.t. minimizing the total tour length.*

Proof. Let f^* be the optimal flow according to (4.1). By requiring $f(a) = z_j \ \forall a \in A_R$, it is clear that all requests are served. This demand network C_D is acyclic, and by its construction we have the resulting min-cost flow where

- For a partition of $V_D = V_s \cup V_t$ such that $s \in V_s$ and $t \in V_t$, the subset of arcs $\delta^+(V_s) = \{(u, v) \in A_D : u \in V_s, v \in V_t\}$ is an (s, t)-cut. If $Cut = \delta^+(V_s)$ is a maximal (s, t)-cut in C_D, then $\sum_{a \in A_-} f(a) = \sum_{a \in A_+} f(a) = \sum_{a \in Cut} f(a)$, and this maximal (s, t)-cut in C_D corresponds to the maximum number of passengers traversing a same edge in the original circuit C.
- for each flow unit, the traversed (s, t)-path in C_D by this flow unit corresponds to one place in a VIPA during a subtour in the original circuit C traversing $v_0 \to v_1 \to \ldots \to v_n \to v_0$. Hereby, selecting Cap many such (s, t)-paths creates the load for one VIPA traversing C. Thus, $\left\lceil \frac{val(f^*)}{\text{Cap}} \right\rceil$ many such subtours are needed to serve all requests.

We can note certain similarities between the optimal offline solution computed via colorings of interval graphs and this optimal offline solution computed via a min-cost flow.

The clique number $w(\mathcal{G}(\mathcal{I}))$ corresponds to the value $\sum_{a \in Cut} f(a)$ of the max-cut Cut and the coloring of G_σ corresponds to the (s,t)-paths traversing C, therefore, by using the same strategy, where one VIPA can serve all (uniform) requests from up to Cap color classes in a single subtour traversing C. We can, thus, turn any optimal computed flow f^* in C_D 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 (s,t)-paths 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.

As the considered min-cost flow problem consists in finding a solution that minimizes the total cost while meeting the demand of all request arcs in the network, therefore

$$\sum_{r_j \in \sigma} z_j = \sum_{a \in A_R} f(a)$$

By construction, for each flow unit, the traversed (s,t)-path in C_D by this flow unit contains at least one arc $a \in A_R$. On the other hand, the (s,t)-paths in C_D correspond to an assignment of places in VIPA(s) to passengers, and subtours for the VIPA(s) can be easily created by repeatedly choosing Cap (s,t)-paths and assigning uniform requests (i.e. single passengers) covered by the corresponding (s,t)-paths to the Cap places of one VIPA.

Clearly, at least $\left\lceil \frac{val(f^*)}{\text{Cap}} \right\rceil = \left\lceil \frac{\sum_{a \in Cut} f(a)}{\text{Cap}} \right\rceil$ 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 linear program (4.1) provides an optimal flow f^*, we can guarantee to obtain a feasible transportation schedule performing the minimal number of subtours by always choosing Cap (s,t)-paths (except for the last subtour where we choose all remaining ones) so that the minimal total tour length equals

$$OPT(\sigma) = \left\lceil \frac{val(f^*)}{\text{Cap}} \right\rceil \cdot |C| \,.$$

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 $z_j \leq$ Cap holds). □

Note that the decomposition shown in Theorem 4.4 is not necessarily unique. Furthermore, we have the same implications as mentioned for the coloring approach in Remark 4.3 concerning the selection of (s,t)-paths.

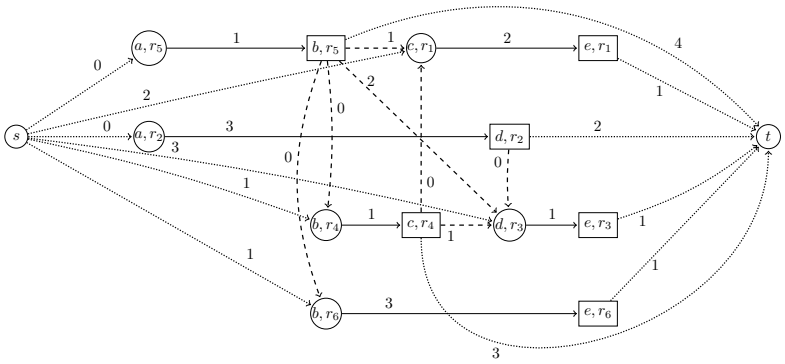


Figure 4.1: This figure illustrates the network $C_D = (V_D, A_D)$ of Example 4.1 built in order to compute the optimal solution w.r.t. minimizing the total tour length. The requests are illustrated by solid arcs, the source and sink arcs by dotted arcs and the link arcs by dashed arcs.

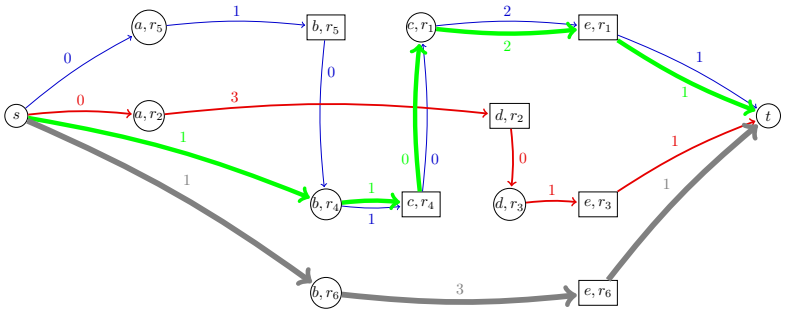


Figure 4.2: This figure illustrates the resulting flow computed by the LP (4.1) on the network $C_D = (V_D, A_D)$ of Example 4.1. The resulting flow has a value of 4. For each flow unit, the corresponding (s, t)-path is indicated by a different color and thickness for its arcs.

4.2.2 Competitive Analysis

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. As competitive results, (ratios) against an oblivious adversary are the strongest, in the analysis of competitive ratios we prove that the algorithms are $c-$ competitive against the oblivious adversary. 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 4.5. Consider a circuit $C = (v_0, v_1, \ldots, v_\ell)$ 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 $\mathrm{Cap} \cdot |C|$ uniform requests that force SIR to perform one full round (subtour) of length $|C| = \ell + 1$ 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 $r_j = ((j-1)\,|C|\,, v_{\ell-1}, v_\ell, 1)$ for $(\ell-1)\mathrm{Cap}+1 \leq j \leq \ell\mathrm{Cap}$
- Cap requests $r_j = ((j-1)\,|C|\,, v_\ell, v_0, 1)$ for $\ell\mathrm{Cap}+1 \leq j \leq (\ell+1)\mathrm{Cap}$

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 $t = |C| = \ell + 1$ 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 $SIR(\sigma) = \mathrm{Cap} \cdot |C| \cdot |C|\,$.

The adversary waits at the origin v_0 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 $OPT(\sigma) = |C|\,$. The tours performed by SIR and OPT are illustrated in Fig 4.3. Therefore, we obtain

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

as a lower bound for the competitive ratio of SIR. ◊

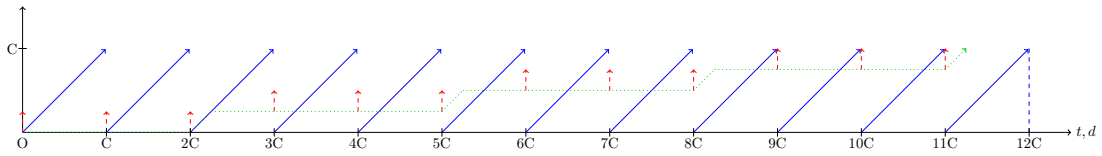


Figure 4.3: This figure illustrates the tour performed by SIR (in blue) and the adversary (dotted in green) in order to serve the requests (dashed arcs in red) from Example 4.5 for $\mathrm{Cap} = 3, \ell = 3$ and $|C| = 4$.

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 4.5, shows that $2 \cdot \mathrm{Cap}$ is a lower bound on the competitive ratio of SIR during lunch time, see Figure 4.4 for an illustration.

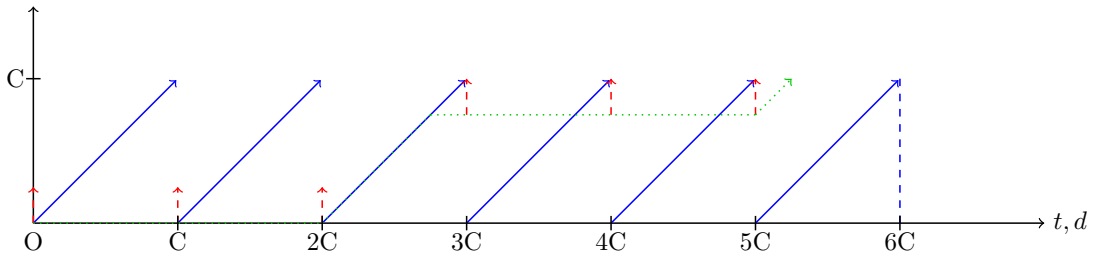


Figure 4.4: This figure illustrates the tour performed by SIR (in blue) and the adversary (dotted in green) in order to serve the first Cap and the last Cap requests (dashed arcs in red) from the sequence presented in Example 4.5 for Cap $= 3, \ell = 3$ and $|C| = 4$. These requests satisfy the criterias of the lunch scenario.

As for the morning respectively 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 4.5 shows that Cap is a lower bound on the competitive ratio of SIR during morning respectively evening, see Figure 4.5 and Figure 4.6, respectively.

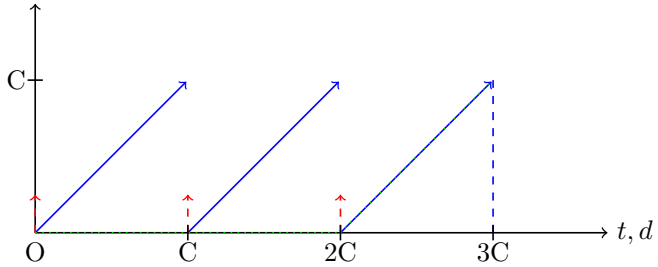


Figure 4.5: This figure illustrates the tour performed by SIR (in blue) and the adversary (dotted in green) in order to serve the first Cap requests (dashed arcs in red) from the sequence presented in Example 4.5 for Cap $= 3, \ell = 3$ and $|C| = 4$. These requests satisfy the criterias of the morning scenario.

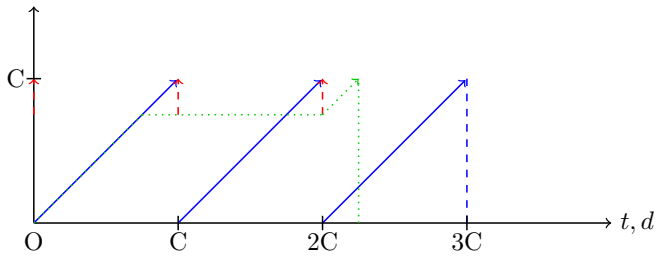


Figure 4.6: This figure illustrates the tour performed by SIR (in blue) and the adversary (dotted in green) in order to serve the last Cap requests (dashed arcs in red) from the sequence presented in Example 4.5 for Cap $= 3, \ell = 3$ and $|C| = 4$. These requests satisfy the criterias of the evening scenario.

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

Theorem 4.6. *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*

- *Cap* $\cdot |C|$*-competitive in general,*
- $2 \cdot$ *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 4.5,
- during lunch: using the sequence σ' restricted to the first Cap and the last Cap requests $(t_j, v_0, v_1, 1)$ and $(t_j, v_\ell, v_0, 1)$ from the sequence σ presented in Example 4.5 as in Figure 4.4,
- 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 $(t_j, v_\ell, v_0, 1)$) from the sequence σ presented in Example 4.5, as Figure 4.5 (resp. Figure 4.6) 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 4.5). This leads to

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

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

  is the maximum possible ratio between $SIR(\sigma)$ and $OPT(\sigma)$ taken over all possible sequences in general.

- $|G_{\sigma'}| = |\sigma'| = 2\text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

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

  is the maximum possible ratio between $SIR(\sigma)$ and $OPT(\sigma)$ taken over all possible sequences during the lunch.

- $|G_{\sigma''}| = |\sigma''| = \text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

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

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

□

As for SIF_M and SIF_E, we can ensure optimality for these two strategies:

Theorem 4.7. *SIF_M (resp. 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.

□

Recall that the two previous algorithms SIF_E and SIF_M can be merged together to obtain a version for the lunch scenario, SIF_L. We show in the next theorem that SIF_L is 2-competitive.

Theorem 4.8. *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 SIF_L results if the requests are released in a way that SIF_L never serves a request from Q_1 with one from Q_2 together, therefore by each subtour of length $|C|$ performed by SIF_L Cap requests are served either from the clique Q_1 or from Q_2, yielding $SIF_L(\sigma) = \lceil \frac{|G_\sigma|}{\text{Cap}} \rceil \cdot |C|$ 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 $OPT(\sigma) = \frac{|G_\sigma|}{2\text{Cap}} \cdot |C|$, therefore

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

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

□

4.3 Minimizing the Makespan

4.3.1 Optimal Offline Solution for the TramMP w.r.t. Minimizing the Makespan

A mixed-integer program with multicommodity flow where the set of commodities corresponds to the set of vehicles, has been proposed for a PDP where time windows are specified for the requests, on a graph where the objective function minimizes the total routing cost (total tour length with waiting time)[44]. We adapted this formulation to the Offline Tram Mode Problem where no time windows are specified but a release time of requests that must be respected and changed the objective function to minimizing the makespan. Accordingly, we propose the following integer program in order to obtain the optimal offline solution $OPT(\sigma)$ w.r.t. minimizing the makespan.

Given a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 as a network $C = (V, E)$, k VIPAs with capacity Cap, and a request sequence σ with n requests $r_j = (t_j, x_j, y_j, z_j)$ such that $z_j \leq \text{Cap}$, otherwise the request is split.

Let m denote the number of requests, i.e., $m = |\sigma|$, to be served. We build a complete directed graph $G_M = (V_M, A_M)$ where the node set $V_M = V_x \cup V_y \cup \{s, t\}$ is composed of

- all origins x_j of all requests r_j in σ in V_x,
- all destinations y_j of all r_j in σ in V_y,
- the nodes s as source and t as sink that correspond to the origin of the circuit C.

With each node $v \in V_M$ is associated a load $q(v)$ such that $q(s) = q(t) = 0$, $q(x_j) = z_j$ and $q(y_j) = -z_j$ for $r_j \in \sigma$. Each request r_j is thus associated with a pickup node x_j and a delivery node y_j. The arc set A_M is composed of link arcs (v, v') for each pair of nodes $v \in V_M$, $v' \in V_M$.

The routing decisions are represented by the variables below.

- The time at which the VIPA vh_i begins service at node $v \in V_M$, is denoted by the decision variable $B_i(v)$.
- The load of the VIPA vh_i after visiting node $v \in V_M$ is denoted by $Q_i(v)$.
- the trace of the VIPA vh_i is reflected by

$$f_i(v, v') = \begin{cases} 1, & \text{if the VIPA } vh_i \text{ travels along the arc } a = (v, v') \\ 0, & \text{otherwise} \end{cases}$$

$$\forall a \in A_M, i \in \{1 \cdots k\}$$

Finally, with each arc $(v, v') \in A_M$ are associated a cost $c(v, v')$ and a driving time $d(v, v')$ corresponding to the distance of the shortest path from v to v' in the circuit $C = (V, E)$. The objective function (4.2a) is to minimize T_{max}, where T_{max} is greater or equal than the time at which the last VIPA arrives at the depot $\max_{i \in \{1 \cdots k\}} B^i(t)$.

The corresponding integer program is detailed in (4.2)

$$\min T_{max} \tag{4.2a}$$

$$s.t.\ T_{max} \geq B_i(t) \qquad \forall i \in \{1 \cdots k\} \tag{4.2b}$$

$$\sum_{i \in \{1 \cdots k\}} \sum_{a \in \delta^-(x_j)} f_i(a) = 1, \qquad \forall x_j \in V_x \tag{4.2c}$$

$$\sum_{a \in \delta^-(x_j)} f_i(a) = \sum_{a \in \delta^-(y_j)} f_i(a) \qquad \forall r_j \in \sigma,\ i \in \{1 \cdots k\} \tag{4.2d}$$

$$\sum_{a \in \delta^-(s)} f_i(a) = 1 \qquad \forall i \in \{1 \cdots k\} \tag{4.2e}$$

$$\sum_{a \in \delta^-(v)} f_i(a) = \sum_{a \in \delta^+(v)} f_i(a) \qquad \forall v \in V_x \cup V_y, i \in \{1 \cdots k\} \tag{4.2f}$$

$$\sum_{a \in \delta^+(t)} f_i(a) = 1 \qquad \forall i \in \{1 \cdots k\} \tag{4.2g}$$

$$B_i(v') \geq (B_i((v) + d(v, v'))f_i(v, v') \qquad \forall v, v' \in V_M, i \in \{1 \cdots k\} \tag{4.2h}$$

$$Q_i(v') \geq (Q_i(v) + q(v))f_i(v, v') \qquad \forall v, v' \in V_M, i \in \{1 \cdots k\} \tag{4.2i}$$

$$B_i(y_j) - B_i(x_j) - d(x_j, y_j) \geq 0 \qquad \forall r_j \in \sigma \qquad (4.2j)$$
$$B_i(t) - B_i(s) \leq T \qquad \forall i \in \{1 \cdots k\} \qquad (4.2k)$$
$$B_i(x_j) \geq t_j \qquad \forall x_j \in V_x, i \in \{1 \cdots k\} \qquad (4.2l)$$
$$\max\{0, q(v)\} \leq Q_i(v) \leq \min\{\text{Cap}, \text{Cap} + q(v)\} \qquad \forall v \in V_M, i \in \{1 \cdots k\} \qquad (4.2m)$$
$$f_i(a) \in \{0, 1\} \qquad \forall, a \in A_M, i \in \{1 \cdots k\} \qquad (4.2n)$$

Constraints (4.2c) and (4.2d) ensure that each request is served exactly once and that the pickup and delivery nodes of a request are visited by the same VIPA. Constraints (4.2e), (4.2f) and (4.2g) guarantee that all tours start at the origin v_0 of the circuit C, flow conservation constraints guarantee that all tours end in t (and thus, in the origin v_0 of the circuit). Consistency of the time and load variables is ensured by constraints (4.2h) and (4.2i). Precedence constraints are imposed through inequalities (4.2j). Finally, inequalities (4.2k) bound the duration of each route (the route of each VIPA cannot exceed the time horizon T) while (4.2l) ensures that a request cannot be served before it is released and (4.2m) impose capacity constraints.

This formulation is non linear because of constraints (4.2h) and (4.2i). By introducing constants $M(v, v')$ and $W(v, v')$, these constraints can be linearized as follows:

$$B_i(v') \geq B_i((v) + d(v, v') - M(v, v')(1 - f_i(v, v')) \qquad \forall v, v' \in V_M$$
$$Q_i(v') \geq Q_i(v) + q(v) - W(v, v')(1 - f_i(v, v')) \qquad \forall v, v' \in V_M$$

The linear program (4.2) provides an optimal non-preemptive solution of the Offline Tram Mode Problem w.r.t. minimizing the makespan [44]. The resulting computed routing decisions of the Example 4.1 are illustrated in Figure 4.7.

It is straightforward to construct the tours from the resulting computed routing decisions. In fact a positive flow $f_i(a) > 0$ on an arc a corresponds to a move of vh_i on the arc $a = (v, v')$ starting at time $B_i(v)$ and arriving at time $B_i(v')$. If $f_i(a) > 0$ and the nodes v and v' correspond to the same station, then a positive flow $f_i(a) > 0$ on an arc a corresponds to a waiting move where the VIPA waits at the same station from time $B_i(v)$ until time $B_i(v')$. All tours start at s at time 0 (thus, at the origin v_0 of the circuit C), and end in t (thus, at the origin v_0 of the circuit C) at time $B_i(t)$. The makespan $\max_{i \in \{1 \cdots k\}} B_i(t)$ is the time when the last VIPA returns to the origin v_0 of the circuit C. The resulting optimal transportation schedule of the Example 4.1 w.r.t. minimizing the makespan is the following.

$$\Gamma^1 = (a,0) \xrightarrow{r_2} (a,2) \xrightarrow{r_5} (a,5) \xrightarrow{r_6} (b,6) \xrightarrow{r_2} (d,8) \xrightarrow{r_3} (e,9) \to (a,10)$$
$$\Gamma^2 = (a,0) \xrightarrow{r_4} (b,3) \to (b,4) \xrightarrow{r_1} (c,5) \to (e,7) \to (a,8)$$

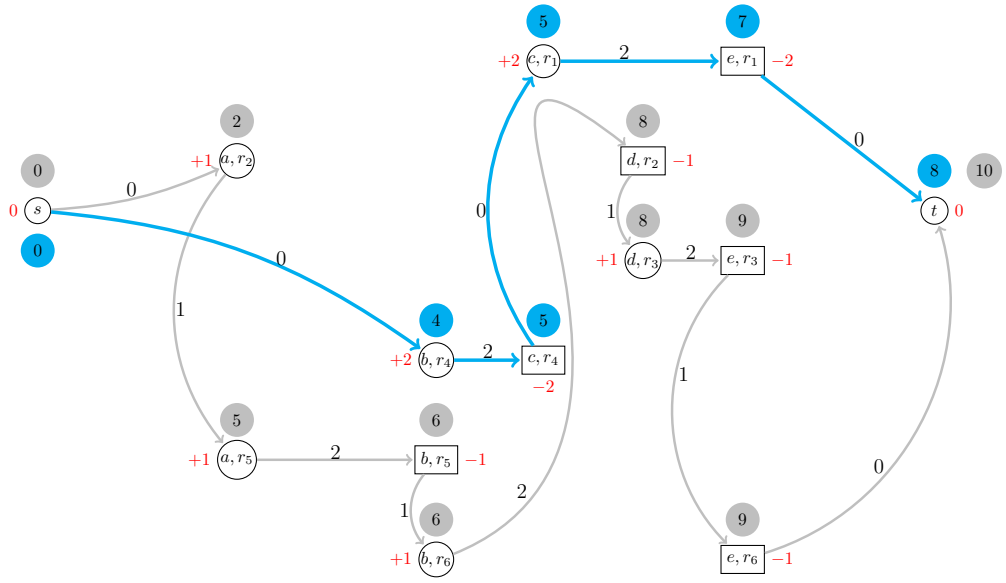


Figure 4.7: This figure illustrates the resulting flow computed by the LP (4.2) that represents the routes of the VIPAs on the network $C_M = (V_M, A_M)$ of Example 4.1. The tour of the first VIPA finishes at time 10, and the tour of the second one finishes at time 8. The makespan has a value of 10, it is indeed minimum because $r_5 = (5, a, b, 1)$ and $r_6 = (6, b, e, 1)$ cannot be served earlier then their release times, and $5 + d(a, b) + d(b, a) = 5 + 1 + 4 = 10$ and $6 + d(b, e) + d(e, a) = 6 + 3 + 1 = 10$. The origins of the requests are represented by circles, the destinations by squares. The values of $B_1(v)$ respectively $B_2(v')$ on the corresponding nodes are illustrated by a value enclosed by a gray respectively blue circle above the nodes. The arcs are labeled by the values of $Q_i(v)$ (the load of the VIPA vh_i after visiting a node v. The values next to the nodes v correspond to the load $q(v)$ of the node, $(q(s) = q(t) = 0)$.

4.3.2 Competitive Analysis

Concerning the objective of minimizing the makespan, the adversary can cheat SIR by releasing a request on a station where the VIPA operated by SIR shortly left. (Again in these competitive analysis we use the oblivious adversary.)

Example 4.9. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and one unit-speed server. The adversary releases a sequence $\sigma = (r_1, r_2)$ with only 2 requests

$$r_1 = (0, v_0, v_i, 1) \text{ and } r_2 = (\varepsilon, v_0, v_i, 1).$$

SIR starts its VIPA at time $t = 0$ to serve r_1. It returns to v_0 at time $t = |C|$ and starts a second round to serve r_2, yielding

$$SIR(\sigma) = 2\,|C|\,.$$

The adversary waits at the origin v_0 until $t = \varepsilon$, starts its VIPA with both requests r_1 and r_2 and is back to v_0 at time $t = |C| + \varepsilon$, yielding

$$OPT(\sigma) = |C| + \varepsilon.$$

$\Diamond$

This gives a lower bound of 2 for the competitive ratio of SIR w.r.t. minimizing the makespan, even during the morning scenario. The same is true for SIF_E during the evening:

Example 4.10. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, and one unit-speed server. The adversary releases a single request

$$r1 = (n, v_n, v_0, \text{Cap})$$

with $z_1 = \text{Cap}$. SIF_E starts its VIPA at time $t = n$ to serve r, arrives at $t = 2n$ at v_n and is back to v_0 at time $t = 2n + 1$, yielding $SIF_E(\sigma) = 2n + 1$. The adversary starts its VIPA at $t = 0$, arrives at v_n at $t = n$ (when r is released) and is back to v_0 at time $t = n + 1$, yielding $OPT(\sigma) = n + 1$. $\Diamond$

This also gives a lower bound of 2 for the competitive ratio of SIF_E w.r.t minimizing the makespan. The same is true for SIF_L during the lunch using the same example. Just for SIF_M, the situation might be better. We conjecture that the following sequence is a worst case for SIF_M:

Example 4.11. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and one unit-speed server. The adversary releases a sequence σ with 3 pdp-requests

$$r_1 = (0, v_0, v_i, 1),\ r_2 = (n + 1, v_0, v_i, \text{Cap} - 1) \text{ and } r_3 = (n + 1, v_0, v_i, 1),$$

announcing that r_3 is the last request in σ. SIF_M starts its VIPA at time $t = n + 1$ fully loaded to serve r_1 and r_2, and finishes the first subtour at time $t = 2n + 2 = 2|C|$. Because r_3 is the last request in σ, SIF_M starts a second subtour to serve r_3 (without being fully loaded) at time $t = 2n + 2$ and is back to v_0 at time $t = 3n + 3$, yielding $SIF_M(\sigma) = 3n + 3$. The adversary starts its VIPA directly at $t = 0$ to serve r_1, is back to the origin v_0 at time $t = n + 1$ and can immediately serve r_2 and r_3 together in a second subtour, finishing its tour at $t = 2n + 2$, yielding $OPT(\sigma) = 2n + 2$.

$\Diamond$

This shows that 3/2 is a lower bound for the competitive ratio of SIF_M w.r.t minimizing the makespan in the morning. We conjecture that this bound is tight.

4.4 Minimizing the Waiting Time

In this section, we present a model to solve the Offline Tram Mode Problem optimally w.r.t. minimizing the Total Waiting Time using a multicommodity flow in a time expanded network based on the original circuit. Then we prove that there is no competitive (deterministic) online algorithm for the Online $TramMP$ w.r.t. minimizing the waiting time.

4.4.1 Optimal offline Solution for the TramMP w.r.t. Minimizing the Total Waiting Time

Given a circuit $C = (v_0, \ldots, v_\ell)$ with origin v_0, k VIPAs with capacity Cap, and a request sequence σ with pdp-requests $r_j = (t_j, x_j, y_j, z_j)$.
We build a time-expanded request network $C_T = (V_T, A_T)$ based on σ and the studied circuit C.
The node set V_T is constructed as follows.
For each station $v \in V$ and each time point $t \in [0, T]$, there is a node $(v, t) \in V_T$, which represents station v at time t as a station where VIPA can simply pass or can pick up or deliver users.
We distinguish $|\sigma|$ subsets V_y^j of nodes in V_T where each subset V_y^j consists of all possible destinations (y_j, t_j^{drop}) of the corresponding request r_j, for all $t_j \leq t_j^{pick} \leq T - d(x_j, y_j) - d(y_j, v_0)$. Note that t_j^{pick} and t_j^{drop} come from the pdp-task τ_j generated by the operator in order to serve the request r_j where t_j^{pick} respectively t_j^{drop} is the time at which the passengers of r_j should be picked up respectively delivered.

The arc set $A_T = A_W \cup A_M \cup A_-$ is composed of

- wait arcs, from $(v, t) \in V_T$ to $(v, t+1)$ with $t \in \{0, 1, \ldots, T-1\}$ in A_W
- transport arcs, from $(v, t) \in V_T$ to $(v', t + d(v, v'))$ for each edge (v, v') of C and each time point $t \in T$ with $t + d(v, v') \leq T$, in A_M
- sink arcs, from all destinations $(y_j, t_j^{drop}) \in V_y^j$ to (v_0, T), in A_-^j. We have $A_- = \bigcup_{j=1}^{|\sigma|} A_-^j$.

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

On the time-expanded network C_T, we define a VIPA flow F to encode the route of the VIPAs through C_T and a set of commodities f_j to represent the routing of each request $r_j \in \sigma$.

To correctly initialize the system, we use the node $(v_0, 0) \in V_T$ and $(v_0, T) \in V_T$ as source and sink for the flow F and set the balance of the source accordingly to the number k of available vehicles, see (4.3b). For all internal nodes $(v, t) \in V_T \setminus \{(v_0, 0), (v_0, T)\}$, we use normal flow conservation constraints, see (4.3c), which also automatically ensures that a flow F of value k is entering the sink (v_0, T).

We use, for each request r_j, the node (x_j, t_j) as source of the corresponding commodity f_j and the node (v_0, T) as the commodity's sink. As each commodity f_j corresponds

to request r_j, the flow f_j is bounded by z_j accordingly, but we may avoid load preemption by requiring that $f_j \in \{0,1\}$. To ensure that a request is served but not more than once, we require that exactly one outgoing arc from the commodity's origin (x_j, t_j) of the corresponding request r_j is chosen, see (4.3d). We use normal flow conservation constraints, see (4.3f), which also automatically ensures that, for each request r_j, a flow of 1 of its corresponding commodity f_j is leaving the source node, $x_j, t_j)$, of the commodity, and a flow of 1 is entering the sink node. To ensure that passengers are dropped at their corresponding destinations, we require for each commodity f_j that exactly one arc from the corresponding sink arcs A^j_- is chosen, see (4.3e). To ensure that the capacity of the VIPA is respected on all arcs $a \in A_M$, we require that

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

Thus, the capacities for F on the transportation arcs are not given by constants but by a function.

Note that, due to these flow coupling constraints (4.3g), the constraint matrix of the ILP is not totally unimodular (as in the case of uncoupled flows) and therefore integrality constraints for both flows are required (4.3h) and (4.3i), reflecting that solving the problem is $\mathcal{NP}$-hard.

We consider a min-cost flow problem. Accordingly, our objective function (4.3a) considers costs $c(a)$ only on the wait arcs for the commodities f_j. The corresponding integer linear program is detailed in (4.3).

$$\min \sum_{r_j \in \sigma} \sum_{a \in A_W} c(a) f_j(a) z_j \tag{4.3a}$$

$$\sum_{a \in \delta^+(v_0,0)} F(a) = k \tag{4.3b}$$

$$\sum_{a \in \delta^-(v,t)} F(a) = \sum_{a \in \delta^+(v,t)} F(a) \qquad \forall (v,t) \neq (v_0,0), (v_0,T) \tag{4.3c}$$

$$\sum_{a \in \delta^-(x_j,t_j)} f_j(a) = 1 \qquad \forall r_j \in \sigma \tag{4.3d}$$

$$\sum_{a \in A^j_-} f_j(a) = 1 \qquad \forall r_j \in \sigma \tag{4.3e}$$

$$\sum_{a \in \delta^-(v,t)} f_j(a) = \sum_{a \in \delta^+(v,t)} f_j(a) \qquad \forall r_j \in \sigma, \forall (v,t) \neq (x_j,t_j) \tag{4.3f}$$

$$\sum_{r_j \in \sigma} f_j(a) \cdot z_j \leq \text{Cap} \cdot F(a) \qquad \forall a \in A_M \tag{4.3g}$$

$$F(a) \in \{0,1\} \qquad \forall a \in A_T \tag{4.3h}$$

$$f_j(a) \in \{0,1\} \qquad \forall a \in A_T, \forall r_j \in \sigma \tag{4.3i}$$

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

The integer linear program (4.3) solves the offline version of the Tram Mode Problem w.r.t. minimizing the total waiting time (where the whole sequence σ of requests is known at time $t = 0$) to optimality. The resulting computed flows of the Example 4.1 are illustrated in Figure 4.8.

Remark 4.12. This model with $F(a) \leq 1 \ \forall a \in A_T$ handles the condition that no two VIPAs must use a same arc simultaneously such that there is no VIPA that blocks another, and the significant reliability requirements of using the VIPAs are ensured. ⧫

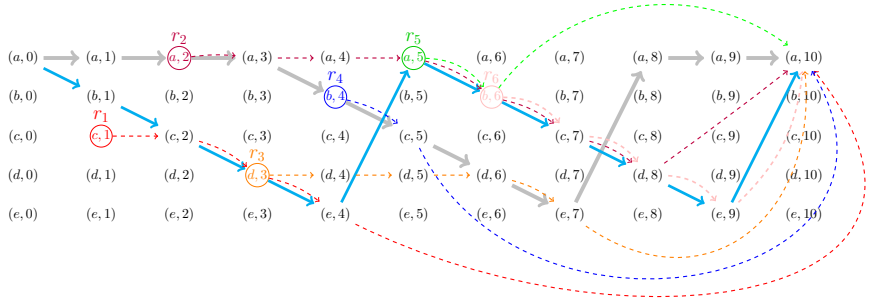


Figure 4.8: This figure illustrates the resulting flow computed by the LP (4.3) on the network $C_T = (V_T, A_T)$ of Example 4.1. The resulting flow has a value of 8 as a total waiting time. We identify the route of the VIPAs from the arcs a with $F(a) > 0$ represented by solid arcs. Each request r_j corresponds to a commodity f_j and the nodes enclosed by circles illustrate the corresponding commodity's origins. For each request, the corresponding arcs a with $f_j(a) > 0$ are indicated by dashed arcs with a different color.

Theorem 4.13. *The integer linear program (4.3) solves the offline version of the Tram Mode Problem w.r.t. minimizing the total waiting time (where the whole sequence σ of requests is known at time $t = 0$) to optimality.*

Proof. From constraints (4.3b) it is ensured that there are exactly k available VIPAs at the start station. By starting the flow F from the depot $(v_0, 0)$, it is ensured that each VIPA starts its tour from the depot, and with the constraint (4.3c) we ensure that there exists k paths P in G_T from the source node $(v_0, 0)$ to the sink node (v_0, T). The tracks of the VIPAs over time can be recovered from the flow $F(a)$ on the arcs by standard flow decomposition, see [2]. Hereby, VIPA flows on transportation arcs correspond to a move in the tour of the corresponding VIPA. After constructing a sequence of moves $\mathcal{M}$ for the K tours for the VIPAs, we insert between them the actions. The actions are incurred by the set of commodities f_j. Hereby, from a positive flow for a commodity f_j, we can determine through which arcs, the request r_j is served by determining the arcs a that

have a $f_j(a) = 1$. The set of arcs having $f_j(a) = 1$ is the (s,t)-path of the commodity f_j from the commodity's origin (x_j, p_j) to the commodity's destination (v_0, T) (there cannot be more than one $(s,t) - path$ for each commodity f_j by constraints (4.3d), (4.3e) and (4.3f). A request r_j served through the (s,t)-path implies that a VIPA traverses the arcs from (x_j, t_j^{pick}) to (y_j, t_j^{drop}). (constraints (4.3g)), therefore we can add the corresponding pickup and delivery actions between the corresponding moves of the tour that contains these arcs. □

4.4.2 Competitive Analysis

Concerning the competitive analysis of the Online Tram Mode Problem w.r.t. minimizing the waiting time, we obtain the more general result that no (deterministic) online algorithm for the Online $TramMP$ is competitive w.r.t. minimizing the waiting time against two common types of adversaries.

We first consider an **oblivious adversary** who knows the complete behavior of a (deterministic) online algorithm ALG and chooses a worst-case sequence for ALG. Hereby, an oblivious adversary is allowed to move VIPAs towards the origins x_j of not yet released requests r_j (but also has to respect the release time t_j to serve accepted requests r_j).

Theorem 4.14. *There is no competitive (deterministic) online algorithm for the Online $TramMP$ w.r.t. minimizing the waiting time against an oblivious adversary.*

Proof. The adversary can cheat any online algorithm ALG for this problem by releasing Cap requests as follows. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and one unit-speed server with capacity Cap. The adversary releases a sequence $\sigma = (r_1, \ldots, r_{\text{Cap}})$ of Cap equal requests

$$r_i = (|C| - 1, v_n, v_0, 1).$$

The adversary starts its VIPA at time $t = 0$ to arrive at v_n when the requests are released, and serves them all immediately, yielding

$$OPT(\sigma) = 0.$$

ALG starts its tour at $t = |C| - 1$, arrives to v_n at $2(|C| - 1)$ and serves all Cap requests, yielding

$$ALG(\sigma) = \text{Cap} \cdot (|C| - 1).$$

this shows

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

so that there is no finite number c bounding the ratio between $OPT(\sigma)$ and $ALG(\sigma)$ for all possible request sequences σ of the Online $TramMP$ w.r.t. minimizing the waiting time. □

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

We show that also no (deterministic) online algorithm ALG for the Online $TramMP$ is competitive w.r.t. minimizing the waiting time against a non-abusive adversary.

Theorem 4.15. *There is no competitive (deterministic) online algorithm for the Online $TramMP$ w.r.t. minimizing the waiting time against a non-abusive adversary.*

Proof. Any online algorithm designed for the Online Tram Mode Problem shall have the following choices at each time t at any position $v \in V$ of the circuit:

- either to start moving to serve a released request,
- or to wait and accumulates request to serve them.

We show that we can construct a sequence in phases that can force ALG to make the customers waiting (in all cases whether ALG decides to move or to wait) while OPT can serve all requests without waiting time. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and one unit-speed server with capacity Cap. The adversary first releases one request $r_1 = (0, v_0, v_i, 1)$ for some $i \in \{1, \ldots, n\}$.

Case 1: ALG starts at time $t = 0$ to serve r_1 only, performs a full round and would be back to v_1 at time $t = |C|$. But the adversary releases one request r_2 shortly after ALG has started his tour $r_2 = (\varepsilon, v_0, v_i, \text{Cap} - 1)$ for some $i \in \{1, \ldots, n\}$. Therefore, ALG will start a second round to serve r_2 at time $t = |C|$, and each passenger of r_2 has waited for $|C|$ yielding

$$ALG(\sigma) = |C| \cdot (\text{Cap} - 1).$$

The adversary waits at the origin v_0 until $t = \varepsilon$, starts its VIPA to serve both requests r_1 and r_2 at once, yielding

$$OPT(\sigma) = \varepsilon.$$

Therefore, we obtain

$$\frac{ALG(\sigma)}{OPT(\sigma)} = \frac{|C| \cdot (\text{Cap} - 1)}{\varepsilon} \to \infty$$

(when choosing ε arbitrarily small).

Case 2: ALG decided to wait and delay the service of r_1 in an attempt to take advantage of future released requests to serve them together. But OPT starts immediately to serve r_1. The adversary waits until $t = |C|$ to release a last request $r_2 = (|C|, v_0, v_i, \text{Cap} - 1) \forall i \in \{1, \ldots, n\}$. Then, OPT is back from his first round and can start his second round immediately to serve the second request, yielding

$$OPT(\sigma) = 0.$$

However, ALG starts his first round to serve r_1 and r_2 together (Cap customers), yielding

$$ALG(\sigma) = |C| .$$

this shows

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

so that there is no finite number c bounding the ratio between $OPT(\sigma)$ and $ALG(\sigma)$ for all possible request sequences σ of the Online $TramMP$ w.r.t. minimizing the waiting time against a non-abusive adversary.

□

Hereby, the ratio between $OPT(\sigma)$ and $ALG(\sigma)$ is unbounded even if the requests have a specific pattern. Thus, the result does not become better for the morning, evening or lunch scenarios for any of the algorithms SIF_M, SIF_E, SIF_L and SIR. Table 4.1 details the strategy used by the adversary according to the online algorithm used.

Corollary 4.16. *None of the algorithms SIF_M, SIF_E, SIF_L and SIR is competitive w.r.t. minimizing the total waiting time.*

Table 4.1: The strategy used by the adversary according to the online algorithm used in TramMP w.r.t. minimizing the total waiting time.

Online Algorithm	**Oblivious adversary**	**Non-abusive adversary**
$\boldsymbol{SIF_M}$	the oblivious adversary has the same power as the non-abusive adversary. SIF_M uses the strategy of delaying the service of requests. The adversary releases the same request sequence as in case 2 of Theorem 4.15.	
$\boldsymbol{SIF_E}$	the oblivious adversary releases a sequence of Cap requests as the example of Theorem 4.14.	the non-abusive adversary uses the same strategy as in case 2 of Theorem 4.15, but releases one request $(0, v_n, v_0, 1)$.
$\boldsymbol{SIF_L}$		the non-abusive adversary releases a sequence similar to the one in case 2 of Theorem 4.15.
$\boldsymbol{SIR}$	the oblivious adversary uses the same strategy as the example of Theorem 4.14 but releases only one request.	the non-abusive adversary releases a sequence similar to the one in case 1 of Theorem 4.15.

4.5 Minimizing the Number of Stops

An ILP formulation of the static version (where all requests are known in advance) of the TramMP has been proposed in [109]. We can turn any optimal solution of the static version of the TramMP to an optimal solution for the offline TramMP w.r.t. minimizing the total number of stops by

- waiting until time t_m in the origin v_0 (to ensure that all requests are released before they are served),
- solving the static version of the TramMP.

4.5.1 Optimal Offline Solution for the TramMP w.r.t. Minimizing the Total Number of Stops

The offline TramMP w.r.t. minimizing the total number of stops may be viewed as an extension of interval graph coloring models (see, e.g., [69]). We may use the proposed methods of Section 4.2.1 as a base to compute an optimal offline solution. In the following, we briefly present the method that may be applied.

In order to compute the optimal offline solution w.r.t. minimizing the total number of stops we need to

- wait until time t_m in the origin v_0 (to ensure that all requests are released before they are served),
- compute the optimal offline solution w.r.t. minimizing the total number of stops assuming that the VIPA has a unit capacity, by solving an LP similar to the one proposed in Section 4.2.1 w.r.t. the total tour length. In this case, the objective function considers costs $d(a) = d(u, v)$ for the flow f only on link arcs $a = (u, v)$ in A_L, where $d(u, v)$ is the length of a shortest path from u to v in the circuit C,
- solve a partitioning problem to partition the paths obtained into subtours for the VIPAs such that the total number of stops is minimized.

Remark 4.17. We may also use the method proposed in Section 4.2.1 of colorings of interval graphs to compute the optimal offline solution w.r.t. minimizing the total number of stops assuming that the VIPA has a unit capacity. But the algorithm needs to be modified (the color classes are not selected randomly). ⧫

4.5.2 Competitive Analysis

Concerning the objective of minimizing the number of stops, the strategy of the adversary is to force the online algorithm to stop to pick up or deliver one passenger at a time by serving only one request per subtour with 2 stops, whereas the adversary can pick up and deliver Cap passengers at each stop. We first show a lower bound for SIR.

Example 4.18. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, and one unit-speed server. The adversary releases a sequence σ of $\mathrm{Cap} \cdot n$ pdp-requests similar to the sequence presented in Example 4.5 that force SIR to perform one full round (subtour) with two stops for each request, whereas the adversary is able to serve all requests in a single subtour using $n+1$ stops:

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

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

SIR starts its VIPA at time $t = 0$ to serve $r_1 = (0, v_0, v_1, 1)$, stops to pick up the passenger from v_0 and to deliver him at v_1, and finishes the first subtour without serving any further request. When the VIPA operated by SIR is back to the origin v_0, the second request $r_2 = (|C|\,, v_1, v_2, 1)$ is released and SIR starts at $t = |C|$ a second subtour using 2 stops to serve r_2, without serving any further request in this subtour. This is repeated for each request yielding $SIR(\sigma) = 2 \cdot \mathrm{Cap} \cdot (n+1)$.
The adversary waits at the origin v_0 until $t = (\mathrm{Cap}-1)\,|C|$ and serves all requests $r_1, \ldots, r_{\mathrm{Cap}}$ from v_0 to v_1 (2 stops). 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 (1 stop v_2). This is repeated for all Cap requests from v_i to v_{i+1}, yielding $OPT(\sigma) = (n+1)+1$. Therefore, we obtain

$$\frac{SIR(\sigma)}{OPT(\sigma)} = \frac{2 \cdot \mathrm{Cap} \cdot (n+1)}{(n+1)+1} = 2 \cdot \mathrm{Cap}$$

as a lower bound for the competitive ratio of SIR. ◊

In the special case of the lunch scenario, a sequence σ', containing the first Cap and the last Cap requests from the sequence presented in Example 4.18, shows that $SIR(\sigma') = 2 \cdot 2 \cdot \mathrm{Cap}$ and $OPT(\sigma') = 3$ yielding $\frac{4}{3}\mathrm{Cap}$ as a lower bound of the competitive ratio of SIR, see Figure 4.4 for an illustration.

As for the morning respectively evening scenario, a sequence σ'', containing the first Cap resp. last Cap requests from the sequence presented in Example 4.5, shows that Cap is a lower bound on the competitive ratio of SIR, see Figure 4.5 (Figure 4.6 respectively).

We can prove that the previously presented examples are indeed worst cases for SIR w.r.t. minimizing the total number of stops:

Theorem 4.19. *For one or several VIPAs with capacity Cap operating in tram mode on a circuit C, SIR is w.r.t the objective of minimizing the total number of stops*

- $2 \cdot$ *Cap-competitive in general,*

- $\frac{4}{3} \cdot$ *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 uses 2 stops for each color (i.e. serving a single uniform request only per subtour), yielding $|G_\sigma| \cdot 2$ as total number of stops, whereas the adversary can accumulate the requests and pick up Cap passengers and deliver other Cap passengers every time he stops except the first and last stop in his tour.

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 4.18,
- during lunch: using the sequence σ' restricted to the first Cap and the last Cap requests $(t_j, v_0, v_1, 1)$ and $(t_j, v_\ell, v_0, 1)$ from the sequence σ presented in Example 4.18 as in Figure 4.4,
- 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 $(t_j, v_\ell, v_0, 1)$) from the sequence σ presented in Example 4.5, as Figure 4.5 (respectively Figure 4.6) shows.

Furthermore, to maximize the ratio between this total number of stops obtained by SIR and the optimal offline solution, we need to ensure that all requests in σ can be served with as few stops 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 with $\ell + 1$ stops suffices to serve all of them (see again Example 4.18). This leads to

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

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

  is the maximum possible ratio between $SIR(\sigma)$ and $OPT(\sigma)$ taken over all possible sequences in general.

- $|G_{\sigma'}| = |\sigma'| = 2\text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

  $$\frac{SIR(\sigma')}{OPT(\sigma')} = \frac{2 \cdot \text{Cap} \cdot 2}{3} = \frac{4}{3} \cdot \text{Cap}$$

  is the maximum possible ratio between $SIR(\sigma)$ and $OPT(\sigma)$ taken over all possible sequences during the lunch.

- $|G_{\sigma''}| = |\sigma''| = \text{Cap}$ and $w(\mathcal{G}(\mathcal{I})) = \text{Cap}$ s.t.

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

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

□

As for SIF_M respectively SIF_E, the lower bound can be improved by the fact that, the requests have the same origin respectively the same destination, and the VIPA starts its subtour only when enough requests are released to reach Cap passengers. The strategy of the adversary is to force SIF_M (respectively for SIF_E) to deliver (respectively to pick up) one passenger at each stop except the origin v_0 of the circuit.

The adversary can cheat SIF_M by releasing $\text{Cap} \cdot \text{Cap}$ requests as follows.

Example 4.20. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and a server with capacity Cap s.t. $\text{Cap} \leq n$. The adversary releases a sequence σ of $\text{Cap} \cdot \text{Cap}$ uniform pdp-requests that force SIF_M to deliver one passenger at each stop, whereas the adversary is able to serve Cap requests in a single subtour using 2 stops:

$$\text{for } \ 1 \leq j \leq \text{Cap} \qquad \begin{cases} r_j = & (0, v_0, v_1, 1)) \\ r_j = & (0, v_0, v_2, 1) \\ \vdots & \\ r_j = & (0, v_0, v_{\text{Cap}}, 1) \end{cases}$$

$$\text{for } \ \text{Cap} + 1 \leq j \leq 2\text{Cap} \qquad \begin{cases} r_j = & (1 \cdot |C|, v_0, v_1, 1)) \\ r_j = & (1 \cdot |C|, v_0, v_2, 1) \\ \vdots & \\ r_j = & (1 \cdot |C|, v_0, v_{\text{Cap}}, 1) \end{cases}$$

$$\vdots$$

$$\text{for } (\text{Cap} - 1) \cdot \text{Cap} + 1 \leq j \leq \text{Cap} \cdot \text{Cap} \begin{cases} r_j = & (\text{Cap} \cdot |C|, v_0, v_1, 1)) \\ r_j = & (\text{Cap} \cdot |C|, v_0, v_2, 1) \\ \vdots & \\ r_j = & (\text{Cap} \cdot |C|, v_0, v_{\text{Cap}}, 1) \end{cases}$$

SIF_M starts its VIPA at time $t = 0$ to serve the first Cap requests, stops to pick up the Cap passengers from the origin v_0 and then stops at v_1 to deliver the first passenger, at v_2 to deliver the second, this is repeated for all Cap requests arriving to v_{cap}. Thus, SIF_M finishes the first subtour serving Cap requests with $\text{Cap} + 1$ stops. When the VIPA operated by SIF_M is back to the origin v_0, the next Cap requests for $\text{Cap} + 1 \leq j \leq 2\text{Cap}$ are released and SIF_M starts at $t = |C|$ a second subtour serving

Cap requests with Cap + 1 stops. This is repeated for each subset of Cap requests yielding $SIF_M(\sigma) = \text{Cap}(\text{Cap} + 1)$

The adversary waits at the origin v_0 until $t = \text{Cap} \cdot |C|$ and serves the Cap requests $r_1, r_{\text{Cap}+1}, \cdots, r_{(\text{Cap}-1)\cdot\text{Cap}+1}$ going from v_0 to v_1 at once, he stops at v_0 to pick up all Cap and then at v_1 to deliver them. This is repeated for each subset of Cap requests having the clustered by their destinations yielding $OPT(\sigma) = \text{Cap} \cdot 2$. Therefore, we obtain

$$\frac{SIF_M(\sigma)}{OPT(\sigma)} = \frac{\text{Cap}(\text{Cap} + 1)}{\text{Cap} \cdot 2} = \frac{\text{Cap} + 1}{2}$$

as a lower bound for the competitive ratio of SIF_M. ◊

The same is true for SIF_E during the evening using the same example but switching the origins and destinations. We conjecture that this lower bound is tight for SIF_M and SIF_E For SIF_L the lower bound is Cap + 1 as the next example shows.

Example 4.21. Consider a circuit $C = (v_0, \ldots, v_n)$ with origin v_0 and a server with capacity Cap s.t. $\text{Cap} \leq n$. The adversary releases a sequence σ of $2 \cdot \text{Cap} \cdot \text{Cap}$ uniform pdp-requests similar to the sequence presented in Example 4.20 but with $\text{Cap} \cdot \text{Cap}$ requests having the origin of the circuit as their destination. This sequence forces SIF_L to deliver one passenger at each stop for the first $\text{Cap} \cdot \text{Cap}$ requests, and to pick up one passenger at each stop for the last Cap·Cap requests, yielding $SIF_L(\sigma) = 2\text{Cap}(\text{Cap}+1)$, whereas the adversary is able to serve 2Cap requests in a single subtour using 2 stops except the last subtour using 3 stops, yielding $OPT(\sigma) = \text{Cap} \cdot 2 + 1$. Therefore, we obtain

$$\frac{SIF_L(\sigma)}{OPT(\sigma)} = \frac{2\text{Cap}(\text{Cap} + 1)}{\text{Cap} \cdot 2 + 1} = \text{Cap} + 1$$

as a lower bound for the competitive ratio of SIF_L. ◊

4.6 Computational Results

This section deals with computational experiments for the proposed online algorithms w.r.t. different objective functions (total tour length, makespan and total waiting time). 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 4.2-4.8 support this expectation.

They compare the total tour length TTL, the makespan MS, and the total waiting time TWT computed by the online algorithms SIR, SIF_M, SIF_E, SIF_L with the corresponding optimal offline solution OPT.

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 [112].

The computations are performed with the help of a simulation tool developed by Yan Zhao [123]. The instances use a circuit as subnetwork with 1-10 VIPAs, 5-200

requests, represented by m in the tables, 1-12 as the maximum load z_j of a request. For each period (morning, evening, lunch and general), for every parameter set (number of requests 5, 20 and 200), we created 50 test instances and compute them varying the number of VIPAs used and varying their capacity 1-10.

In these test instances, the time horizon is $[0, 130]$ and the length of the circuit used is equal to $|C| = 25$. In this table, the instances are grouped by first the period of the time (morning, evening, lunch, and general), then for each period, the instances are grouped by the number of requests (5, 20 and 100) and the capacity of the VIPA (1, 5 and 10). 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_M, SIF_E, SIF_L have been implemented in Java. For solving the integer linear programs to get the optimal solutions w.r.t. the different objective functions we use Gurobi 8.21. In the following tables, the instances are grouped by number of requests (1st column) and the capacity (2nd column). In Tables 4.2, 4.4 and 4.9 where the computational results w.r.t. minimizing the total tour length are shown, the number of VIPAs does not affect the total tour length. Whether we use 1, 2 or 5 VIPAs the total tour length remains the same. Table 4.2 shows the values of the total tour length SIR^{TTL} obtained by SIR for several test instances for the general scenario, where requests have no common origins or common destinations in comparison to the value of the optimal offline solution OPT^{TTL} w.r.t. minimizing the total tour length and the ratio between them. The competitive ratio c is shown for each of the scenarios, it is always greater than $\frac{SIR^{TTL}}{OPT^{TTL}}$.

Table 4.2: This table shows the total tour length computed by SIR SIR^{TTL} in comparison to the optimal offline solution OPT^{TTL} in general and the ratio between them.

SIR general, $\|C\| = 25$					
m	Cap	SIR^{TTL}	OPT^{TTL}	$\frac{SIR^{TTL}}{OPT^{TTL}}$	$c = \text{Cap} \cdot \|C\|$
5	1	113.75	106.25	1.35	25
5	5	113.75	25	5.25	125
5	10	87.5	25	3.5	250
20	1	453.75	337.5	1.76	25
20	5	287.5	75	3.83	125
20	10	168.75	50	3.38	250
200	1	4700	2506.25	1.87	25
200	5	2600	550	4.73	125
200	10	2168.75	275	7.89	250

In Table 4.2, the competitive ratio of SIR in the general case (where the requests are arbitrary) is far from being reachable. In Table 4.3, we can notice that the average ratio between SIR and OPT w.r.t. minimizing the makespan is greater in case we use only one VIPA. The difference between ratios when using 2 and 5 VIPAs is not remarkable. The improvement of the ratio in average when using 5 VIPAs instead of 2 is only 1.68%. The average is computed only for the instances of 5 and 20 requests. Thus, when we have a small number of requests, it is of no interest to operate more than 2 VIPAs on the circuit. Concerning the objective of minimizing the total waiting time, we can notice that while using 5 VIPAs with instances of 5 requests, the competitive ratio is reached (∞). This is because OPT will behave as the oblivious adversary. OPT is allowed to move VIPAs

Table 4.3: This table shows the computational results for several test instances of the algorithm SIR for the general scenario, w.r.t. different objective functions makespan MS and total waiting time TWT in comparison to the value of the respective optimal offline solutions OPT^{MS} and OPT^{TWT} and the ratio between them. The competitive ratio c or a lower bound LB is shown for each objective function. The results are grouped by the number of VIPA used. A hyphen '-' indicates that no solution has been found (due to the restricted time horizon). A cross '×' indicates that no optimal solution has been found by the ILP within four hours and therefore a cross '×' in the ratio column indicates that no ratio can be computed due to the absence of the optimal solution. In case the solution is preceded by a percentage, it means that only this percentage of instances give a solution and the others are infeasible.

		SIR general, MS, LB=2			SIR general, TWT, $c = \infty$		
m	Cap	SIR^{MS}	OPT^{MS}	$\frac{SIR^{MS}}{OPT^{MS}}$	SIR^{TWT}	OPT^{TWT}	$\frac{SIR^{TWT})}{OPT^{TWT}}$
One VIPA							
5	1	123.75	106.25	1.16	42.6	37.25	1.14
5	5	123.75	90.6	1.37	42.6	17.25	2.47
5	10	123.75	90.6	1.37	42.6	17.25	2.47
20	1	-	-	-	-	-	-
20	5	-	-	-	-	-	-
20	10	-	-	-	-	-	-
200	1	-	-	-	-	-	-
200	5	-	-	-	-	-	-
200	10	-	-	-	-	-	-
Two VIPAs							
5	1	95.7	82.4	1.16	42.6	37.25	1.14
5	5	91.6	76.5	1.20	42.6	17.25	2.47
5	10	91.6	76.6	1.20	42.6	17.25	2.47
20	1	-	-	-	-	-	-
20	5	119.4	102.5	1.16	172.5	54.34	3.17
20	10	104.4	90.4	1.15	154.5	52.12	2.96
200	1	-	-	-	-	-	-
200	5	-	-	-	-	-	-
200	10	-	-	-	365.5	×	×
Five VIPAs							
5	1	95.7	82.4	1.16	18.4	0	∞
5	5	91.6	76.5	1.20	36.5	0	∞
5	10	91.6	76.6	1.20	36.5	0	∞
20	1	123.5	95.5	1.29	128.65	57.6	2.23
20	5	103.4	90.23	1.15	84.6	27.5	3.08
20	10	92.6	80	1.16	84.6	27.5	3.08
200	1	-	-	-	-	-	-
200	5	(29%)124.8	×	×	665.5	×	×
200	10	121.6	×	×	253.4	×	×

towards the origins x_j of not yet released requests r_j (but also has to respect the release time t_j to serve accepted requests r_j), leading to 0 as total waiting time of customers.

In Table 4.4, we can notice that the ratio of SIF_L (in average 1.16%) is better than the ratio of SIR (in average 2.94%).

In Table 4.5, we can notice that there is approximately no improvement of the ratio of SIR and SIF_L w.r.t. minimizing the makespan if we use 5 VIPAs instead of 2. Overall, SIF_L leads to better ratios than SIR during the lunch. SIF_L could serve 20 requests

Table 4.4: This table shows the total tour length SIR^{TTL} computed by SIR in comparison to the optimal offline solution OPT^{TTL} for the lunch scenario and the ratio between them.

TTL lunch						
			SIR $c = 2 \cdot$ Cap		SIF_L $c = 2$	
m	Cap	OPT_{TTL}	SIR_{TTL}	$\frac{SIR_{TTL}}{OPT_{TTL}}$	SIF_L^{TTL}	$\frac{SIF_L^{TTL}}{OPT_{TTL}}$
5	1	93.75	112.5	1.2	100	1.07
5	5	25	75	3	25	1
5	10	25	68.75	2.75	25	1
20	1	337.5	418.75	1.24	400	1.19
20	5	75	206.25	2.75	81.25	1.08
20	10	43.75	206.25	4.71	56.25	1.29
200	1	2818.75	4300	1.53	2968.75	1.05
200	5562.5		2125	3.78	600	1.07
200	10	202.6	550.45	5.5	468.75	1.7

with one VIPA with a capacity 5 but SIR could not. One VIPA operating on a circuit using SIF_L algorithm may serve up to 20 requests within the time horizon $[0, 125]$.

In Table 4.7,4.10 SIF_M respectively SIF_E lead to better ratios than SIR during the morning respectively evening. In Tables 4.8 and 4.11, w.r.t. minimizing the total waiting time, SIF_M leads to better ratios than SIR during the morning. SIF_E leads to better ratios than SIR during the evening.

Table 4.5: This table shows the computational results of the algorithms SIR respectively SIF_L w.r.t. the makespan for the lunch scenario, in comparison to the value of the corresponding optimal offline solution OPT^{MS} and OPT^{TWT} and the ratio between them. The competitive ratio c is shown.

			SIR lunch, MS, LB=2		SIF_L lunch, MS, LB=2	
m	Cap	OPT^{MS}	SIR^{MS}	$\frac{SIR^{MS}}{OPT^{MS}}$	SIF_L^{MS}	$\frac{SIF_L^{MS}}{OPT^{MS}}$
One VIPA						
5	1	93.75	112.5	1.20	112.5	1.2
5	5	67.8	88.4	1.30	75.7	1.11
5	10	67.8	88.4	1.30	75.5	1.11
20	1	-	-	-	-	-
20	5	100.43	-	-	120.8	1.25
20	10	85.6	121.78	1.42	104.5	1.22
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPAs						
5	1	93.75	112.5	1.20	112.5	1.20
5	5	67.8	88.4	1.30	75.7	1.12
5	10	67.8	88.4	1.30	75.5	1.11
20	1	-	-	-	-	-
20	5	89.23	-	-	109.4	1.23
20	10	85.6	121.78	1.42	104.5	1.22
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Five VIPAs						
5	1	93.75	112.5	1.20	112.5	1.20
5	5	67.8	74.5	1.10	75.5	1.11
5	10	67.8	74.5	1.10	75.5	1.11
20	1	-	-	-	-	-
20	5	85.6	106.7	1.25	104.5	1.22
20	10	85.6	106.7	1.25	104.5	1.22
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	×	120.5	×	113.5	×

Table 4.6: This table shows the computational results of the algorithms SIR respectively SIF_L for the lunch scenario, in comparison to the value of the corresponding optimal offline solution OPT^{TWT} and the ratio between them. The competitive ratio c is unbounded for both algorithms.

			SIR lunch, TWT, $c = \infty$		SIF_L lunch, TWT, $c = \infty$	
m	Cap	OPT^{TWT}	SIR^{TWT}	$\frac{SIR^{TWT}}{OPT^{TWT}}$	SIF_L^{TWT}	$\frac{SIF_L^{TWT}}{OPT^{TWT}}$
One VIPA						
5	1	88.6	92.5	1.04	92.5	1.04
5	5	23.5	36.8	1.57	42.5	1.81
5	10	23.5	36.8	1.57	42.5	1.81
20	1	-	-	-	-	-
20	5	95.2	-	-	-	-
20	10	85.6	105.25	1.23	428.25	5.00
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPAs						
5	1	76.4	92.5	1.21	92.5	1.21
5	5	18.7	25.2	1.35	28.5	1.52
5	10	18.7	25.2	1.35	28.5	1.52
20	1	-	-	-	-	-
20	5	84.6	-	-	223.5	2.64
20	10	82	85.25	1.04	428.25	5.22
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Five VIPAs						
5	1	76.4	92.5	1.21	92.5	1.21
5	5	18.7	25.2	1.35	28.5	1.52
5	10	18.7	25.2	1.35	28.5	1.52
20	1	-	-	-	-	-
20	5	73.23	85.25	1.16	152.3	2.08
20	10	73.23	85.25	1.16	428.25	5.85
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	×	1654.6	×	1042.5	×

Table 4.7: This table shows the computational results of the algorithms SIR_M respectively SIF_M w.r.t. the makespan for the morning scenario, in comparison to the value of the corresponding optimal offline solution OPT^{MS} and the ratio between them. A lower bound for the competitive ratio c is shown.

			SIR morning, MS, LB=2		SIF_M morning, MS, LB= $\frac{3}{2}$	
m	Cap	OPT^{MS}	SIR_M^{MS}	$\frac{SIR_M^{MS}}{OPT^{MS}}$	SIF_M^{MS}	$\frac{SIF_M^{MS}}{OPT^{MS}}$
One VIPA						
5	1	125	125	1.00	125	1.00
5	5	105	119	1.13	105	1.00
5	10	105	119	1.13	105	1.00
20	1	-	-	-	-	-
20	5	100	115.5	1.16	110.6	1.10
20	10	98.4	106.25	1.08	110.6	1.12
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPAs						
5	1	105	125	1.19	105	1.00
5	5	105	112.5	1.07	105	1.00
5	10	105	112.5	1.07	105	1.00
20	1	-	-	-	-	-
20	5	95.4	115.5	1.21	110.6	1.16
20	10	95.4	106.25	1.11	95.4	1.00
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Five VIPAs						
5	1	105	125	1.19	105	1.00
5	5	105	119	1.13	105	1.00
5	10	105	119	1.13	105	1.00
20	1	-	-	-	-	-
20	5	95.4	115.5	1.21	95.4	1.00
20	10	95.4	106.25	1.11	95.4	1.00
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	96.23	124.1	1.29	118.6	1.23

Table 4.8: This table shows the computational results of the algorithms SIR_M respectively SIF_M w.r.t. minimizing the total waiting time for the morning scenario, in comparison to the value of the corresponding optimal offline solution OPT^{TWT} and the ratio between them. The competitive ratio c is unbounded.

			SIR morning, TWT, $c = \infty$		SIF morning, TWT, $c = \infty$	
m	Cap	OPT^{TWT}	SIR_M^{TWT}	$\frac{SIR_M^{TWT}}{OPT^{TWT}}$	SIF_M^{TWT}	$\frac{SIF_M^{TWT}}{OPT^{TWT}}$
One VIPA						
5	1	101.2	101.2	1.00	101.2	1.00
5	5	28.4	36.2	1.27	28.4	1.00
5	10	28.4	36.2	1.27	28.4	1.00
20	1	-	-	-	-	-
20	5	145.6	258	2.02	145.6	1.13
20	10	152.6	175.4	2.63	152.6	2.28
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPA						
5	1	101.2	101.2	1.00	101.2	1.00
5	5	28.4	36.2	1.27	28.4	1.00
5	10	28.4	36.2	1.27	28.4	1.00
20	1	-	-	-	-	-
20	5	98.2	235.4	2.40	137.6	1.40
20	10	66.8	201.56	3.02	197.5	2.96
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Five VIPA						
5	1	101.2	101.2	1.00	101.2	1.00
5	5	28.4	36.2	1.27	28.4	1.00
5	10	28.4	36.2	1.27	28.4	1.00
20	1	-	-	-	-	-
20	5	98.2	235.4	2.40	137.6	1.40
20	10	66.8	235.4	3.52	197.5	2.95
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	×	335.8	×	165.28	×

Table 4.9: This table shows the values of the total tour length obtained by SIR for the morning and evening scenarios, in comparison to the value of the corresponding optimal offline solution OPT^{TTL} and the ratio between them. The competitive ratio c is shown for each of the scenarios, it is always greater than $\frac{SIR^{TTL}}{OPT^{TTL}}$, unless when the capacity of the VIPA is equal to 1, the ratio $\frac{SIR^{TTL}}{OPT^{TTL}}$ is equal to competitive ratio c.

TTL							
		SIR morning, TTL, $c = $ Cap			SIR Evening, TTL, $c = $ Cap		
m	Cap	SIR^{TTL}	OPT^{TTL}	$\frac{SIR^{TTL}}{OPT_{TTL}}$	SIR^{TTL}	OPT^{TTL}	$\frac{SIR^{TTL}}{OPT^{TTL}}$
5	1	125	125	1.00	125	125	1.00
5	5	86.67	25	3.47	100	25	4.00
5	10	86.67	25	3.47	81.25	25	3.25
20	1	500	500	1.00	500	500	1.00
20	5	125	100	1.25	113.3	100	1.13
20	10	106.25	50	2.13	110.4	50	2.21
200	1	5000	5000	1.00	5000	5000	1.00
200	5	1270.8	1000	1.27	2881.25	1000	2.88
200	10	585.5	500	1.17	575.45	401.5	1.43

Table 4.10: This table shows the makespan of the algorithms SIR and SIF_E for the evening scenario, in comparison to the value of the corresponding optimal offline solution OPT^{MS} and the ratio between them. A lower bound of the competitive ratio c is shown.

			SIR evening, MS, LB=2		SIF_E, evening, MS, LB=2	
One VIPA						
m	Cap	OPT^{MS}	SIR^{MS}	$\frac{SIR^{MS}}{OPT^{MS}}$	SIF_E^{MS}	$\frac{SIF_E^{MS}}{OPT^{MS}}$
5	1	125	125	1.00	125	1.00
5	5	87.8	106	1.21	102.2	1.16
5	10	87.8	106	1.21	102.2	1.16
20	1	-	-	-	-	-
20	5	103.5	124.6	1.20	121.5	1.17
20	10	103.5	121.2	1.17	108.5	1.04
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPAs						
			SIR evening, MS, LB=2		SIF_E, evening, MS, LB=2	
m	Cap	OPT^{MS}	SIR^{MS}	$\frac{SIR^{MS}}{OPT^{MS}}$	SIF_E^{MS}	$\frac{SIF_E^{MS}}{OPT^{MS}}$
5	1	125	125	1.00	125	1.00
5	5	87.8	104.2	1.19	102.2	1.16
5	10	87.8	104.2	1.19	102.2	1.16
20	1	-	-	-	-	-
20	5	75.2	124.6	1.66	116.6	1.55
20	10	75.2	120.5	1.60	116.6	1.55
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Five VIPAs						
			SIR evening, MS, LB=2		SIF_E, evening, MS, LB=2	
m	Cap	OPT^{MS}	SIR^{MS}	$\frac{SIR^{MS}}{OPT^{MS}}$	SIF_E^{MS}	$\frac{SIF_E^{MS}}{OPT^{MS}}$
5	1	125	125	1.00	125	1.00
5	5	87.8	90.2	1.03	95.6	1.08
5	10	87.8	90.2	1.03	95.6	1.08
20	1	-	-	-	-	-
20	5	75.2	92.6	1.23	87.6	1.16
20	10	75.2	92.6	1.23	87.6	1.16
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	×	118.26	×	102.46	×

Table 4.11: This table shows the total waiting time of the algorithms SIR and SIF_E for the evening scenario, in comparison to the value of the corresponding optimal offline solution OPT^{TWT} and the ratio between them. The competitive ratio c is unbounded.

			SIR (evening)		SIF_E (evening)	
m	Cap	OPT^{TWT}	SIR^{TWT}	$\frac{SIR^{TWT}}{OPT^{TWT}}$	SIF_E^{TWT}	$\frac{SIF_E^{TWT}}{OPT^{TWT}}$
One VIPA						
5	1	88.1	120.3	1.37	120.3	1.37
5	5	22.3	52.25	2.34	60.4	2.71
5	10	22.3	52.25	2.34	60.4	2.71
20	1	-	-	-	-	-
20	5	-	-	-	257.8	-
20	10	78.2	201.5	2.58	168.4	2.15
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	-	-
Two VIPAs						
			SIR (evening)		SIF (evening)	
m	Cap	OPT^{TWT}	SIR^{TWT}	$\frac{SIR^{TWT}}{OPT^{TWT}}$	SIF_E^{TWT}	$\frac{SIF_E^{TWT}}{OPT^{TWT}}$
5	1	88.1	120.3	1.37	120.3	1.37
5	5	22.3	52.25	2.34	60.4	2.71
5	10	22.3	52.25	2.34	60.4	2.71
20	1	-	-	-	-	-
20	5	78.2	302.7	3.87	212.5	2.72
20	10	78.2	250.4	3.20	168.4	2.15
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	-	-	-	328.6	×
Five VIPAs						
			SIR (evening)		SIF (evening)	
m	Cap	OPT^{TWT}	SIR^{TWT}	$\frac{SIR^{TWT}}{OPT^{TWT}}$	SIF_E^{TWT}	$\frac{SIF_E^{TWT}}{OPT^{TWT}}$
5	1	88.1	120.3	1.37	120.3	1.36
5	5	22.3	52.25	2.34	60.4	2.71
5	10	22.3	52.25	2.34	60.4	2.71
20	1	-	-	-	-	-
20	5	-	-	-	-	-
20	10	73.2	302.7	4.14	252.23	3.44
200	1	-	-	-	-	-
200	5	-	-	-	-	-
200	10	×	468.5	×	328.6	×

Chapter 5

Elevator Mode Problem

In this chapter we treat the PDP related to the elevator mode of the VIPA. 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 v_0 (say, the "left" end).

The input for the Online or Offline Elevator Mode Problem $(M, \sigma, T, 1, \text{Cap})$ consists of the following data:

- a weighted graph $G = (V, E, d)$, a bi-directed path, where the nodes correspond to stations, edges to their links, and edge weights $d : E \to \mathbb{R}_+$ determine the driving times between two nodes $v_i, v_j \in V$ with respect to the distance d corresponding to the length of a shortest path from v_i to v_j.
- a sequence $\sigma = \{r_1, \ldots, r_h\}$ of pdp-requests[1] $r_j = (t_j, x_j, y_j, z_j)$ with $z_j \leq \text{Cap}$,
- a time horizon $[0, T]$,
- one VIPA with a capacity Cap as the maximum number of passengers which can be simultaneously transported.

The output of the Online or Offline Elevator Mode Problem is a feasible transportation schedule S serving all requests in σ, consisting of one tour Γ for the VIPA.

The goal is to construct a transportation schedule $S = \{\Gamma\}$ w.r.t minimizing one of the following objective functions: total tour length, makespan, total waiting time, and total number of stops. As there is only one VIPA operating on the line, the objective functions are updated accordingly.

5.1 An Online Agorithm

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. MRIN has been analyzed w.r.t. minimizing the makespan [29],

[1] In this chapter, the term "request" means pdp-request.

giving a competitive ratio of 3/2. We generalize MRIN to the Pickup and Delivery Problem to solve the Online Elevator Mode Problem. In fact, 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 that consists of moving the VIPA away from the origin of the line as long as there are yet unserved requests in the same direction of the VIPA moves. If there are no more unserved requests in the same direction, then the VIPA changes direction and moves towards the origin of the line (see Algorithm 5). Then we, analyze it w.r.t. different objective functions in the next sections.

Algorithm 5 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 $s \leq x_j \leq y_j$
    if $\sigma'_{up} \neq \varnothing$ **then**
        Serve all requests (or up to Cap passengers) in σ'_{up} (moving away from v_O to furthest destination y_k among all $r_j \in \sigma'_{up}$)
    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
    update s and σ' (remove all served requests, add all newly released requests)

Important step in the MAIN algorithm When executing the step of "serving all requests (or up to Cap passengers) in σ'_{up} or σ'_{down}", in Algorithm 5, we need to handle the case when there are more than Cap passengers waiting to be served. Therefore we propose the following.

In case $\sum_{r_j \in \sigma'_{up}} z_j >$ Cap, then

- sort σ'_{up} by multiple levels of criteria.
  - the up requests of σ'_{up} are sorted in decreasing order of their destinations the request with the furthest delivery node from the origin v_0 of the line is the first one
  - a second criteria for ordering is then applied on the subset of requests having the same delivery node, that is an increasing order of their origins, the request with the nearest pickup node to the origin v_0 of the line is the first one;

- plan a new subtour for the server starting from the origin v_0 of the line, respecting the above canonical ordering of the requests:
  - take the first request of the ordered list σ'_{up}
  - check if, when inserting it to the subtour, the load of the VIPA on all up arcs (v_i, v_{i+1}) of the line will be less or equal than Cap
  - if the capacity of the VIPA is not violated, then insert the request to the subtour and remove the request from σ'_{up}
  - else stop.
  - take the last request of the ordered list σ'_{up}
  - check if, when inserting it to the subtour, the load of the VIPA on all up arcs (v_i, v_{i+1}) of the line will be less or equal than Cap
  - if the capacity of the VIPA is not violated, then insert the request to the subtour and remove the request from σ'_{up}
  - else stop.

In case $\sum_{r_j \in \sigma'_{down}} z_j >$ Cap, then

- sort σ'_{down} by multiple levels of criteria:
  - the down requests of σ'_{down} are sorted in increasing order of their destinations, the request with the nearest delivery node to the origin v_0 of the line is the first one.
  - a second criteria for ordering is then applied on the subset of requests having the same delivery node, that is a decreasing order of their origins, the request with the furthest pickup node from the origin v_0 of the line is the first one.
- plan a new subtour for the server starting from the origin v_0 of the line, respecting the above canonical ordering of the requests:
  - take the first request of the ordered list σ'_{down}
  - check if, when inserting it to the subtour, the load of the VIPA on all down arcs (v_i, v_{i+1}) of the line will be less or equal than Cap.
  - if the capacity of the VIPA is not violated, then insert the request to the subtour and remove the request from σ'_{down}
  - else stop.
  - take the last request of the ordered list σ'_{down}
  - check if, when inserting it to the subtour, the load of the VIPA on all down arcs (v_i, v_{i-1}) of the line will be less or equal than Cap.
  - if the capacity of the VIPA is not violated, then insert the request to the subtour and remove the request from σ'_{down}
  - else stop.

5.2 Minimizing the Total Tour Length

5.2.1 Optimal Offline solution for the EMP w.r.t. Minimizing the Total Tour Length

In order to obtain the optimal offline solution $OPT(\sigma)$ w.r.t. minimizing the total tour length, we compute a min cost flow in a suitable network. Given a line $L = (v_0, \ldots, v_\ell)$ with origin v_0 as a subnetwork, one VIPA of capacity Cap, and a request sequence σ with pdp-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 G_E, 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:
  - $load(v_i, v_{i+1}) = \sum_{(v_i, v_{i+1}) \in (x_j, y_j), x_j < y_j} z_j \ \forall i \in \{0, \ell - 1\}, \forall r_j \in U$
  - $load(v_{i+1}, v_i) = \sum_{(v_{i+1}, v_i) \in (x_j, y_j), x_j > y_j} z_j \ \forall i \in \{0, \ell - 1\}, \forall r_j \in D$
- 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 $m_{(i,i+1)} = \lceil \frac{load(v_i, v_{i+1})}{\text{Cap}} \rceil$ times and each arc (v_{i+1}, v_i) must be visited $m_{(i+1,i)} = \lceil \frac{load(v_{i+1}, v_i))}{\text{Cap}} \rceil$ times. In case the multiplicity $m_{(i,i+1)}$ resp. $m_{(i+1,i)}$ is equal to zero, the corresponding up resp. down arc is removed.

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

- the node set $V_E = V^{up(o)} \cup V^{up(d)} \cup V^{down(o)} \cup V^{down(d)} \cup \{s, t\}$ contains
  - the origin nodes of all up arcs with non-zero multiplicity in $V^{up(o)}$,
  - the destination nodes of all up arcs with non-zero multiplicity in $V^{up(d)}$,
  - the origin nodes of all down arcs with non-zero multiplicity in $V^{down(o)}$,
  - the destination nodes of all down arcs with non-zero multiplicity in $V^{down(d)}$,
  - 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 $v_i^{up(o)} \in V^{up(o)}$ and all $v_i^{down(o)} \in V^{down(o)}$ in A_s,
- up arcs $(v_i^{up(o)}, v_{i+1}^{up(d)})$ whenever $m_{(i,i+1)} \neq 0$ in A_U,
- down arcs $(v_{i+1}^{down(o)}, v_i^{down(d)})$ whenever $m_{(i+1,i)} \neq 0$ in A_D,
- link arcs in A_L going from all $v_i^{up(d)} \in V^{up(d)}$ to all $v_i^{down(o)} \in V^{down(o)}$, and from all $v_i^{down(d)} \in V^{down(d)}$ to all $v_i^{up(o)} \in V^{up(o)}$,
- sink arcs from all $v_i^{up(d)} \in V^{up(d)}$ and from all $v_i^{down(d)} \in V^{down(d)}$ to the sink t in A_t.

Note that G_E contains directed cycles. 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 expressed in driving times. 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 (5.1e). We finally add the constraint (5.1d) 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:

$$\min \sum_{a \in A_E} d(a) f(a) \tag{5.1a}$$

$$s.t. \sum_{a \in \delta^-(s)} f(a) = 1 \tag{5.1b}$$

$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a) \quad \forall v \neq \{s, t\} \tag{5.1c}$$

$$\sum_{a \in \delta(W)} f(a) \geq 2 \quad \forall W \subset V_E \setminus \{s, t\}, 2 \leq |W| \leq |V_E| - 3 \tag{5.1d}$$

$$f(a) = m(a) \quad \forall a \in \{A_U \cup A_D\} \tag{5.1e}$$

$$f(a) \in \mathbb{Z}_+ \tag{5.1f}$$

where $\delta^-(v, t)$ 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 (5.1d) 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 5.1. The family of constraints (5.1d) 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 (5.1d). Then we check if there is an isolated cycle in the solution obtained, if yes, we add only the constraint (5.1d) 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 [2]. 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 (5.1) corresponds to a transportation schedule with minimal total tour length for the offline problem behind the elevator mode.

Example 5.2. Consider a line $L = (v_0, \ldots, v_\ell)$ with origin v_0, a unit distance between v_i and v_{i+1} for each i, with a set σ of 9 pdp-requests shown in Figure 5.1, and a VIPA with capacity Cap = 2. The resulting network $G_E = (V_E, A_E)$ of the line presented in this example is illustrated in Figure 5.2, and the solution computed by the LP (5.1) is shown in Figure 5.3.

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 $G_E = (V_E, A_E)$.

◊

5.2.2 Competitive Analysis

Example 5.3. Consider a line $L = (v_0, \ldots, v_\ell)$ 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. Recall that, competitive results, (ratios) against an oblivious adversary are the strongest, in the analysis of competitive ratios we prove that the algorithms are $c-$ competitive against the oblivious adversary. The adversary releases a sequence $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of uniform pdp-requests that force MAIN to leave the origin of the line and to 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 $2\,|L|$:

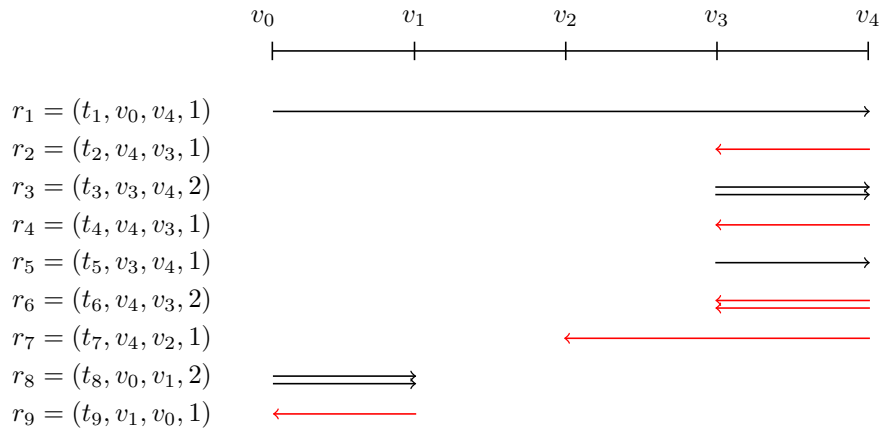
The first block σ_1 of $\ell \cdot \text{Cap}$ pdp-requests, consists of the following requests:

$r_1 = (0, v_0, v_1, 1)$

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



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

Figure 5.1: This figure illustrates the line $L = (v_0, \ldots, v_\ell)$ with origin v_0, and a set σ of 9 pdp-requests and a VIPA with capacity $\mathrm{Cap} = 2$. The requests are partitioned into two subsets "up-requests" (arcs in black) and "down-requests" (arcs in red). 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.

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

The second block σ_2 consists of the following $2\ell' \cdot \mathrm{Cap}$ pdp-requests:
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1)$ for $\ell\mathrm{Cap} + 1 \leq j \leq (\ell+1)\mathrm{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_{\ell-1}, v_\ell, 1)$ for $(\ell+1)\mathrm{Cap} + 1 \leq j \leq (\ell+2)\mathrm{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1)$ for $(\ell+2\ell'-2)\mathrm{Cap} + 1 \leq j \leq (\ell+2\ell'-1)\mathrm{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_{\ell-1}, v_\ell, 1)$ for $(\ell+2\ell'-1)\mathrm{Cap} + 1 \leq j \leq (\ell+2\ell')\mathrm{Cap}$
The third block σ_3 consists of the following $\ell \cdot \mathrm{Cap}$ pdp-requests:
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1)$ for $(\ell+2\ell'+1)\mathrm{Cap} + 1 \leq j \leq (\ell+2\ell'+2)\mathrm{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_{\ell-1}, v_{\ell-2}, 1)$ for $j = (\ell+2\ell'+2)\mathrm{Cap} + 1$
$r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1)$ for $(\ell+2\ell'+2)\mathrm{Cap} + 2 \leq j \leq (\ell+2\ell'+3)\mathrm{Cap}$
$r_j = (t_{j-1} + 2d(v_0, v_2), v_1, v_0, 1)$ for $j = (2\ell+2\ell'-1)\mathrm{Cap} + 1$

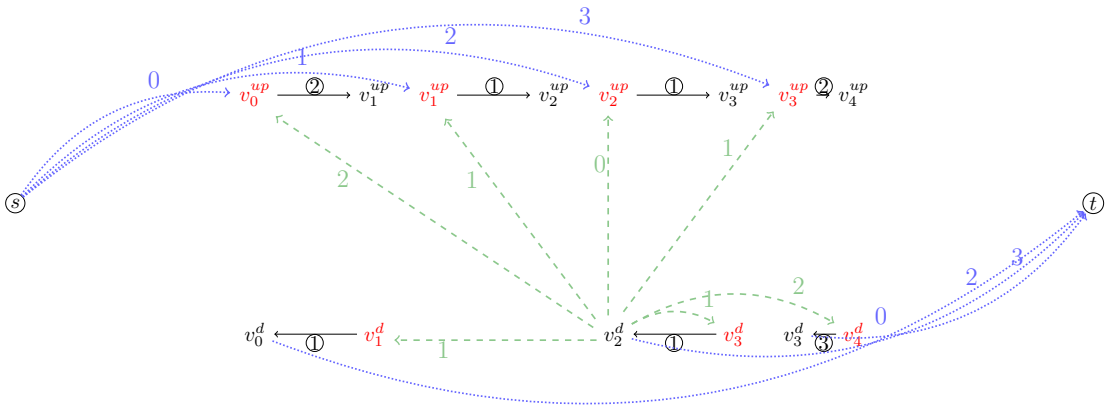


Figure 5.2: This figure illustrates the network $G_E = (V_E, A_E)$ of Example 5.2. The up and down arcs are illustrated by continuous arcs. Source and sink arcs are illustrated by dotted arcs. To ease the readability, we show a subset of the link arcs (dashed in the figure). The values within circles correspond to the value of the multiplicity $m(a)$ of the up or down-arc a in G_E. The values on the source, sink or link arcs correspond to the distance $d(a)$ of the arc a.

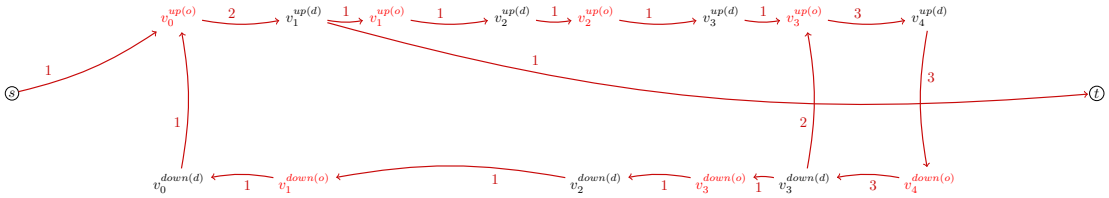


Figure 5.3: This figure illustrates the flow computed by the presented integer linear program for a min-cost flow problem in the network $G_E = (V_E, A_E)$ of Example 5.2. The values above the arcs correspond to the value of the flow $f(a)$ or the number of times the VIPA traverses the arc a in the transportation schedule.

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

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 pdp-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 $\ell \cdot \text{Cap}$ requests yielding

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

Then at $t = t_{\ell \mathrm{Cap}+1}$ MAIN starts to serve $r_{\ell \mathrm{Cap}+1}$ from v_n to v_{n-1} and performs a subtour of length $2d(v_0, v_\ell) = 2\,|L|$. When the VIPA operated by MAIN is back to the origin v_0, the request $r_{\ell \mathrm{Cap}+2}$ 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 $\ell' \cdot 2\mathrm{Cap}$ requests yielding

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

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

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

Therefore

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

The adversary waits at the origin v_0 until $t = t_{\mathrm{Cap}}$ and serves all requests $r_1, \ldots, r_{\mathrm{Cap}}$ from v_0 to v_1. Then he waits until $t = t_{2\mathrm{Cap}}$ 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} until the adversary arrives to v_ℓ. OPT served the first block of $\ell \cdot \mathrm{Cap}$ requests with a total tour length equal to $|L|$. Then the adversary begins to oscillate his VIPA between v_ℓ and $v_{\ell-1}$ and serves each time Cap requests, this is repeated $2\ell'$ times leading to a total tour length for σ_2 equal to $2\ell'$. Finally the 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 $OPT(\sigma) = 2\,|L| + 2\ell'$. Therefore, we obtain

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

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

as 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 5.4. *MAIN is $2Cap \cdot |L|$-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 pdp-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

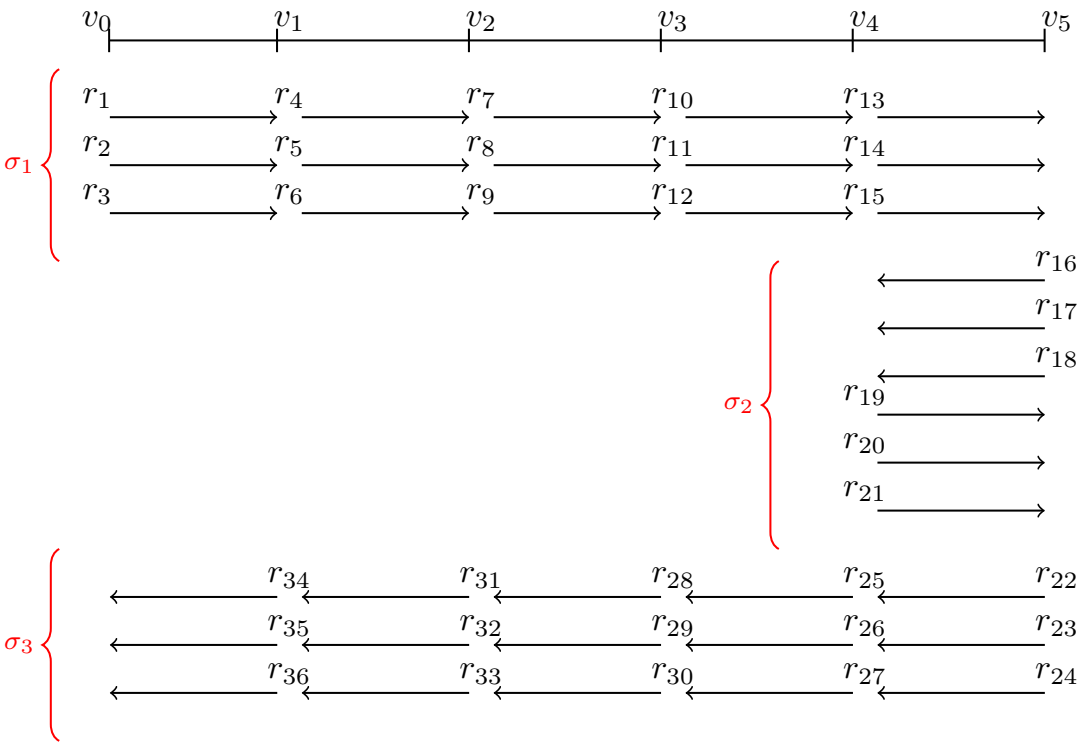


Figure 5.4: This figure illustrates the set $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3$ of pdp-requests (arcs under the line $L = (v_0, \ldots, v_5)$ with origin v_0) from Example 5.3 for Cap $= 3, \ell = 5$ and $\ell' = 1$.

- 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_ℓ.

For that, v_ℓ 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 5.3, with ℓ' blocks each consisting of

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

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

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

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

$$OPT(\sigma) = |L| \cdot 2\ell' \cdot |L|\,.$$

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 5.3 with

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

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 5.3) that

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

This finally leads to

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

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

as the maximum possible ratio between $MAIN(\sigma)$ and $OPT(\sigma)$ 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 5.3 shows that $2 \cdot \mathrm{Cap}$ is a lower bound on the competitive ratio of MAIN for the lunch scenario. 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 5.3 shows that Cap is a lower bound on the competitive ratio of MAIN for the morning respectively the evening scenario. We can show that these examples are the worst cases for MAIN during lunch, morning and evening:

Theorem 5.5. *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 pdp-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 pdp-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 pdp-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

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

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

is the maximum possible ratio between $MAIN(\sigma)$ and $OPT(\sigma)$ 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

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

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

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

□

5.3 Minimizing the Makespan

5.3.1 Optimal Offline Solution for the EMP w.r.t. Minimizing the Makespan

In Chapter 4 Section 4.3.1, we presented an adapted model for the Tram Mode Problem w.r.t. minimizing the makespan based on a formulation for the PDP with time windows proposed in [44].

In case of the Elevator Mode Problem, minimizing the makespan turns out to be the same as minimizing the total tour length with the waiting time of the VIPA, as only one VIPA is used on a line. Accordingly, we adapt the model used in Section 4.3.1 in Chapter 4. In the following we mention only the differences: Given a line $L = (v_0, \ldots, v_n)$, with origin v_0 as a network $L = (V, E)$, a VIPA with capacity Cap, and a request sequence σ with n pdp-requests $r_j = (t_j, x_j, y_j, z_j)$ such that $z_j \leq \text{Cap}$, otherwise the request is split.
We construct the complete directed graph $L_M = (V_M, A_M)$ in a similar way as C_D for the offline solution of the Tram Mode Problem w.r.t. minimizing the makespan. The main differences are in decision variables of the integer program proposed. Instead of using k different decision variables for the routing, the times, and the load of the VIPAs, we use only one for each, because in the Elevator Mode Problem, we have $k = 1$. Therefore, the routing decisions are represented by the variables below.

- The time at which the VIPA begins service at node $v \in V_M$, is denoted by the decision variable $B(v)$.
- The load of the VIPA after visiting node $v \in V_M$ is denoted by $Q(v)$.
- the trace of the VIPA is reflected by

$$f(v,v') = \begin{cases} 1, & \text{if the VIPA travels along the arc } a = (v,v') \\ 0, & \text{otherwise} \end{cases}$$

Finally, with each arc $(v,v') \in A_M$ we associate a cost $c(v,v')$ and a driving time $d(v,v')$ corresponding to the distance of the shortest path from v to v' in the original line $L = (V,E)$.

The objective function in (4.2) for the Tram Mode Problem is to minimize T_{max}, where T_{max} is greater or equal than the time at which the last VIPA arrives at the depot $\max_{i\in\{1\cdots k\}} B^i(t)$. As we are considering only one VIPA, we can replace this objective function by $B(t)$ (5.3a), and obtain the same result.

The corresponding integer program is detailed in (5.3)

$$\begin{aligned}
&\min\ B(t) && && (5.3a)\\
&s.t. \sum_{a\in\delta^-(x_j)} f(a) = 1, && \forall x_j \in V_x && (5.3b)\\
&\sum_{a\in\delta^-(s)} f(a) = 1 && && (5.3c)\\
&\sum_{a\in\delta^-(v)} f(a) = \sum_{a\in\delta^+(v)} f(a) && \forall v \in V_x \cup V_y && (5.3d)\\
&\sum_{a\in\delta^+(t)} f(a) = 1 && && (5.3e)\\
&B(v') \geq (B((v) + d(v,v'))f(v,v') && \forall v,v' \in V_M && (5.3f)\\
&Q(v') \geq (Q(v) + q(v))f(v,v') && \forall v,v' \in V_M && (5.3g)\\
&B(y_j) - B(x_j) - d(x_j,y_j) \geq 0 && \forall r_j \in \sigma && (5.3h)\\
&B(t) - B(s) \leq T && && (5.3i)\\
&B(x_j) \geq t_j && \forall x_j \in V_x && (5.3j)\\
&\max\{0,q(v)\} \leq Q(v) \leq \min\{\text{Cap}, \text{Cap} + q(v)\} && \forall, v \in V_M && (5.3k)\\
&f(a) \in \{0,1\} && \forall, a \in A_M && (5.3l)\\
&B(v) \geq 0 && \forall, v \in V_M && (5.3m)\\
&Q(v) \geq 0 && \forall, v \in V_M && (5.3n)
\end{aligned}$$

Constraints (5.3b) ensure that each request is served exactly once. Constraints (5.3c),(5.3d) and (5.3e) guarantee that a tour starts at the origin v_0 of the circuit C. Consistency of the time and load variables is ensured by constraints (5.3f) and (5.3g). Precedence constraints are imposed through inequalities (5.3h). Finally, inequalities (5.3i) bound the duration of the tour (the tour of the VIPA cannot exceed the time horizon T). Constraints (5.3j) ensure that a request cannot be served before it is released and (5.3k) impose capacity constraints.

Similarly to the formulation of the offline solution in Tram Mode Problem w.r.t. minimizing the makespan, this formulation is also non linear because of constraints

(5.3f) and (5.3g). By introducing constants $M(v, v')$ and $W(v, v')$, these constraints can be linearized as follows:

$$B(v') \geq B((v) + d(v, v') - M(v, v')(1 - f(v, v')) \qquad \forall v, v' \in V_M \qquad (5.4a)$$
$$Q(v') \geq Q(v) + q(v) - W(v, v')(1 - f(v, v')) \qquad \forall v, v' \in V_M \qquad (5.4b)$$

In Figure 5.5, we show the computed flow of the Example 5.2 by solving the integer linear program (5.3). The computed flow and routing decisions yield an optimal transportation schedule for the Offline Elevator Mode Problem w.r.t. minimizing the makespan.

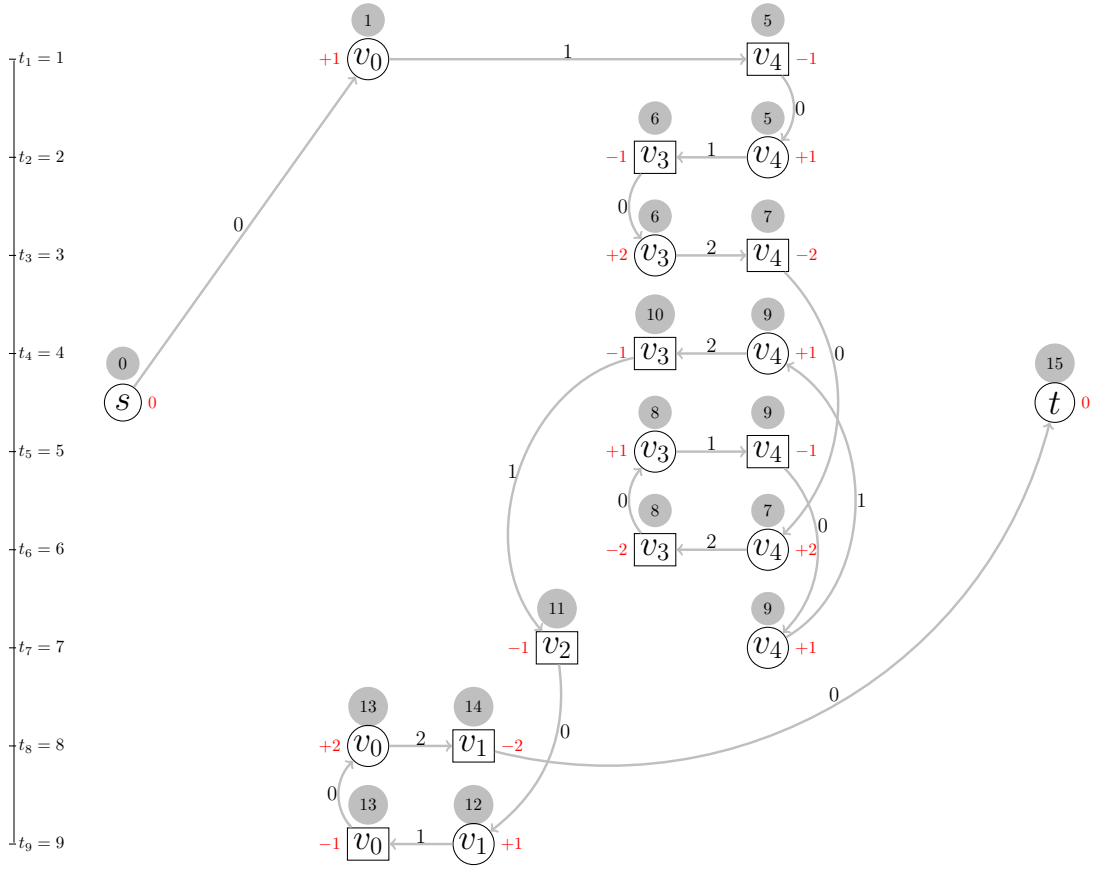


Figure 5.5: This figure illustrates the resulting flow computed by the LP (4.2) on the network $C_M = (V_M, A_M)$ of Example 5.2. The makespan has a value of 15. The origins of the requests are represented by circles, the destinations by square. the values of $B_1(v)$ on the nodes are illustrated by a value enclosed by a gray circle above the nodes The arcs are labeled by the values of $Q(v)$ (the load of the VIPA after visiting a node v. The values next to the nodes v correspond to the load of the $q(v)$, $(q(s) = q(t) = 0)$. the nodes of the requests are placed in the figure vertically by their increasing order of release dates.

We construct the tours from the resulting computed routing decisions in the same way as in TramMP (see Section 4.3.1). The resulting optimal transportation schedule of the Example 5.2 w.r.t. minimizing the makespan is the following.

$$\Gamma^1 = (v_0,0) \xrightarrow{r_1} (v_0,1) \xrightarrow{r_2} (v_4,5) \xrightarrow{r_3} (v_3,6) \xrightarrow{r_6} (v_4,7) \xrightarrow{r_5} (v_3,8) \xrightarrow{r_7} (v_4,9) \xrightarrow{r_4} (v_3,10) \to (v_2,11) \to (v_1,12) \xrightarrow{r_9} (v_0,13) \xrightarrow{r_8} (v_1,14) \to (v_0,15)$$

5.3.2 Competitive Analysis

The adversary can again "cheat" the strategy of MAIN, as the following example shows. (Again in these competitive analysis we use the oblivious adversary.)

Example 5.6. 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. The adversary releases a sequence $\sigma = (r_1, r_2)$ with 2 pdp-requests

$$r_1 = (0, v_0, v_n, 1) \text{ and } r_2 = (\varepsilon, v_0, v_n, 1).$$

MAIN determines at time $t = 0$ the set $\sigma' = \{r_1\}$ and serves r_1 by moving from v_0 to v_n. At time $t = n$, we have $s = v_n$ and $\sigma' = \varnothing$, so it moves back to v_0, arriving at time $t = 2n$. Now, $s = v_0$ and $\sigma' = \{r_2\}$, so MAIN starts its VIPA again to serve r_2 by moving from v_0 to v_n, and finally returns to v_0 at time $t = 4n$. The adversary waits in v_0 until $t = \varepsilon$ and serves both requests r_1 and r_2 by moving from v_0 to v_n, and returns to v_0 at time $t = 2n + \varepsilon$. Therefore, we obtain

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{4n}{\varepsilon + 2n} = 2$$

as a lower bound for the competitive ratio of MAIN w.r.t. minimizing the makespan, even in the morning scenario. ◊

By slightly modifying the worst case example of the morning scenario, one can obtain a worst case example for the evening scenario w.r.t. minimizing the makespan.

Example 5.7. 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. The adversary releases a sequence $\sigma = (r_1, r_2)$ with 2 requests

$$r_1 = (0, v_n, v_0, 1) \text{ and } r_2 = (n + \varepsilon, v_n, v_0, 1).$$

MAIN determines at time $t = 0$ the set $\sigma' = \{r_1\}$ and moves from v_0 to v_n and starts serving r_1 at time $t = n$ and arrives at the origin v_0 at $t = 2n$. But the adversary releases a new request $r_2 = (n + \varepsilon, v_n, v_0, 1)$. Now, $s = v_0$ and $\sigma' = \{r_2\}$, so MAIN starts its

VIPA again to serve r_2 by moving from v_0 to v_n, and finally returns to v_0 at time $t = 4n$. The adversary waits in v_0 until $t = \varepsilon$ moves from v_0 to v_n, and serves both requests r_1 and r_2, then arrives to v_0 at time $t = 2n + \varepsilon$. Therefore, we obtain

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{4n}{\varepsilon + 2n} = 2$$

as a lower bound for the competitive ratio of MAIN w.r.t. minimizing the makespan, even in the evening scenario. ◊

We conjecture that this bound is tight in general and can ensure it for the morning and evening scenario:

Theorem 5.8. *MAIN is 2-competitive for the Online Elevator Mode Problem w.r.t minimizing the makespan for one VIPA operating in elevator mode on a line during the morning and evening scenarios, with uniform requests, i.e., all requests have a load of 1.*

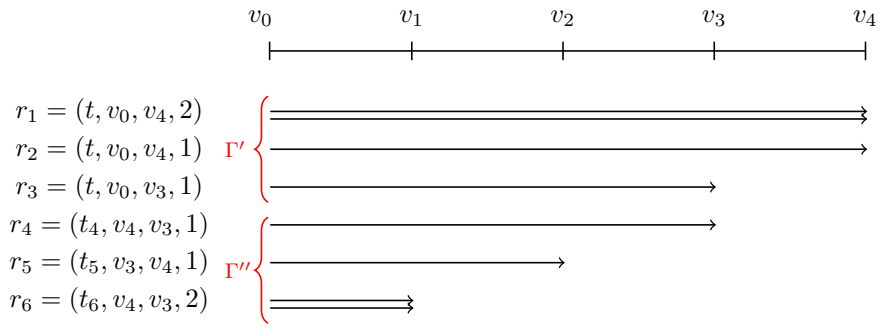
Before proving the theorem, we provide the following lemma.

Lemma 5.9. *MAIN computes an optimal transportation schedule for the Elevator Mode Problem w.r.t. minimizing the makespan if the whole sequence σ is known in advance, i.e. if all release dates are zero, during the morning respectively the evening.*

Proof. First, note that minimizing the makespan amounts to minimizing the total tour length taking into account the waiting time of the VIPA because we are using only one VIPA on the line. The objective of this lemma is to prove that MAIN provides the optimal solution w.r.t. minimizing the makespan if the whole sequence σ is known in advance, i.e. if all release dates are zero, during the morning respectively the evening. Therefore this lemma will provide the proof that MAIN solves optimally the classical DARP or PDP with one server operating on a line as a metric space, in case all requests have the same origin or all requests have the same destination. During the morning respectively the evening, all requests have the same pickup respectively delivery node that is the origin v_0 of the line. If $\sum_{r_j \in \sigma} z_j \leq$ Cap, then MAIN serves all the requests in σ together and $MAIN(\sigma) = OPT(\sigma)$. If $\sum_{r_j \in \sigma} z_j >$ Cap, MAIN proceeds as follows. MAIN sorts the requests in σ in the decreasing order of their destinations respectively in the increasing order of their origins during the morning respectively the evening, as Figures 5.6 and 5.7 show. Then MAIN serves progressively Cap passengers each time he leaves the origin respecting the order of sorting. For example, in the examples shown in Figures 5.6 and 5.7, MAIN first serves r_1, r_2 and r_3 in a subtour Γ' and then serves r_4, r_5 and r_6 in the subtour Γ''.

In order to express the value of the makespan or the total tour length with the waiting time of MAIN we may rely on the model presented in Section 5.2.1 used to solve the optimal offline solution w.r.t the total tour length. Therefore we proceed as follows.

- Partition the requests into two subsets:



$load(uparcs)$	8	6	5	3
$load(downarcs)$	0	0	0	0
$m(uparcs) = \lceil \frac{load}{\text{Cap}} \rceil$	2	2	2	1
$m(downarcs) = \lceil \frac{load}{\text{Cap}} \rceil$	0	0	0	0

Figure 5.6: This figure illustrates the line $L = (v_0, \ldots, v_\ell)$ with origin v_0, and a set σ of 6 requests and a VIPA with capacity $\text{Cap} = 4$. There is only one partition of requests, "up-requests", since we are in the morning scenario (illustrated by the arcs). Each arc represents a load of 1, for example r_1 is represented by 2 arcs going from v_0 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 arcs.

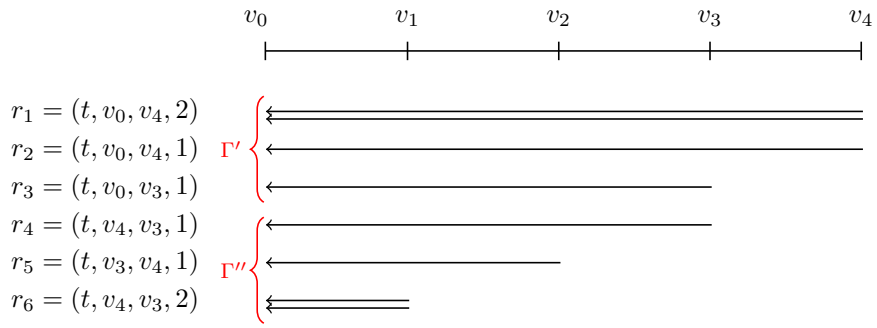
  - U of "up-requests" $r_j \in \sigma$ with $x_j < y_j$,
  - D of "down-requests" $r_j \in \sigma$ with $x_j > y_j$.

- 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:

  - $load(v_i, v_{i+1}) = \sum_{(v_i, v_{i+1}) \in (x_j, y_j), x_j < y_j} z_j \ \forall i \in \{0, \ell - 1\}, \forall r_j \in U$
  - $load(v_{i+1}, v_i) = \sum_{(v_{i+1}, v_i) \in (x_j, y_j), x_j > y_j} z_j \ \forall i \in \{0, \ell - 1\}, \forall r_j \in D$

- 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 $m_{(i,i+1)} = \lceil \frac{load(v_i, v_{i+1})}{\text{Cap}} \rceil$ times and each arc (v_{i+1}, v_i) must be visited $m_{(i+1,i)} = \lceil \frac{load(v_{i+1}, v_i))}{\text{Cap}} \rceil$ times.

Let $m_i = \max\{m(v_{i-1}, v_i), m(v_i, vi - 1)\}$, then we may express the makespan of MAIN



$load(uparcs)$	0	0	0	0
$load(downarcs)$	8	6	5	3
$m(uparcs) = \lceil \frac{load}{\text{Cap}} \rceil$	0	0	0	0
$m(downarcs) = \lceil \frac{load}{\text{Cap}} \rceil$	2	2	2	1

Figure 5.7: This figure illustrates the line $L = (v_0, \ldots, v_\ell)$ with origin v_0, and a set σ of 6 requests and a VIPA with capacity $\text{Cap} = 4$. There is only one partition of requests, "down-requests", since we are in the evening scenario (illustrated by the arcs). Each arc represents a load of 1, for example r_1 is represented by 2 arcs going from v_4 to v_0. 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.

during the morning and evening scenario by

$$
\begin{aligned}
MAIN(\sigma) =& m_n \cdot 2d(v_0, v_n) + \sum_{\sigma_i \in \sigma, 1 \le i \le n-1} (m_i - m_{i+1}) \cdot 2d(v_0, v_i) \quad \text{s.t. } m_i - m_{i+1} \ge 0 \\
=& m_n \cdot 2d(v_0, v_n) + (m_{n-1} - m_n) \cdot 2d(v_0, v_{n-1}) + \cdots + (m_1 - m_2) \cdot 2d(v_0, v_1) \\
=& m_n \cdot 2(d(v_0, v_1) + d(v_1, v_2) + \cdots + d(v_{n-1}, v_n)) + (m_{n-1}) \cdot 2(d(v_1, v_2) + \cdots \\
& + d(v_{n-2}, v_{n-1})) - m_n \cdot 2(d(v_1, v_2) + \cdots + d(v_{n-2}, v_{n-1})) + \cdots + \\
& m_1 \cdot 2d(v_0, v_1) - m_2 \cdot 2d(v_0, v_1) \\
=& m_n \cdot 2d(v_{n-1}, v_n) + \cdots + m_2 \cdot 2d(v_1, v_2) + m_1 \cdot 2d(v_0, v_1)
\end{aligned}
$$

As for OPT, in order to express its value of the makespan, we may also rely on the model presented in Section 5.2.1 used to solve the optimal offline solution w.r.t the total tour length. Actually we can use this model to compute a transportation schedule w.r.t. minimizing the makespan only if all release dates are the same, i.e. $t_j = 0$ for all requests $r_j \in \sigma$, and not in the offline situation. The multiplicity computed for each arc enables us to precise exactly how many times the VIPA must traverse each of the up and down arcs in order to serve all requests. In general, the total tour length of OPT serving the

sequence σ is

$$OPT(\sigma) = \sum_{1 \leq i \leq n} \max\{m(v_i, v_{i+1}), m(v_{i+1}, v_i)\} \cdot 2d(v_i, v_{i+1}).$$

Let $m_i = \max\{m(v_{i-1}, v_i), m(v_i, vi-1)\}$, then

$$\begin{aligned} OPT(\sigma) = &\sum_{1 \leq i \leq n} m_i \cdot 2d(v_i, v_{i+1}) 1 \leq i \leq n \\ &= m_1 \cdot 2d(v_0, v_1) + m_2 \cdot 2d(v_1, v_2) + \cdots + m_n \cdot 2d(v_{n-1}, v_n). \end{aligned}$$

Therefore $MAIN(\sigma) = OPT(\sigma)$ for the Elevator Mode Problem w.r.t. minimizing the makespan if the whole sequence σ is known in advance, all release dates are zero, during the morning respectively the evening. □

Now we can prove the Theorem 5.8.

Proof. We show the theorem by induction on the number of requests in the set $\sigma = r_1, ...r_{m-1}, r_m$ of pdp-requests $r_j = (t_j, x_j, y_j, 1)$. It clearly holds if σ contains at most one request. Assume now that the claim of the theorem holds for any sequence of $m-1$ requests and prove it for m requests. Suppose that request $r_m = (t_m, x_m, y_m, z_m)$ is the last request of $\sigma = r_1, ..., r_{m-1}, r_m$. If $t = t_m = 0$, from Lemma 5.9 we may deduce that $MAIN(\sigma) = OPT(\sigma)$ and MAIN is 2-competitive. Thus, the capacity would not enter the analysis in the following proof of competitive ratio w.r.t. minimizing the makespan during the morning or the evening. Assume that $t > 0$, let $pos(t)$ and $pos^*(t)$ be the positions of MAIN and the adversary server at time t, respectively. Note that, as the metric space $M = (V, d)$ is a line, if the server of MAIN for example is at the origin then $pos(t) = 0$ and each of the positions reflect the distance from the origin to the corresponding station ($d(v_0, x_j) = x_j$ and so on).

Let $a = \max_{r_j \in \sigma \setminus \{r_m\}}\{x_j, y_j\}$ be the pickup or the delivery node among all pickup and delivery nodes of the requests r_j (excluding r_m), unserved by the MAIN server at time t, which is furthest away from the origin of the line. If all requests in $\sigma = r_1, ..., r_{m-1}$ have already been served by MAIN at time t, then we set $a = 0$. Let a_m be the origin or the destination of the request r_m which is furthest away from the origin, i.e. if $x_m < y_m$ then $a_m = y_m$, else $a_m = x_m$.

We start by proving the case when $pos(t) = pos^*(t) = v_0$ (both VIPAs are located at the origin of the line at time t). Afterwards, we analyze all the cases of the general situation when MAIN and the adversary have different positions.

(1) Case $pos(t) = pos^*(t) = v_0$:

- If MAIN did not serve yet r_j but the adversary has already served it, then $t \geq 2d(v_0, a) = 2a$, $OPT(\sigma) = t + 2a_m$ and $MAIN(\sigma) = t + 2 \cdot Max(a, a_m)$.

If $a \leq a_m$, then $MAIN(\sigma) = OPT(\sigma) = t + 2a_m$ and MAIN is obviously 2-competitive.
If $a \geq a_m$, then $MAIN(\sigma) = t + 2a$.

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{t+2a}{t+2a_m} = \frac{t+2a_m}{t+2a_m} + \frac{2(a-a_m)}{t+2a_m} \leq 1 + \frac{2(a-a_m)}{t+2a_m}$$

$$2a - 2a_m \leq 2a \leq t \leq t + 2a_m \Rightarrow \frac{2(a-a_m)}{t+2a_m} \leq 1$$

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{t+2a}{t+2a_m} \leq 2$$

- If MAIN did not serve yet r_j and neither did the adversary, then

$$OPT(\sigma) = MAIN(\sigma) = t + 2 \cdot Max(a, a_m)$$

and MAIN is obviously 2-competitive.

(2) Case $pos(t) = pos^*(t) \neq v_0$ or $pos(t) \neq pos^*(t)$:
At time t, we can have many options for the request r_j:

(i) MAIN did not serve yet r_j, but the adversary has already served it.

$$OPT(\sigma) = t + pos^*(t) + 2a_m \tag{5.5}$$
$$MAIN(\sigma) = t + pos(t) + 2 \cdot Max(a, a_m) \tag{5.6}$$

(ii) MAIN did not serve yet r_j, neither did the adversary

$$OPT(\sigma) = t + pos^*(t) + 2 \cdot Max(a, a_m) \tag{5.7}$$
$$MAIN(\sigma) = t + pos(t) + 2 \cdot Max(a, a_m) \tag{5.8}$$

(iii) MAIN did not serve yet r_j, the adversary already started to serve r_j.

If the adversary already started to serve r_j, one of the following possibilities may occur.

(a) The user is not yet delivered to a, but already picked up from v_0 during the morning.
(b) The user is not yet delivered to v_0, neither picked up from a during the evening.
(c) The user is not yet delivered to v_0, but already picked up from a during the evening.

We obtain , for option (a) and (b),
$OPT(\sigma) = t + d(pos^*(t), a) + a + 2a_m = t + |a - pos^*(t)| + a + 2a_m$ (5.9)
for option (c), $OPT(\sigma) = t + pos^*(t) + 2a_m$ (5.10)
therefore, $OPT(\sigma) \geq t + |a - pos^*(t)| + a + 2a_m \geq t + pos^*(t) + 2a_m$ (5.11)
$MAIN(\sigma) = t + pos(t) + 2 \cdot Max(a, a_m)$ (5.12)

(iv) MAIN and the adversary already started to serve r_j. Therefore one, of the following possibilities may occur.

(a) In both algorithms the user is not yet delivered to a, but already picked up from v_0 during the morning.

(b) In both algorithms the user is not yet delivered to v_0, neither picked up from a during the evening.

(c) In both algorithms the user is not yet delivered to v_0, but already picked up from a during the evening.

(d) In MAIN, the user is not yet delivered to v_0, neither picked up from a and the adversary did not yet delivered the user to v_0, but already picked him up from a during the evening.

We obtain for option (a) and (b),

$$OPT(\sigma) = t + d(pos^*(t), a) + a + 2a_m = t + |a - pos^*(t)| + a + 2a_m \quad (5.13)$$

for option (c) and (d), $OPT(\sigma) = t + pos^*(t) + 2a_m$ (5.14)

therefore, $OPT(\sigma) \geq t + |a - pos^*(t)| + a + 2a_m \geq t + pos^*(t) + 2a_m$ (5.15)

for option (a), (b) and (d),

$$MAIN(\sigma) = t + d(pos(t), a) + a + 2a_m = t + |a - pos(t)| + a + 2a_m \quad (5.16)$$

for option (c), $MAIN(\sigma) = t + pos(t) + 2a_m$ (5.17)

therefore, $MAIN(\sigma) \leq t + pos(t) + 2a_m \leq t + |a - pos(t)| + a + 2a_m$ (5.18)

(v) MAIN already started to serve r_j and the adversary has already served it. If MAIN already started to serve r_j, one of the following possibilities may occur.

(a) The user is not yet delivered to a, but already picked up from v_0 during the morning.

(b) The user is not yet delivered to v_0, neither picked up from a during the evening.

(c) The user is not yet delivered to v_0, but already picked up from a during the evening.

Therefore we obtain

$OPT(\sigma) = t + pos^*(t) + 2a_m$ and (5.19)

For option (a) and (b), $MAIN(\sigma) = t + |a - pos(t)| + a + 2a_m$ (5.20)

For option (c), $MAIN(\sigma) = t + pos(t) + 2a_m$ (5.21)

therefore, $MAIN(\sigma) \leq t + pos(t) + 2a_m \leq t + |a - pos(t)| + a + 2a_m$ (5.22)

We can notice that:

$$\begin{aligned} OPT(\sigma) &\geq t + pos^*(t) + 2 \cdot Max(a, a_m) && \text{by (5.7)} \\ &\geq t + pos^*(t) + 2a_m && \text{by (5.5), (5.11), (5.15) and (5.19)} \\ MAIN(\sigma) &\leq t + pos(t) + 2 \cdot Max(a, a_m) && \text{by (5.6), (5.8) and (5.12)} \\ &\leq t + |a - pos(t)| + a + 2a_m && \text{by (5.18) and (5.22)} \end{aligned}$$

Therefore without loss of generality we can prove the theorem by using

$$OPT(\sigma) \geq t + pos^*(t) + 2a_m \geq t + 2a_m \text{ and } MAIN(\sigma) \leq t + |a - pos(t)| + a + 2a_m$$

Where MAIN has already started to serve r_j and the adversary has already served it, thus $t \geq 2a$.

$$\frac{MAIN(\sigma)}{OPT(\sigma)} \leq \frac{t + |a - pos(t)| + a + 2a_m}{t + 2a_m} \leq 1 + \frac{|a - pos(t)| + a}{t + 2a_m}$$

$$|a - pos(t)| + a \leq t \leq t + 2a_m \Rightarrow \frac{2a - pos(t)}{t + 2a_m} \leq 1$$

$$\text{and } \frac{MAIN(\sigma)}{OPT(\sigma)} \leq 2$$

and MAIN is 2-competitive.

□

We conjecture that this competitive ratio hold even if the requests are not uniform.

Conjecture 2. MAIN is 2-competitive for the Online Elevator Mode Problem w.r.t minimizing the makespan for one VIPA operating in elevator mode on a line during the morning and evening scenarios.

5.4 Minimizing the Total Waiting Time

5.4.1 Optimal Offline Solution for the EMP w.r.t. Minimizing the Total Waiting Time

In Chapter 4 Section 4.4.1, we presented an integer linear program to solve the Offline Tram Mode Problem w.r.t. minimizing the total waiting time, based on a formulation for the PDP with time windows proposed in [44]. In case of the Elevator Mode Problem, we use the same model for computing the optimal offline solution w.r.t. minimizing the total waiting time. In Figure 5.8, we show the computed flow of the Example 5.2 by solving the integer linear program (4.3) of Section 4.4.1 in Chapter 4. The computed flow yields an optimal transportation schedule for the Offline Elevator Mode Problem w.r.t. minimizing the total waiting time.

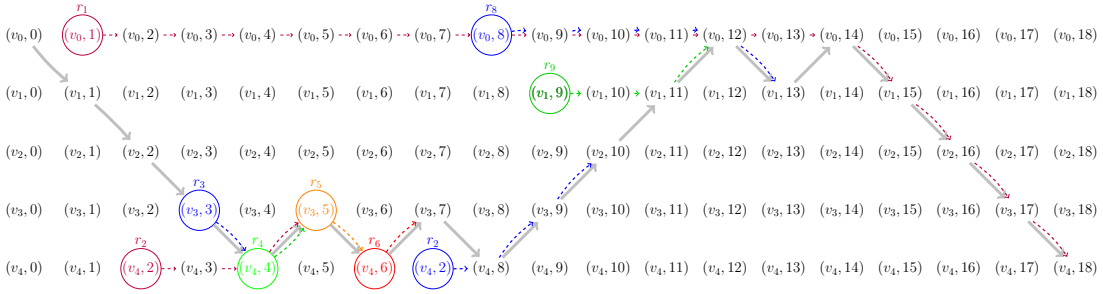


Figure 5.8: This figure illustrates the time-expanded network $L_T = (V_T, A_T)$ of Example 5.2. It shows the flow computed by the integer linear program (4.3) in L_T. The resulting flow has a value of 26 as a total waiting time. We identify the route of the VIPA from the arcs a with $F(a) > 0$ represented by solid arcs. Each request r_j corresponds to a commodity f_j and the nodes enclosed by circles illustrate the corresponding commodity's origins. For each request, the corresponding arcs a, with $f_j(a) > 0$, are indicated by dashed arcs with a different color. To ease the readability, the end of time horizon is not presented, thus there are missing nodes from $(v, 19)$ to $(v, 22)$, where $v \in V$. The arcs having positive flow containing nodes (v, t) with $19 \leq t \leq 22$ are also not illustrated. This includes the sink arcs a having a positive flow $f_j(a) > 0$ of the different commodities, and the arcs a with $F(a) > 0$ that represent the way back of the VIPA from $(v_4, 18)$ to the sink $(v_0, 22)$ (the origin of the line).

5.4.2 Competitive Analysis

Concerning the objective of minimizing the total waiting time, it turns out that MAIN is not competitive. In this example, the oblivious and the non-abusive adversaries behave the same way.

Example 5.10. Consider a line $L = (v_0, \ldots, v_n)$ with origin v_0. The adversary releases a sequence $\sigma = (r_1, r_2)$ with only 2 requests

$$r_1 = (0, v_0, v_n, 1) \text{ and } r_2 = (\varepsilon, v_0, v_n, \text{Cap} - 1).$$

MAIN starts its tour at $t = 0$, serves r_1 by moving from v_0 to v_n and is back to v_0 at time $t = 2|L|$ s.t. each passenger of r_2 has waited for $2|L|$, yielding

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

The adversary waits at the origin v_0 until $t = \varepsilon$, starts its VIPA to serve both requests r_1 and r_2 at once, yielding

$$OPT(\sigma) = \varepsilon.$$

Therefore we obtain

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{2|L| \cdot (\text{Cap} - 1)}{\varepsilon} \to \infty$$

(when choosing ε arbitrarily small) which implies that $MAIN$ is not competitive w.r.t. minimizing the total waiting time TWT. ◊

5.5 Minimizing the Total Number of Stops

5.5.1 Optimal Offline Solution for the EMP w.r.t. Minimizing the Total Number of Stops

Similarly to Chapter 4 Section 4.5, in order to compute the optimal offline solution w.r.t. minimizing the total number of stops we need to

- wait until time t_m in the origin v_0 (to ensure that all requests are released before they are served),
- compute the optimal offline solution w.r.t. minimizing the total number of stops assuming that the VIPA has a unit capacity, by solving an LP similar to the one proposed in Section 5.2.1 w.r.t. the total tour length. In this case, the objective function considers costs $d(a) = d(u, v)$ for the flow f only on link arcs $a = (u, v)$ in A_L, where $d(u, v)$ is the length of a shortest path from u to v in the line L.
- solve a partitioning problem to partition the paths obtained into subtours for the VIPA such that the total number of stops is minimized

5.5.2 Competitive Analysis

Concerning the objective of minimizing the number of stops, the oblivious adversary can cheat by accumulating all requests having the same origin and destination and serve them together.

Example 5.11. Consider a line $L = (v_0, \ldots, v_\ell)$ 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 $\sigma = (\sigma_1, \sigma_3)$ of the sequence of the Example 5.3 of $2\ell \cdot \text{Cap}$ uniform pdp-requests that force MAIN to serve each request using two stops, whereas the adversary is able to serve all requests using $2\ell + 1$ stops.

The first block σ_1 consists of the following $\ell \cdot \text{Cap}$ requests:

$$
\begin{aligned}
&r_1 = (0, v_0, v_1, 1)\\
&r_j = (t_{j-1} + 2d(v_0, v_1), v_0, v_1, 1) \text{ for } 2 \leq j \leq \text{Cap}\\
&r_j = (t_{j-1} + 2d(v_0, v_1), v_1, v_2, 1) \text{ for } j = \text{Cap} + 1\\
&r_j = (t_{j-1} + 2d(v_0, v_2), v_1, v_2, 1) \text{ for } \text{Cap} + 2 \leq j \leq 2\text{Cap}\\
&\quad\vdots\\
&r_j = (t_{j-1} + 2d(v_0, v_{\ell-1}), v_{\ell-1}, v_\ell, 1) \text{ for } j = (\ell - 1)\text{Cap} + 1\\
&r_j = (t_{j-1} + 2d(v_0, v_\ell), v_{\ell-1}, v_\ell, 1) \text{ for } (\ell - 1)\text{Cap} + 2 \leq j \leq \ell\text{Cap}
\end{aligned}
$$

The third block σ_3 consists of the following $\ell \cdot \text{Cap}$ requests:

$$
\begin{aligned}
&r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1) \text{ for } \ell\text{Cap} + 1 \leq j \leq (\ell+1)\text{Cap} \\
&r_j = (t_{j-1} + 2d(v_0, v_\ell), v_{\ell-1}, v_{\ell-2}, 1) \text{ for } j = (\ell+1)\text{Cap} + 1 \\
&r_j = (t_{j-1} + 2d(v_0, v_\ell), v_\ell, v_{\ell-1}, 1) \text{ for } (\ell+1)\text{Cap} + 2 \leq j \leq (\ell+2)\text{Cap} \\
&\quad \vdots \\
&r_j = (t_{j-1} + 2d(v_0, v_2), v_1, v_0, 1) \text{ for } j = (2\ell-1)\text{Cap} + 1 \\
&r_j = (t_{j-1} + 2d(v_0, v_1), v_1, v_0, 1) \text{ for } (2\ell-1)\text{Cap} + 2 \leq j \leq 2\ell\text{Cap}
\end{aligned}
$$

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 $2d(v_0, v_1) = 2$ and 2 stops 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 and 2 stops to serve r_2, without serving any further request in this subtour. This is repeated for each request until serving the first block of $\ell \cdot \text{Cap}$ requests yielding

$$MAIN(\sigma_1) = \text{Cap} \cdot \ell \cdot 2$$

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

$$MAIN(\sigma_3) = \text{Cap} \cdot \ell \cdot 2$$

Therefore

$$MAIN(\sigma) = 2\text{Cap} \cdot \ell \cdot 2$$

The adversary waits at the origin v_0 until $t = t_{\text{Cap}}$ and serves all requests $r_1, \ldots, r_{\text{Cap}}$ using 2 stops 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, using only one stop at v_2 for delivering the passengers. The pick up of the passengers of $r_{\text{Cap}+1}, \ldots, r_{2\text{Cap}}$ is done at v_1, at the same time of the delivery of the passengers of the first Cap requests. This is repeated for all Cap requests from v_i to v_{i+1} until the adversary arrives to v_ℓ. OPT served the first block of $\ell \cdot \text{Cap}$ requests with a total number of stops equal to $\ell + 1$ stops. Finally the 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, using ℓ stops, yielding $OPT(\sigma) = 2\ell + 1$. Therefore, we obtain

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{2\text{Cap} \cdot \ell \cdot 2}{2\ell + 1} \underset{\ell \to +\infty}{\longrightarrow} 2\text{Cap}$$

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 5.12. *MAIN is* $2\,Cap$*-competitive for the Online Elevator Mode Problem w.r.t minimizing the total number of stops.*

Proof. The worst transportation schedule results if all requests are uniform and the VIPA operated by MAIN performs a separate subtour using 2 stops 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 $2 \cdot |\sigma|$ as total number of stops.
To maximize the ratio between the total number of stops 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 subtours as possible in OPT.

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

- OPT picks up Cap passengers and delivers Cap passengers at each stop.
- MAIN is forced to use 2 stops per passenger.

This can be achieved with the subsequences σ_1 and σ_3 from Example 5.11 with

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

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 5.11) that

$$MAIN(\sigma_1) = MAIN(\sigma_3) = \text{Cap} \cdot \ell \cdot 2).$$

This finally leads to

$$\frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{2\text{Cap} \cdot \ell \cdot 2}{2\ell + 1} \underset{\ell \to +\infty}{\longrightarrow} 2\text{Cap}$$

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

Concerning the lunch scenario, 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 5.11 shows that $\frac{4}{3} \cdot \text{Cap}$ is a lower bound on the competitive ratio of MAIN. As for the morning resp. evening scenario, 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 5.11 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 5.13. *For one VIPA with capacity Cap operating in elevator mode on a line, MAIN is w.r.t. the objective of minimizing the total number of stops*

- $\frac{4}{3} \cdot$ *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 $2 \cdot |\sigma|$ as total number of stops. 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 number of stops 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 uses 2 stops to serve a request $r_j = (t_j, v_0, v_1, 1)$ or $r_j = (t_j, v_1, v_0, 1)$, OPT stops once at v_0 to pick up Cap passengers, then stops at v_1 to deliver Cap passengers of the first Cap requests, and and pick up Cap passengers. Finally MAIN stops at v_0 to deliver Cap passengers. Overall OPT uses 3 stops. 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 uses 2 stops 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 uses 2 stops but to serve Cap requests together.
- all requests in σ can be served with as few 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

    $$MAIN(\sigma) = 2 \cdot |\sigma| = 2 \cdot 2 \cdot \text{Cap and } OPT(\sigma) = 3$$

    $$\text{s.t. } \frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{2 \cdot 2 \cdot \text{Cap}}{3} = \frac{4}{3}\text{Cap}$$

    is the maximum possible ratio between $MAIN(\sigma)$ and $OPT(\sigma)$ 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

    $$MAIN(\sigma) = 2 \cdot |\sigma| = 2 \cdot \text{Cap and } OPT(\sigma) = 2$$

$$\text{s.t. } \frac{MAIN(\sigma)}{OPT(\sigma)} = \frac{2 \cdot \text{Cap}}{2} = \text{Cap}$$

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

□

5.6 Computational Results

This section deals with computational experiments for the MAIN algorithm w.r.t. different objective functions (total tour length, makespan and total waiting time). 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 algorithm in average. The computational results presented in Table 5.1 support this expectation. They compare the total tour length $MAIN^{TTL}$, the makespan $MAIN^{MS}$ and the total waiting time $MAIN^{TWT}$ computed by MAIN with the corresponding optimal offline solutions OPT^{TTL}, OPT^{MS} and OPT^{TWT}. 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 [112]. The computations are performed with the help of a simulation tool developed by Yan Zhao [123]. The instances use a line as subnetwork with 1 VIPA, 5-200 requests, represented by m in the table, 1-12 as the maximum load z_j of a request. For each period (morning, evening, lunch and general), for every parameter set, we created 50 test instances and compute them using 1 VIPA but varying its capacity 1-10. The instances are grouped by first the period of the time, then for each period, 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 algorithm MAIN has been implemented in Java. For solving the integer linear programs to get the optimal solutions w.r.t. the different objective functions we use Gurobi 8.21.

In Table 5.1, a hyphen '-' indicates that no solution has been found (due to the restricted time horizon). A cross '×' indicates that no optimal solution has been found by the ILP within four hours and therefore a cross '×' in the ratio column indicates that no ratio can be computed due to the absence of the optimal solution. In case the solution is preceded by a percentage, it means that only this percentage of instances give a solution and the others are infeasible.

In Table 5.1, we can notice the following:

- w.r.t. minimizing the total tour length, MAIN has (in average) good ratios, that never reach the competitive ratios.
- w.r.t. minimizing the makespan, MAIN has also (in average) good ratios. Note that in 15% of the instances with 200 requests, MAIN can serve all 200 requests within a time horizon of [0-160].

For the lunch scenario, for the instances with 200 requests with a VIPA of capacity 10, the total tour length is (in average 165) close to the $T = 160$. Therefore, MAIN would have served 200 requests if we slightly increase T during the lunch.

- w.r.t. minimizing the waiting time, the ratio between MAIN and OPT is small in general, except during the lunch, where OPT would move VIPAs towards the origins x_j of not yet released requests r_j (but also would respect the release time t_j to serve accepted requests r_j). Thus in the optimal offline solution the passengers wait less.

Table 5.1: This table shows the computational results for several test instances of the algorithm $MAIN$ w.r.t. different objective functions (total tour length, makespan and total waiting time) in comparison to the value of the respective optimal offline solutions, the time horizon is $[0, 160]$ and the length of the line is $|L| = 32$.

MAIN (Morning)										
		total tour length TTL $c = \text{Cap}$			Makespan MS $c = 2$			Total waiting time TWT $c = \infty$		
m	Cap	$MAIN^{TTL}$	OPT^{TTL}	$\frac{TTL}{OPT}$	$MAIN^{MS}$	OPT^{MS}	$\frac{MS}{OPT}$	$MAIN^{TWT}$	OPT^{TWT}	$\frac{TWT}{OPT}$
5	1	97.3	97.3	1	110.8	110.8	1	5.6	5.6	1
5	5	55.5	25.78	2.15	84.5	63.74	1.33	23.2	15.7	1.48
5	10	55.5	25.78	2.15	84.5	63.74	1.33	23.2	15.7	1.48
20	1	327.2	327.2	1	-	-	-	-	-	-
20	5	135.8	135.8	1	135.8	135.8	1	22.6	22.6	1
20	10	78.3	47.5	1.65	92.35	69.4	1.33	92.35	69.4	1.33
200	1	3394.8	3394.8	1	-	-	-	-	-	-
200	5	1050.5	694.8	1.51	-	-	-	-	-	-
200	10	760.8	362.34	2.1	-	-	-	-	-	-
MAIN (Evening)										
		total tour length TTL $c = \text{Cap}$			Makespan MS $c = 2$			Total waiting time TWT $c = \infty$		
m	Cap	$MAIN^{TTL}$	OPT^{TTL}	$\frac{TTL}{OPT}$	$MAIN^{MS}$	OPT^{MS}	$\frac{MS}{OPT}$	$MAIN^{TWT}$	OPT^{TWT}	$\frac{TWT}{OPT}$
5	1	85.5	57.3	1.49	117.67	75.5	1.56	25.2	16.9	1.49
5	5	55.5	25.78	2.15	68.6	45.2	1.52	29.6	13.4	2.21
5	10	55.5	25.78	2.15	68.6	45.2	1.52	29.6	13.4	2.21
20	1	365.2	327.2	1.12	-	-	-	-	-	-
20	5	135.8	73	1.86	145.6	86.5	1.68	92.6	31.2	2.97
20	10	108.3	47.5	2.28	124.6	69.6	1.79	92.35	28.5	3.24
200	1	3434.8	3394.8	1.01	-	-	-	-	-	-
200	5	1050.5	694.8	1.51	-	-	-	-	-	-
200	10	860.8	362.34	2.38	-	-	-	-	-	-
MAIN (Lunch)										
		total tour length TTL $c = 2 \cdot \text{Cap}$			Makespan MS $LB = 2$			Total waiting time TWT $c = \infty$		
m	Cap	$MAIN^{TTL}$	OPT^{TTL}	$\frac{TTL}{OPT}$	$MAIN^{MS}$	OPT^{MS}	$\frac{MS}{OPT}$	$MAIN^{TWT}$	OPT^{TWT}	$\frac{TWT}{OPT}$
5	1	61.15	29,5	2.07	89.5	78.15	1.15	42.8	8.9	4.81
5	5	74.76	37	2.02	84.5	57.6	1.47	26.5	1.02	25.98
5	10	74.76	37	2.02	84.5	57.6	1.47	26.5	1.02	25.98
20	1	222.35	177	1.26	-	-		-	-	-
20	5	78.4	67	1.17	90.8	79.2	1.15	86.2	17.6	4.9
20	10	75.58	34	2.22	84.6	61.6	1.37	83.6	16.4	5.1
200	1	4918.75	1855.5	2.65	-	-	-	-	-	-
200	5	2696.42	368.5	7.32	-	-	-	-	-	-
200	10	1166.25	165	7.06	-	-	-	-	-	-
MAIN (General)										
		total tour length TTL, $\lvert L\rvert = 32$ $c = 2 \cdot \text{Cap} \cdot \lvert L\rvert$			Makespan MS $LB = 2$			Total waiting time TWT $c = \infty$		
m	Cap	$MAIN^{TTL}$	OPT^{TTL}	$\frac{TTL}{OPT}$	$MAIN^{MS}$	OPT^{MS}	$\frac{MS}{OPT}$	$MAIN^{TWT}$	OPT^{TWT}	$\frac{TWT}{OPT}$
5	1	56.3	36	1.56	72.4	46.8	1.55	72.8	32.9	2.21
5	5	42.45	27	1.57	50.78	37.9	1.34	36.5	16.3	2.24
5	10	42.45	27	1.57	50.78	37.9	1.34	36.5	16.3	2.24
20	1	140.8	120,5	1.17	146.5	131.7	1.11	-	-	-
20	5	95.54	43	2.22	120.3	72.8	1.65	89.6	25.2	3.56
20	10	80.15	33,5	2.39	100.4	55.8	1.8	79.5	23.2	3.43
200	1	5023.25	1435,5	3.5	-	-	-	-	-	-
200	5	1662.4	292	5.69	-	-	-	-	-	-
200	10	180.8	120,5	1.5	(15%)154.5	126.6	1.22	(15%)152.3	×	×

Chapter 6

Taxi Mode Problem

In this chapter, we treat the PDP related to the taxi mode as the most advanced circulation mode for VIPAs in the dynamic fleet management system, the Taxi Mode Problem. The transport requests are released over time and need to be served within a specified time window. A request can be either accepted or rejected, but once accepted, it must be served, i.e., there must be a VIPA available at the associated stations. Hereby, we consider the user-friendly (and operator-unfriendly) situation, where the decision whether a request is accepted or rejected is taken at the moment the customer sends his request.

In the following, we consider the Taxi Mode Problem, where the objective is a hierarchical one; to maximize the number of accepted customer requests (primary) and to serve them at minimum costs (secondary).

This means, the operator decides which request can be accepted and which has to be rejected. We distinguish two main problems that arise from the Taxi Mode Problem and provide a solution approach for each of them:

- **Non-Preemptive Taxi Mode Problem,** where at each time, at most one customer can be transported by a VIPA (a customer can be a group of people less than the capacity of the VIPA and load preemption is not allowed), and a VIPA cannot serve other requests until the current one is delivered. Note that, due to the time windows and the additional technical restrictions of the VIPAs, it is not always possible to serve all transport requests. Hence, the studied PDP includes firstly to accept or reject requests and secondly to generate non-preemptive tours for the VIPAs to serve the accepted requests.

- **Preemptive Taxi Mode Problem,** where at each time, a VIPA may serve one or many requests with or without load preemption respecting the VIPAs capacity. This depends on the policy of the operator. It is not always possible to serve all transport requests. Hence, the studied PDP also includes firstly to accept or reject requests and secondly to generate preemptive tours (where VIPAs may exchange customers at a station) for the VIPAs to serve the accepted requests.

The input for the Online or Offline (Preemptive or Non-Preemptive) 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 full-requests[1] $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ 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 or 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. Thus, we treat 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. The tours differ depending on whether or not we require non-preemption.

In this chapter, we provide a solution approach for each of the studied problem variants, the Non-Preemptive and the Preemptive Taxi Mode Problems, by means of flows in time-expanded networks as, e.g., proposed by [63, 70, 90] 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 (as 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, compared to the optimal offline solution which is computed based on the entire request sequence already known.

In order to solve the offline versions of the Taxi Mode Problem, we consider a max-profit flow problem in suitable networks where moving the VIPAs in the system induces some small costs and a high profit is given for each accepted request. Due to the high profit on an accepted request, the solver emphasizes on serving these, while the small fee on moving the VIPAs ensures that VIPAs are not sent out unnecessarily. This results in more natural transportation schedules than if there are no costs for the VIPAs.

In Section 6.1.1, we present a method to compute optimal offline solutions for the Non-Preemptive Taxi Mode Problem. In Section 6.1.2, we propose a replan strategy for the Online Non-Preemptive Taxi Mode Problem that, in fact, solves the online problem by computing a sequence of offline subproblems on certain subsequences of requests.

In Section 6.2.1, we propose two methods to solve the Offline Preemptive Taxi Mode Problem optimally first using a path formulation and second using multicommodity

[1] In this chapter, the term "request" means full-request.

flow. Due to the long computation time of finding a good or an optimal solution in the Preemptive Taxi Mode Problem, we propose a flow-based heuristic (see Section 6.2.1.3). This flow-based heuristic computes a solution in two phases: (1) a preprocessing phase where we aim at refining the time-expanded network and (2) computing a transportation schedule on a reduced time-expanded network. In Section 6.2.2, we propose a "modified replan" strategy for the Online Preemptive Taxi Mode Problem that, in fact, solves the online problem by computing a sequence of "heuristic offline subproblems" on certain subsequences of requests.

Finally, we evaluate the performance of the proposed replan strategies in comparison with the optimal offline solution in theory (with the help of competitive analysis, in Section 6.3), and in practice as well (with the help of some computational results, in Section 6.4). We summarize the outline of this chapter in Table 6.1.

Table 6.1: Outline of Chapter 6

Non-Preemptive (NP-TaxiMP) Section 6.1	**Offline Section 6.1.1**	Exact method via Max-Profit Flow
	Online Section 6.1.2	Replan strategy REPLAN-NP
Preemptive (P-TaxiMP) Section 6.2	**Offline Section 6.2.1**	Exact method via a path formulation Section 6.2.1.1
		Exact method via a multicommodity coupled flow Section 6.2.1.2
		heuristic via a multicommodity flow FLOW-HEURISTIC Section 6.2.1.3
	Online Section 6.2.2	modified replan strategy hREPLAN-P
NP-TaxiMP and P-TaxiMP	Competitive Analysis Section 6.3	
	Computational results Section 6.4	

6.1 Solving the Non-Preemptive Taxi Mode Problem (NP- TaxiMP)

In this Section, we compute optimal offline solutions for the Non-Preemptive Taxi Mode Problem, then we propose a replan strategy for the Online Non-Preemptive Taxi Mode Problem that, in fact, solves the online problem by computing a sequence of offline subproblems on certain subsequences of requests.

6.1.1 Offline NP-TaxiMP via Max-Profit Flow

In order to solve the Offline Non-Preemptive Taxi Mode Problem, we build a time-expanded demand network $G_D = (V_D, A_D)$ based on σ and the original network G. The node set $V_D = (v_0, 0) \cup V_x \cup V_y \cup (v_0, T)$ is composed of

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

The arc set $A_D = A_+ \cup A_R \cup A_L \cup A_-$ 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_D is acyclic by construction.

The VIPAs shall form a flow F through this time-expanded demand network G_D. To correctly initialize the system, we use the node $(v_0, 0) \in V_D$ as source for the flow F and set its balance accordingly to the number k of available vehicles, see (6.1b). For all internal nodes $(v, t) \in V_D \setminus \{(v_0, 0), (v_0, T)\}$, we use normal flow conservation constraints, see (6.1c), which also automatically ensures that a flow of value k is entering the sink (v_0, T).

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 (6.1d). By the previous constraints, the flow F is (automatically) bounded on all other arcs by 1.

Note that source and flow conservation constraints (6.1b), (6.1c) together give rise to a totally unimodular matrix (the node-arc incidence matrix of the digraph underlying the network G_D), but due to the inequalities (6.1d) the entire constraint matrix is not totally unimodular s.t. integrality constraints (6.1f) are required (to prevent fractional solutions).

We consider a max-profit flow problem to decide which requests can be served without spending more costs than gaining profit by serving them. Accordingly, our objective function (6.1a) 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:

$$\max \sum_{a \in A_R} p(a)F(a) - \sum_{a \in A_D} c(a)F(a) \tag{6.1a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v_0,0)} F(a) = k \tag{6.1b}$$

$$\sum_{a \in \delta^-(v,t)} F(a) = \sum_{a \in \delta^+(v,t)} F(a) \qquad \forall (v,t) \neq (v_0,0),(v_0,T) \tag{6.1c}$$

$$\sum_{a \in A_R^j} F(a) \leq 1 \qquad \forall A_R^j \in A_R \tag{6.1d}$$

$$F(a) \geq 0 \qquad \forall a \in A_D \tag{6.1e}$$

$$F(a) \in \mathbf{Z} \qquad \forall a \in A_D \tag{6.1f}$$

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 integer linear program (6.1) solves the offline version of the Non-Preemptive Taxi Mode Problem (where the whole sequence σ of requests is known at time $t = 0$) to optimality.

Theorem 6.1. *The integer linear program (6.1) provides an optimal solution of the Offline Non-Preemptive 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 demand network G_D 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 $a \in A_D$ with $F^*(a) > 0$ by standard flow decomposition as in [2]. In our case, the correspondence between the flow in G_D and the moves in the VIPA tours Γ^i is particularly easy to see, because constraints (6.1d) together with the flow conservation constraints (6.1c) 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 $a \in A_D$. 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 $F^*(a) \in \{0, 1\}\ \forall a \in A_D$, 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, due to flow conservation, a' is such that, $F^*(a') = 1$. 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 $a \in A_D$ 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 Γ^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 (6.1a) guarantees that σ_A is maximum and that the tours $\Gamma^1, \ldots, \Gamma^k$ have indeed minimum total tour length. □

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 entire constraint matrix becomes totally unimodular which implies:

Corollary 6.2. *The Offline Non-Preemptive Taxi Mode Problem with tight time windows can be solved in polynomial time.*

In the general case, this is not true, but computational experiments show that the running times to solve the Offline Non-Preemptive Taxi Mode Problem are still reasonable (see Section 6.4).

Example 6.3. Consider an instance $(M, \sigma, p, 10, 2, 1)$ of the Offline Non-Preemptive TaxiMP with

- the network G with a depot v_0 with arcs having uniform distance, see Figure 6.1,
- two unit-speed servers (i.e. two VIPAs that travel 1 unit of length in 1 unit of time) with capacity Cap = 2 originally located at the depot v_0,
- the following sequence σ of 6 requests:

$$\begin{aligned} r_1 &= (0, a, c, 1, 4, 1) \quad & r_4 &= (3, b, f, 4, 7, 1) \quad & r_6 &= (5, b, c, 6, 8, 1) \\ r_2 &= (1, c, f, 5, 9, 1) \quad & r_5 &= (3, b, g, 4, 7, 1) \quad & r_7 &= (5, b, e, 6, 9, 1) \\ r_3 &= (1, e, g, 2, 5, 1) \end{aligned}$$

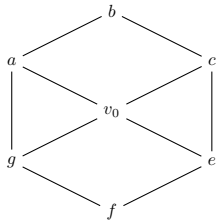


Figure 6.1: This figure illustrates the network G of the instance $(M, \sigma, p, 10, 2, 1)$ of the Offline Non-Preemptive TaxiMP from Example 6.3.

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

The optimal solution F^* in the resulting time-expanded demand network G_D is illustrated in Figure 6.2. We have $\sigma_A = \{r_1, r_2, r_3, r_4, r_6\}$ and the following tours for the two VIPAs:

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (b, 4) \xrightarrow{r_4} (f, 7) \to (v_0, 9)$$
$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (g, 4) \to (b, 6) \xrightarrow{r_6} (c, 7) \xrightarrow{r_2} (f, 9) \to (v_0, 9)$$

The total number of accepted requests is 5 served with a total tour length of 15 and an objective function value of 5.

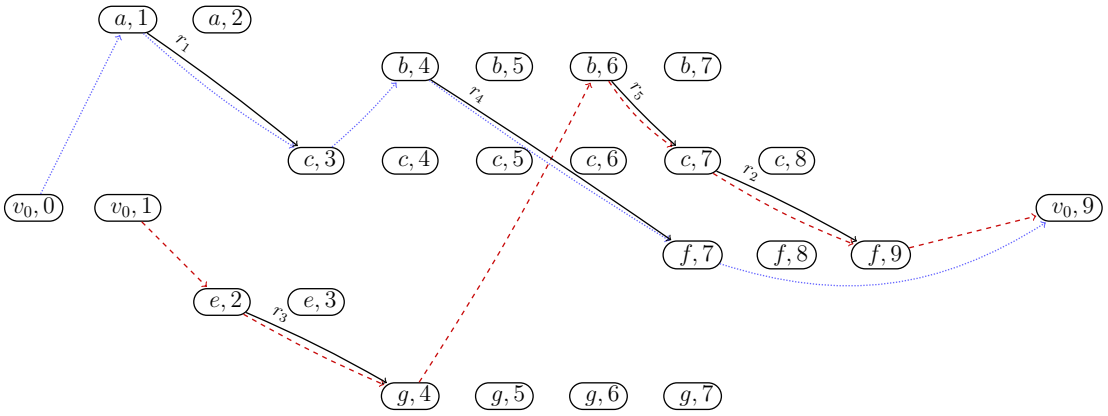


Figure 6.2: This figure shows the arcs with positive flow in the time-expanded demand network G_D for the instance $(M, \sigma, p, 10, 2, 1)$ of the Offline Non-Preemptive TaxiMP from Example 6.3. The computed flow F^* in the time-expanded demand network G_D can be interpreted as transportation schedule. The tour of the first VIPA is indicated by dashed arcs, and the tour of the second VIPA by dotted arcs. Solid arcs correspond to request arcs in A_R. The total number of accepted requests is 5 served with a total tour length of 15.

$\Diamond$

6.1.2 Online NP-TaxiMP via Replan

To handle the online situation (where the requests in σ are released over time during a time horizon $[0, T]$), we propose to solve a sequence of offline subproblems for certain time intervals $[t', T']$ within $[0, T]$ on accordingly modified demand networks.

Recall that the overall idea of a replan strategy is to

- consider at each time $t' \in [0, T]$ the subsequence $\sigma(t')$ of currently waiting requests (i.e., already released but not yet served requests), to determine which requests from $\sigma(t')$ can be accepted, and to compute optimal (partial) non-preemptive tours to serve them,
- 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 Algorithm 6 REPLAN-NP.

Algorithm 6 (REPLAN-NP)

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

Output: σ_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 $t' \leq T$ DO: call OFFLINE-NP(σ_A, $\sigma(t')$, $\Gamma^1, \ldots, \Gamma^k$)
perform the (modified) tours until new requests become known
update t' and $\sigma(t')$

3: return σ_A and $\Gamma^1, \ldots, \Gamma^k$

To compute those optimal solutions for the subsequences $\sigma(t')$, we build a time-expanded demand network $G(t') = (V', A')$ based on $\sigma(t')$ and the original network G and consider a flow in $G(t')$ that corresponds to the studied (partial) tours.

We construct $G(t') = (V', A')$ in a similar way as G_D 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 $\Gamma^1, \ldots, \Gamma^k$. 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 $P(t')_i = y_j$ and $S(t')_i = t_j^{drop}$; otherwise, $P(t')_i$ is the current position v of VIPA i and $S(t')_i = t'$.

We construct $G(t') = (V', A')$ as follows:
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 $1 \leq i \leq k$ as sources in V_+,

- all possible origins (x_j, t_j^{pick}) of all $r_j \in \sigma(t')$ and 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 $r_j \in \sigma(t')$ and all $p_j + d(x_j, y_j) \leq t_j^{drop} \leq q_j$ 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 $t' + d(v, x_j) \leq t_j^{pick}$ 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 $(y_j, t_j^{drop}) \in V_y$ 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 $r_j \in \sigma_A(t')$ respectively newly released request $r_j \in \sigma_N(t')$.

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

$$\max \sum_{a \in A_R} p(a) f'(a) - \sum_{a \in A'} c(a) f'(a) \tag{6.2a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v,t)} f'(a) = k(v) \qquad \forall (v,t) \in V_+ \tag{6.2b}$$

$$\sum_{a \in \delta^-(v,t)} f'(a) = \sum_{a \in \delta^+(v,t)} f'(a) \qquad \forall (v,t) \in V_x \cup V_y \tag{6.2c}$$

$$\sum_{a \in A^j_{R_A}} f'(a) = 1 \qquad \forall A^j_{R_A} \subseteq A_{R_A} \tag{6.2d}$$

$$\sum_{a \in A^j_{R_N}} f'(a) \leq 1 \qquad \forall A^j_{R_N} \subseteq A_{R_N} \tag{6.2e}$$

$$f'(a) \geq 0 \qquad \forall a \in A' \tag{6.2f}$$

$$f'(a) \in \mathbf{Z} \qquad \forall a \in A' \tag{6.2g}$$

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 (6.2d) ensure that previously accepted requests are served whereas constraints (6.2e) allow to reject newly released requests.

Source and flow conservation (6.2b), (6.2c) together give again rise to a totally unimodular matrix, but due to (6.2d) and (6.2e) the entire constraint matrix is not totally unimodular s.t. integrality constraints (6.2g) are again required.

From the computed flow f' in the demand network $G(t')$, it is straightforward 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 Algorithm 7 OFFLINE-NP.

Algorithm 7 (OFFLINE-NP)
Input: σ_A, $\sigma(t')$, $\Gamma^1, \ldots, \Gamma^k$
Output: modified σ_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 demand network $G(t')$
3: solve the max profit flow problem (6.2) on $G(t')$
4: update σ_A and $\Gamma^1, \ldots, \Gamma^k$ accordingly and return them

Example 6.4. Consider the instance $(M, \sigma, p, 10, 2, 1)$ of the NP-$TaxiMP$ from Example 6.3. REPLAN-NP proceeds with this request sequence σ as follows. At the beginning, REPLAN-NP 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-NP 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_0, see Figure 6.3.

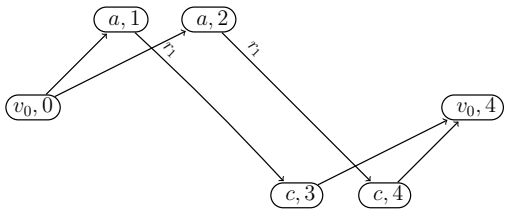


Figure 6.3: The demand network G_0 from Example 6.4.

REPLAN-NP solves the max profit flow problem (6.2) on G_0, obtains

$$\Gamma^1 = (v_0, 0) \to (a, 1) \xrightarrow{r_1} (c, 3) \to (v_0, 4)$$
$$\Gamma^2 = (v_0, 0) \to (v_0, 4)$$

accepts r_1 and moves VIPA 1 towards a.

At time $t' = 1$, $r_2 = (1, c, f, 5, 9, 1)$ and $r_3 = (1, e, g, 2, 5, 1)$ are released. REPLAN-NP 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_1, see Figure 6.4.

REPLAN-NP solves the max profit flow problem (6.2) on G_1, obtains

$$\Gamma^1 = (a, 1) \xrightarrow{r_1} (c, 3) \to (c, 5) \xrightarrow{r_2} (f, 7) \to (v_0, 7)$$
$$\Gamma^2 = (v_0, 1) \to (e, 2) \xrightarrow{r_3} (g, 4) \to (v_0, 7)$$

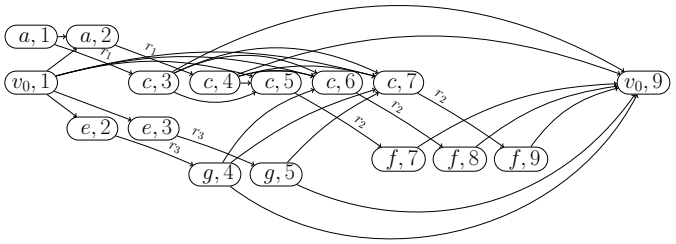


Figure 6.4: The demand network G_1 from Example 6.4.

accepts r_2 and r_3 and moves VIPA 1 towards c (serving r_1) and VIPA 2 towards e.
At time $t' = 3$, r_1 is served, r_3 is currently being served and $r_4 = (3, b, f, 4, 7, 1)$ and $r_5 = (3, b, g, 4, 7, 1)$ are released. REPLAN-NP computes the partial optimal offline solution for $\sigma_A(3) = \{r_2\}$, $\sigma_N(3) = \{r_4, r_5\}$, $S(3) = (3, 4)$ and $P(3) = (c, g)$ on the network G_3, see Figure 6.5.

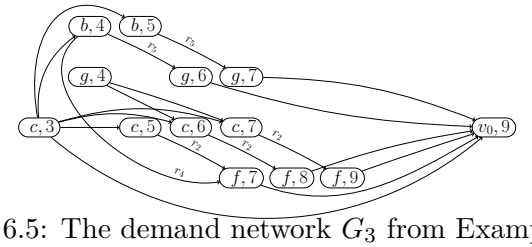


Figure 6.5: The demand network G_3 from Example 6.4.

REPLAN-NP solves the max profit flow problem (6.2) on G_3, obtains

$$\Gamma^1 = (c, 3) \to (b, 4) \xrightarrow{r_4} (f, 7) \to (v_0, 8)$$
$$\Gamma^2 = (g, 4) \to (c, 6) \xrightarrow{r_2} (f, 8) \to (v_0, 8)$$

accepts r_4 and rejects r_5. REPLAN-NP moves VIPA 1 towards b then towards f (serving r_4) and VIPA 2 towards c (note that r_2 is replanned to be served by VIPA 2 with pickup time 6).
At time $t' = 5$, r_3 is served, r_4 is currently being served and $r_6 = (5, b, c, 6, 8, 1)$ and $r_7 = (5, b, e, 6, 9, 1)$ are released. REPLAN-NP computes the partial optimal offline solution for $\sigma_A(5) = \{r_2\}$, $\sigma_N(5) = \{r_6, r_7\}$, $S(5) = (7, 6)$ and $P(5) = (f, c)$ on the network G_5, see Figure 6.6.
REPLAN-NP solves the max profit flow problem (6.2) on G_5, obtains

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

and rejects r_6 and r_7, in order to serve r_2 and r_4 from $\sigma_A(5)$.
In total, REPLAN-NP accepts 4 requests with $\sigma_A = \{r_1, r_2, r_3, r_4\}$ and serves them by the tours

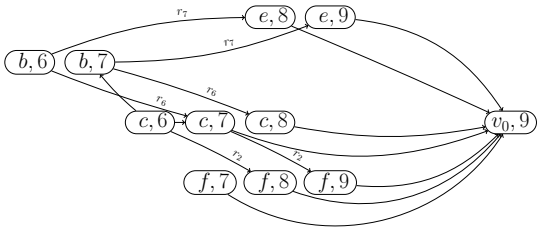


Figure 6.6: The demand network G_5 from Example 6.4.

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

with a total tour length of 14 and an objective function value of 4. ◊

6.2 Solving the Preemptive Taxi Mode Problem (P-TaxiMP)

In this section, we propose two methods to solve the Preemptive Taxi Mode Problem optimally first using a path formulation and second using multicommodity flow.

Due to the long computation time of finding a good or an optimal solution, we then propose a flow-based heuristic that computes a solution in two phases: (1) a preprocessing phase where we aim at computing the arcs in a time-expanded network and (2) computing a transportation schedule on a reduced time-expanded network. We then employ the flow-based heuristic within the framework of a REPLAN strategy to handle the online situation.

6.2.1 Offline P-TaxiMP

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

The arc set $A_T = A_W \cup A_M$ is composed of

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

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

6.2.1.1 Offline P-TaxiMP using a path formulation

The idea of this model is based on a path formulation. Actually, if we consider a full request $r_j = (t_j, x_j, y_j, p_j, q_j, z_j)$ with $p_j + d(x_j, y_j) \leq q_j$, there exists at least one path from x_j to y_j in the graph G. Hereby there exists at least one path from the node (x_j, p_j) to the node (y_j, q_j) in the time expanded network G_T. In this model, after constructing the time expanded network G_T, enumerating all possible paths between the origin and the destination of each request is required. Then, we have to chose one of them to serve the request. In the following, we detail the decision variables used in this model as well as the constraints. The VIPAs shall form a flow F through this time-expanded network G_T. To correctly initialize the system, we use the node $(v_0, 0) \in V_T$ as source for the flow F and set its balance according to the number k of available vehicles, see (6.3b). For all internal nodes $(v, t) \in V_T \setminus \{(v_0, 0), (v_0, T)\}$, we use normal flow conservation constraints, see (6.3c), which also automatically ensures that a flow of value k is entering the sink (v_0, T).

There exist many possible paths in G_T that allow us to serve a request (starting from (x_j, p_j) of a request r_j and finishing at (y_j, q_j)). A path p_i^j of a request r_j is a sequence of nodes $(v, t) \in V_T$ to be visited exactly once provided that the load of a request is always less than the capacity of the VIPA, otherwise the request is split. Each path p_i^j of a request r_j starts at (x_j, p_j) of the request r_j and finishes at (y_j, q_j). We distinguish $|\sigma|$ subsets P^j of paths in the set P_σ such that $P_\sigma = \bigcup_{j=1}^{|\sigma|} P^j$, where each subset P^j consists of the different paths a vehicle can drive along starting from the pickup station arriving to the delivery station of the corresponding request r_j within the specified time window. Hereby, the routing decisions are represented by the binary variables

$$X_{ji} = \begin{cases} 1, & \text{if a request } r_j \text{ is served along path } p_i^j \in P^j \\ 0, & \text{otherwise} \end{cases}$$

To ensure that a request can be rejected and is not served more than once, we require that at most one path is chosen from P^j for each request r_j, see (6.3d). To ensure that the capacity of the VIPA is respected on all arcs $a \in A_T$, we require that

$$\sum_{p_i^j \ni a} X_{ji} z_j \leq \text{Cap} \cdot F(a)$$

We consider a max-profit flow problem to decide which requests can be served without spending more costs than gaining profit by serving them. Accordingly, our objective function (6.3a) considers profits $p(j)$ to serve a request r_j, the costs correspond to the traveled distances

$$c(a) := d(u, v)$$

on all arcs $a = (u, v)$. The corresponding integer linear program is as follows:

$$\max \sum_{r_j \in \sigma, p_i^j \in P^j} p(j) X_{ji} - \sum_{a \in A_T} c(a) F(a) \tag{6.3a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v_0,0)} F(a) = k \tag{6.3b}$$

$$\sum_{a \in \delta^-(v,t)} F(a) = \sum_{a \in \delta^+(v,t)} F(a) \qquad \forall (v,t) \neq (v_0,0), (v_0,T) \tag{6.3c}$$

$$\sum_{p_i^j \in P^j} X_{ji} \leq 1 \qquad \forall r_j \in \sigma \tag{6.3d}$$

$$\sum_{p_i^j \ni a} z_j X_{ji} \leq \text{Cap} \cdot F(a) \qquad \forall a \in A_T, \forall p_i^j \in P_\sigma \tag{6.3e}$$

$$F(a) \geq 0 \qquad \forall a \in A_T \tag{6.3f}$$

$$F(a) \in \mathbf{Z} \qquad \forall a \in A_T \tag{6.3g}$$

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

The integer linear program (6.3) solves the offline version of the Preemptive Taxi Mode Problem where the whole sequence σ of requests is known at time $t = 0$ to optimality, provided that we can enumerate all the possible paths of all requests of σ.

Remark 6.5. This model with $F(a) \leq 1 \ \forall a \in A_T$ handles the condition that no two VIPAs must use a same arc simultaneously such that there is no VIPA that blocks another, and the significant reliability requirements of using the VIPA are ensured. ⧫

Theorem 6.6. *The integer linear program (6.3) provides an optimal solution of the Offline Preemptive Taxi Mode Problem.*

Proof. From constraint (6.3b), it is ensured that there are exactly k available VIPAs at the start station. By starting the flow F from the depot $(v_0, 0)$, it is ensured that each VIPA starts its tour from the depot, and with the constraint (6.3c) we ensure that there exist k paths P in G_T from the source node $(v_0, 0)$ to the sink node (v_0, T). The tracks of the VIPAs over time can be recovered from the flow $F(a)$ on the arcs by standard flow decomposition, see [2]. Hereby, VIPA flows on transportation arcs correspond to a move in the tour of the corresponding VIPA. After constructing a sequence of moves $\mathcal{M}$ for the k tours for the VIPAs, we insert between them the actions. The actions are incurred by the binary variables X_{ji}. Hereby, from a binary variable X_{ji} that is equal to 1, we can determine that the request r_j is accepted, and through which path p_i^j is served. An accepted request r_j served through the path p_i^j implies that a VIPA traverses the same path (constraints (6.3e)), therefore, we can add the corresponding pickup and delivery actions between the corresponding moves of the tour that contains this path. □

6.2.1.2 Offline P-TMP using multicommodity coupled flow

The first presented model in Section 6.2.1.1 solves the Offline Preemptive TaxiMP but requires many steps of preprocessing in order to have all possible paths for each request. In this section, we present a similar model where we do not generate the paths in advance. By introducing a commodity for each request, we are able to compute the optimal path for each request using a multicommodity flow. For that, we use the same time-expanded network $G_T = (V_T, A_T)$ presented in the beginning of Section 6.2.1 based on σ and the original network G.
On the time-expanded network G_T, we define a VIPA flow F to encode the route of the VIPAs through G_T. To correctly initialize the system, we use again the nodes $(v_0, 0) \in V_T$ and $(v_0, T) \in V_T$ as source and sink for the flow F and set the balance of the source accordingly to the number k of available vehicles, see (6.4b). For all internal nodes $(v, t) \in V_T \setminus \{(v_0, 0), (v_0, T)\}$, we use normal flow conservation constraints, see (6.4c), which also automatically ensure that a flow of value k is entering the sink (v_0, T).

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

To ensure that a request can be rejected and is not served more than once, we require that for each f_j at most one outgoing arc from the commodity's origin is chosen, see (6.4d). We use normal flow conservation constraints, see (6.4e), which also automatically ensure that for each commodity f_j in case a flow of 1 is leaving the commodity's origin a flow of 1 is entering the commodity's destination. To ensure that the capacity of the VIPA is respected on all arcs $a \in A_T$, we require that

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

Thus, the capacities for f_j on the transportation arcs are not given by constants but by a function.

Note that due to these flow coupling constraints, the constraint matrix of the network is not totally unimodular (as in the case of uncoupled flows) and therefore integrality constraints for all flows are required (6.4h) and (6.4i), reflecting that solving the problem is $\mathcal{NP}$-hard.

We consider a max-profit flow problem to decide which requests can be served without spending more costs than gaining profit by serving them. Accordingly, our objective function (6.4a) considers profits $p(j)$ for each commodity f_j to serve a request r_j, therefore a profit $p(j)$ to serve a request r_j is considered by putting profit on an arc a where $a \in \delta^-(x_j, p_j)$, whereas all other arcs have zero profits. The costs correspond to the

traveled distances $c(a) := d(u, v)$ on all arcs. The corresponding integer linear program is as follows:

$$\max \sum_{r_j \in \sigma} \sum_{a \in \delta^-(x_j, p_j)} p(j) f_j(a) - \sum_{a \in A_T} c(a) F(a) \tag{6.4a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v_0, 0)} F(a) = k \tag{6.4b}$$

$$\sum_{a \in \delta^-(v,t)} F(a) = \sum_{a \in \delta^+(v,t)} F(a) \qquad \forall (v,t) \neq (v_0, 0), (v_0, T) \tag{6.4c}$$

$$\sum_{a \in \delta^-(x_j, p_j)} f_j(a) \leq 1 \qquad \forall r_j \in \sigma \tag{6.4d}$$

$$\sum_{a \in \delta^-(v,t)} f_j(a) = \sum_{a \in \delta^+(v,t)} f_j(a) \qquad \forall r_j \in \sigma \forall (v,t) \neq (x_j, p_j), (y_j, q_j) \tag{6.4e}$$

$$\sum_{r_j \in \sigma} f_j(a) \cdot z_j \leq \mathrm{Cap} F(a) \qquad \forall a \in A_T \tag{6.4f}$$

$$F(a) \geq 0 \qquad \forall a \in A_T \tag{6.4g}$$

$$F(a) \in \mathbf{Z} \qquad \forall a \in A_T \tag{6.4h}$$

$$f_j(a) \in \{0, 1\} \qquad \forall a \in A_T, \forall r_j \in \sigma \tag{6.4i}$$

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 Preemptive Taxi Mode Problem, where the whole sequence σ of requests is known at time $t = 0$ to optimality.

Remark 6.7. This model with $F(a) \leq 1, \ \forall a \in A_T$ handles the condition that no two VIPAs must use a same arc simultaneously such that there is no VIPA that blocks another, and the significant reliability requirements of using the VIPA are ensured. ⧫

In Figure 6.7, we show the computed flow of the Example 6.4 by solving the integer linear program (6.4).

The next theorem shows that one can construct a transportation schedule from a solution of this integer linear program. It can be proven in a similar way as Theorem 6.6.

Theorem 6.8. *The integer linear program (6.4) provides an optimal solution of the Offline Preemptive Taxi Mode Problem.*

Proof. From constraint (6.4b) it is ensured that there are exactly k available VIPAs at the start station. By starting the flow F from the depot $(v_0, 0)$, it is ensured that each VIPA starts its tour from the depot, and with the constraint (6.4c) we ensure that there exists k paths P in G_T from the source node $(v_0, 0)$ to the sink node (v_0, T). The tracks of the VIPAs over time can be recovered from the flow $F(a)$ on the arcs by standard

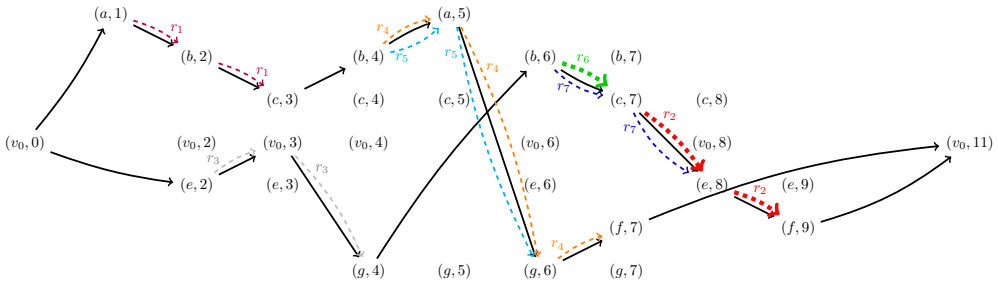


Figure 6.7: This figure shows the tours of the VIPAs, the arcs a with positive flow $F(a)$ in the time expanded network G_T of Example 6.4, illustrated by solid arcs. For each request r_j, the corresponding arcs a with $f_j(a) > 0$ are indicated by dashed arcs with a different color. There are 7 accepted requests over 7.

flow decomposition, see [2]. Hereby, VIPA flows on transportation arcs correspond to a move in the tour of the corresponding VIPA. After constructing a sequence of moves $\mathcal{M}$ for the k tours for the VIPAs, we insert between them the actions. The actions are incurred by the set of commodities f_j. Hereby, from a positive flow for a commodity f_j, we can determine that the request r_j is accepted, and through which arcs is served by determining the arcs a that have a $f_j(a) = 1$. The set of arcs having $f_j(a) = 1$ is the (s,t)-path of the commodity f_j from the commodity's origin (x_j, p_j) to the commodity's destination (y_j, t_j^{drop}) (there cannot be more than one (s,t)-path for each commodity f_j by constraints (6.4d) and (6.4e). An accepted request r_j served through the (s,t)-path implies that a VIPA traverses the arcs from (x_j, t_j^{pick}) to (y_j, t_j^{drop}) (constraints (6.4f)), therefore we can add the corresponding pickup and delivery actions between the corresponding moves of the tour that contains these arcs. □

6.2.1.3 A Flow-Based Heuristic for the Offline P-TaxiMP

Computing an optimal solution either using a path formulation or a multicommodity coupled flow is generally very slow and, thus, not applicable in practice in case of having a large scale on two dimensions (time and network). However, the runtime can be improved by reducing the number of arcs and nodes in the time-expanded network, which corresponds to a reduction of variables in the corresponding integer linear program. As experiments have shown that only a small percentage of arcs in G_T is used in the optimal solution while solving the Offline Preemptive Taxi Mode Problem using multicommodity coupled flow, the idea is to reduce G_T to a network containing only arcs which are taken in the optimal solution with high probability. Therefore, we present a heuristic that solves the heuristic Offline Preemptive Taxi Mode Problem in two phases. In the first phase, we compute those arcs which are likely to be used in an optimal solution. In the second phase, we construct a reduced time-expanded network G_T^h, where we keep only the previously computed arcs by the three steps of the first phase and discard all others; afterwards, we solve the integer linear program (6.4) on this reduced network G_T^h. This

does not lead to a globally optimal solution on G_T, but provides reasonable solutions in short time.

Preprocessing (Phase 1) This phase is performed in two steps:

- compute a classic max profit multicommodity flow problem by adjusting the profits and the costs,
- compute a classical min cost multicommodity flow problem by adjusting the profits and the costs to compute a shortest path for each of the requests if needed.

Max Profit Multicommodity Flow and its Linear Program. In this step, we build a time-expanded network $G_T^h = (V_T^h, A_T^h)$ based on σ and the original network G similar to the time-expanded network $G_T = (V_T, A_T)$ in Section 6.2.1.2 with few differences:
The node set V_T^h is exactly the same as V_T. In addition we distinguish

- $|\sigma|$ subsets V_x^j of nodes in V_T^h where each subset V_x^j consists of all possible origins (x_j, t_j^{pick}) of the corresponding request r_j, for all $p_j \leq t_j^{pick} \leq q_j - d(x_j, y_j)$, so that we have $V_x = \bigcup_{j=1}^{|\sigma|} V_x^j$.
- $|\sigma|$ subsets V_y^j of nodes in V_T^h where each subset V_y^j consists of all possible destinations (y_j, t_j^{drop}) of the corresponding request r_j, for all $p_j + d(x_j, y_j) \leq t_j^{drop} \leq q_j$, so that we have $V_y = \bigcup_{j=1}^{|\sigma|} V_y^j$.

The arc set $A_T^h = A_W \cup A_M$ is composed of the wait arcs A_W and transport arcs A_M that are the same as in A_T. On the time-expanded network $G_T^h = (V_T^h, A_T^h)$ we consider only a set of commodities f_j^1, $j \in \{1 \cdots \sigma\}$ for the requests. Unlike in the exact approach, in this step, the flow corresponding to each request is not coupled to the VIPA flow. The reason why there is no VIPA flow is that in this step we want to construct interesting paths without taking into consideration the route of the VIPAs. Therefore, this step seeks to determine the "gaining and profitable" path for each commodity starting from the commodity's origin and ending at the commodity's destination while passing especially by origins and destinations of other requests (as the profits are set on the possible pickup and delivery nodes of the requests). Accordingly, our objective function (6.6a) considers profits $p(a)$ for the commodity f_j^1 on all $a = ((v,t),(v',t')) \in A_M$ having a pickup or delivery node as an origin or destination of the arc (v,t) or $(v',t') \in V_x \cup V_y$, whereas all other arcs have zero profits.

The corresponding linear program is as follows:

$$\max \sum_{r_j \in \sigma} \sum_{a \in A_T^h} p(a) f_j^1(a) \tag{6.5a}$$

$$\text{s.t.} \sum_{a \in \delta^+(x_j, p_j)} f_j^1(a) = 1 \qquad \forall r_j \in \sigma \tag{6.5b}$$

$$\sum_{a \in \delta^-(y_j, q_j)} f_j^1(a) = 1 \qquad \forall r_j \in \sigma \tag{6.5c}$$

$$\sum_{a \in \delta^-(v,t)} f_j^1(a) = \sum_{a \in \delta^+(v,t)} f_j^1(a) \qquad \forall (v,t) \neq (x_j, p_j), (y_j, q_j), \forall r_j \in \sigma \tag{6.5d}$$

$$f_j^1(a) \geq 0 \qquad \forall a \in A_T^h \tag{6.5e}$$

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).
In order to avoid unnecessary waiting moves in the beginning of the paths for each commodity, we can set some costs on waiting arcs $a = (v,t,v,t')$, e.g,

$$c(a) = t' \; \forall a \in A_w$$

Then we can adjust the objective function by adding these costs to the objective function resulting in

$$\max \sum_{r_j \in \sigma} \sum_{a \in A_T^h} p(a) f_j^1(a) \quad - \sum_{r_j \in \sigma} \sum_{a \in A_T^h} c(a) f_j^1(a)$$

The above linear program can be computed in polynomial time. It is simply a continuous multicommodity flow problem which is a well-known polynomially-solvable problem [2]. Thus, constraining the flow to be integer is not necessary since the constraint matrix is totally unimodular. The resulting computed flows of the Example 6.4 are illustrated in Figure 6.8.

Min Cost Multicommodity Flow and its Linear Program. In this step, we also use a set of commodities for the requests. We compute a min cost multicommodity flow in order to build a shortest path for each request. For that, we use the same time-expanded network $G_T^h = (V_T^h, A_T^h)$ built in the first step of this heuristic, based on σ and the original network G similar to the time-expanded network $G_T = (V_T, A_T)$ in Section 6.2.1.2.

The arc set $A_T^h = A_W \cup A_M$ is composed of the wait arcs A_W and transport arcs A_M that are the same as in A_T. We consider a min-cost multicommodity flow problem to construct a shortest path for each request. Accordingly, our objective function (6.6a) considers costs $c(a) = d(u,v)$ for the flow f_j^2 on all $a \in A_M$, costs $c(a) = t'$ for the flow f_j^2 on all $a = (v,t,v,t') \in A_W$ (so that we compute a shortest path starting from the earliest possible pickup time).

The corresponding linear program is detailed in (6.6).

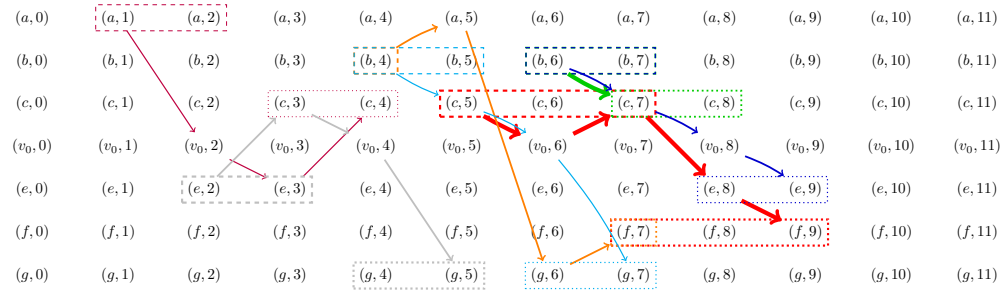


Figure 6.8: This figure illustrates the resulting computed flows f_j^1 based on the original network from Example 6.4. Each request r_j corresponds to a commodity f_j^1 and the result of computing a max-profit multicommodity flow is the $|\sigma|$ different paths shown in this figure. Each path starts from the commodity's origin (x_j, p_j) with the earliest pickup station of the corresponding request and ends with the latest possible delivery station while trying to pass by other "profitable nodes", that are the possible pickup respectively delivery stations of other requests enclosed by dashed respectively dotted rectangles in the figure.

$$\min \sum_{a \in A_T^h} c(a) f_j^2(a) \tag{6.6a}$$

$$\text{s.t.} \sum_{a \in \delta^+(x_j, p_j)} f_j^2(a) = 1 \qquad \forall r_j \in \sigma \tag{6.6b}$$

$$\sum_{a \in \delta^-(y_j, q_j)} f_j^2(a) = 1 \qquad \forall r_j \in \sigma \tag{6.6c}$$

$$\sum_{a \in \delta^-(v,t)} f_j^2(a) = \sum_{a \in \delta^+(v,t)} f_j^2(a) \qquad \forall (v,t) \neq (x_j, p_j), (y_j, q_j) \tag{6.6d}$$

$$f_j^2(a) \geq 0 \qquad \forall a \in A_T^h \tag{6.6e}$$

The above linear program can be computed in polynomial time. Thus, constraining the flow to be integer is not necessary since the constraint matrix is totally unimodular. The resulting computed flows of the Example 6.4 are illustrated in Figure 6.9.

Computing a Transportation Schedule (Phase 2) Each of the commodities of the two steps in the first phase can be rapidly computed, but the computed solution is not a feasible transportation schedule. In this section, we describe the construction of a reduced version $G_H = (V_H, A_H)$ of the original time-expanded network $G_T = (V_T, A_T)$ based on the flows computed in the preprocessing (Phase 1). Hereby, we reduce the total number of nodes as well as of wait and transport arcs. Afterwards, we compute an optimal solution on G_H providing a feasible transportation schedule.

The reduced network G_H is constructed as follows. The node set V_H is initiated by the nodes $(v_0, 0)$ and (v_0, T). The arc set $A_H = A'_W \cup A'_M$ is constructed as follows.

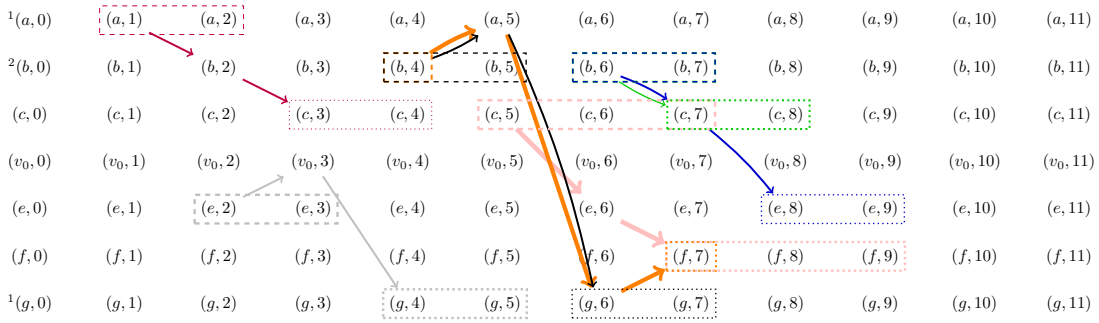


Figure 6.9: The resulting computed flows f_j^2 based on the original network from Example 6.4. Each request r_j corresponds to a commodity f_j^2 and the result of computing a min-cost multicommodity flow is the $|\sigma|$ shortest paths shown in this figure. Each path starting with the earliest pickup station of the corresponding request and ending with the convenient delivery station.

The transport arcs $a = [(v,t),(w,t')] \in A_M$ with $f_j^1(a) > 0$ or $f_j^2(a) > 0$ remain in A'_M, and we add the nodes (v,t) and (w,t') to V_H. Next, for each station $v \in V_H$, we add wait arcs in A'_W between two successive nodes on the time line of v. Note that, in the two steps of the preprocessing (Phase 1), we compute certain "gaining and profitable" paths. In the first step, a path of r_j starts from the earliest pickup node (x_j, p_j) of r_j, tries to pass by other origins and destinations and ends up in the latest possible destination (y_j, q_j) of the request r_j. In the second step of the first phase, the computed path of r_j is one of the shortest paths from (x_j, p_j) to $(y_j, p_j + d(x_j, y_j))$. Using these two steps in the preprocessing, we increase the chance to serve the requests that share (partially) a path together in one tour. However, there may be some requests that can be served sequentially one after another in one tour in the original network G_T but cannot be served together in one tour in the reduced network G_H unless we add additional transport arc from the destination of the first request to the origin of the second (if possible). Therefore, some additional transport arcs can be added using the following steps of Algorithm 8 (AAA).

We denote by V_x all possible origins (x_j, t_j^{pick}) of all requests r_j in σ, for all $p_j \leq t_j^{pick} \leq q_j - d(x_j, y_j)$, and by V_y all possible destinations (y_j, t_j^{drop}) of all r_j in σ, for all $p_j + d(x_j, y_j) \leq t_j^{drop} \leq q_j$. The input of the AAA algorithm is V_x, V_y, and $G_H = (V_H, A_H)$. The output is the final reduced network $G_H = (V_H, A_H)$ on which we compute a feasible transportation schedule by solving the integer linear program (6.4) from Section 6.2.1.2. The resulting reduced network the Example 6.4 is illustrated in Figure 6.10.

Algorithm 8 Add Additional Arcs (AAA)

Input: $V_x, V_y, G_H = (V_H, A_H)$
Output: modified G_H

For all nodes $(v,t) \in V_H \cap V_y$
  For all nodes $(v',t') \in V_H \cap V_x$
   if $t + d(v,v') \leq t'$
    if there does not exist a path $\in A_H$ between (v,t) and (v',t')
     add the arc $((v,t),(v',t'))$ to the transport arcs A'_M
     update G_H
return G_H

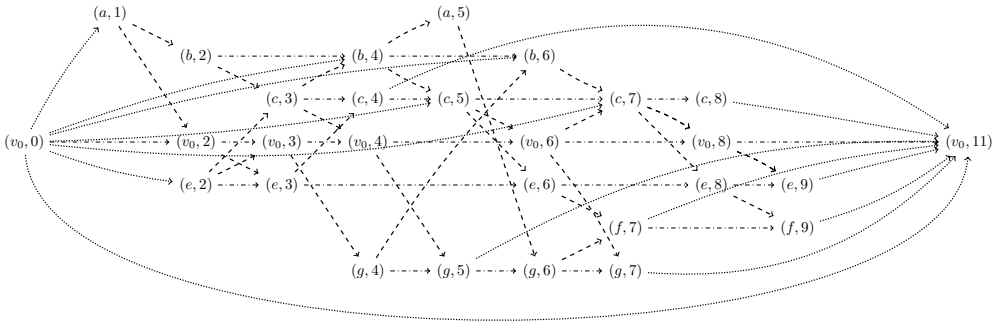


Figure 6.10: This figure illustrates the reduced network G_H of Example 6.4. the transportation arcs are indicated by dashed arcs, the source and the source and sink arcs by dotted arcs. The wait arcs are represented by dashdotted arcs. Note that $((c,3),(b,4))$ and $((g,4),(b,6))$ are added by AAA Algorithm while constructing the reduced network. Overall the number of arcs is significantly reduced in G_H. The original time expanded network G_T has 84 nodes and 313 arcs among which 220 transport arcs, as for the reduced network G_H, it has 28 nodes and 71 arcs among which 27 transport arcs.

Computing a Feasible Transportation Schedule. A feasible transportation schedule is computed by solving the integer linear program (6.4) from Section 6.2.1.2 on the reduced time-expanded network G_H. The problem is always feasible due to the following reason: in the Preemptive Taxi Mode Problem, every customer request can be rejected. Due to the wait arcs, for every station $v \in V$ there is a path from the source node $(v,0)$ to the node (v,T) for VIPA flow and the request flows. Thus, we can directly conclude that the flow-based heuristic always computes a feasible solution for the Offline Preemptive Taxi Mode Problem.

In Figure 6.11, we show the paths used in the optimal offline solution. In this example they are all present in the reduced network G_H. In this example FLOW-HEURISTIC provides the optimal solution, the flow computed has the same values as the optimal

solution shown in Figure 6.7 computed by the exact formulation using a multicommodity flow. The computed flow in the reduced network of Example 6.4 is illustrated in Figure 6.12.

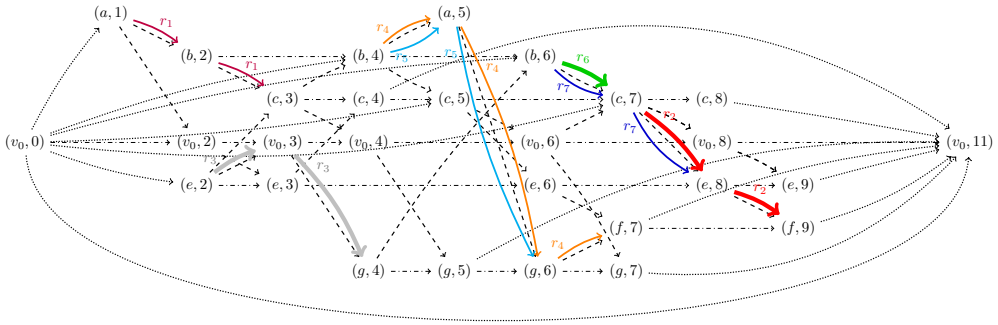


Figure 6.11: This figure illustrates the paths used in the optimal offline solution for each request r_j, and the arcs (dashed) of the reduced network G_H of Example 6.4. For each request r_j, the corresponding arcs are indicated by solid arcs with a different color.

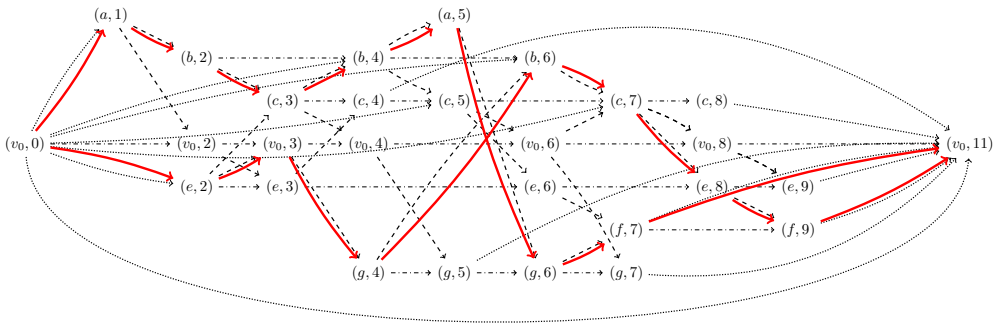


Figure 6.12: This figure illustrates the arcs with positive flow F in the reduced network G_H of Example 6.4. There are 7 accepted requests. In this example FLOW-HEURISTIC provides the optimal solution.

6.2.2 Online P-TaxiMP via Heuristic Replan

To handle the online situation, where the requests in σ are released over time during a time horizon $[0, T]$, we propose a heuristic to solve a sequence of offline subproblems for certain time intervals $[t', T']$ within $[0, T]$ on accordingly modified time-expanded networks. A usual replan strategy is based on computing the optimal solution on the subsequence of requests released in each replanning step. In the proposed hREPLAN-P

(see Algorithm 9), we use the previously presented heuristic to compute offline solutions on the subsequence $\sigma(t')$.

Algorithm 9 (hREPLAN-P)

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

Output: σ_A^h, and tours $\Gamma^1, \ldots, \Gamma^k$

1: initialize $\sigma_A^h = \emptyset$, $\sigma^h(t') = \{r_j \in \sigma : t_j = 0\}$, and $\Gamma^i = (v_0, 0)$ for $1 \leq i \leq k$

2: WHILE $t' \leq T$ DO: call hOFFLINE-P(σ_A^h, $\sigma^h(t')$, $\Gamma^1, \ldots, \Gamma^k$)
perform the (modified) tours until new requests become known
update t' and $\sigma^h(t')$

3: return σ_A^h, and $\Gamma^1, \ldots, \Gamma^k$

To compute those offline solutions for the subsequences $\sigma(t')$, we build a time-expanded network $G_T^h(t') = (V_{T'}^h, A_{T'}^h)$ based on $\sigma(t')$ and the original network G, apply the first phase of FLOW-HEURISTIC, apply the phase 2 of FLOW-HEURISTIC on the reduced network $G_H(t')$ obtained and consider a flow in $G_H(t')$ that corresponds to the studied (partial) tours.

We construct $G_{T'}^h = (V_{T'}^h, A_{T'}^h)$ and $G_H(t') = (V_H', A_H')$ in a similar way as G_T^h and G_H for the offline situation. The main difference is that we do not have a single source $(v_0, 0)$, 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 $P^h(t')$ and start times $S^h(t')$ for the VIPAs from the current tours $\Gamma^1, \ldots, \Gamma^k$. At the beginning, i.e. at time $t = 0$, we clearly have $P^h(t')_i = v_0$ and $S^h(t')_i = 0$. 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 a new fictive pickup node $x_j = P^h(t')_i$ which is the position of the VIPA at time t' and a new fictive earliest pickup time $t_j^{pick} = S^h(t')_i$ which is the time t'. If the VIPA is actually at a node or the expected arrival time if the VIPA is at an arc towards the node; otherwise, $P^h(t')_i$ is the current position v of VIPA i and $S^h(t')_i = t'$.

To keep previously accepted requests, we partition $\sigma^h(t')$ into the subsequences

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

For the first phase of FLOW-HEURISTIC, at the beginning, i.e. at time $t = 0$, for each commodity f_j we clearly have the node (x_j, p_j) as the commodity's origin. At any later time point t', the commodity's origins are as follows:

- for the requests $r_j \in \sigma_C^h(t')$, where VIPA i is currently serving a request r_j, we use a new fictive pickup node $(P^h(t')_i, t')$ as source for the corresponding commodity, the commodity's destination would not change.

- for all requests $r_j \in \sigma_N^h(t') \cup \sigma_A^h(t')$, we use (x_j, p_j) as source for the corresponding commodity.

For the second phase of FLOW-HEURISTIC, the network $G_H(t')$ is constructed in a similar way as G_H based on the arcs in $G_T^h(t')$ having positive flows in the first or second step of the first phase. Some additional arcs may be added by Algorithm 8 (AAA). For the requests $r_j \in \sigma_C^h(t')$, where VIPA i is currently serving a request r_j, we use $(P^h(t')_i, t')$ as a new fictive pickup node. As we are using a reduced network $G_H(t')$ to compute the transportation schedule during the second phase, there may be some arcs absent in $G_H(t')$. In order to ensure that all the accepted requests $r_j \in \sigma_C^h(t') \cup \sigma_A^h(t')$ can be served in the new computed transportation schedule, we add all the transportation arcs (u, t^u, v, t^v) of the previous transportation schedule such that $t^u \geq t'$ to $G_H(t')$ if they were not already present. Moreover, another difference is that we do not have a single source $(v_0, 0)$, but that we need to use the possible start positions and possible start times of the VIPAs as sources. Therefore, we denote by V_+, the VIPAs start positions and start times $(P^h(t')_i, t')$ for $1 \leq i \leq k$ as sources in V_+. In phase 2 of FLOW-HEURISTIC, we compute a transportation schedule by solving the max profit flow problem in $G_H(t') = (V'_H, A'_H)$ detailed in (6.7).

$$\max \sum_{r_j \in \sigma^h(t')} \sum_{a \in \delta^-(x_j, p_j)} p(a) f'_j(a) \sum_{a \in A'_H} c(a) F'(a) \tag{6.7a}$$

$$\text{s.t.} \sum_{a \in \delta^+(v,t')} F'(a) = k(v) \qquad \forall (v,t) \in V_+ \tag{6.7b}$$

$$\sum_{a \in \delta^-(v,t)} F'(a) = \sum_{a \in \delta^+(v,t)} F'(a) \qquad \forall (v,t) \neq V_+ \cup \{(v_0, t')\} \tag{6.7c}$$

$$\sum_{a \in \delta^-(x_j, p_j)} f'_j(a) \leq 1 \qquad \forall r_j \in \sigma_N^h(t') \tag{6.7d}$$

$$\sum_{a \in \delta^-(x_j, p_j)} f'_j(a) = 1 \qquad \forall r_j \in \sigma_C^h(t') \cup \sigma_A^h(t') \tag{6.7e}$$

$$\sum_{a \in \delta^-(v,t)} f'_j(a) = \sum_{a \in \delta^+(v,t)} f'_j(a) \qquad \forall r_j \in \sigma, \forall (v,t) \neq (x_j, p_j), (y_j, q_j) \tag{6.7f}$$

$$\sum_{r_j \in \sigma^h(t')} f'_j(a) \cdot z_j \leq \mathrm{Cap} F'(a) \qquad \forall a \in A'_M \tag{6.7g}$$

$$F'(a) \geq 0 \qquad \forall a \in A'_H \tag{6.7h}$$

$$F'(a) \in \mathbf{Z} \qquad \forall a \in A'_H \tag{6.7i}$$

$$f'_j(a) \in \{0, 1\} \qquad \forall a \in A'_H, \forall r_j \in \sigma^h(t') \tag{6.7j}$$

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 (6.7e) ensure that previously accepted requests are served whereas constraints (6.7e) allow to reject newly released requests.

From the computed flows F' and f'_j in the time-expanded network $G_H(t')$, it is straightforward to determine newly accepted requests (corresponding to requests r_j with $f'_j(a) > 0$ for some $a \in A'_H$) and to construct (partial) tours $\Gamma^1, \ldots, \Gamma^k$ for the VIPAs in the same way as described for the offline situation in Theorem 6.8.

The whole process can be summarized in Algorithm 10 hOFFLINE-P.

Algorithm 10 (hOFFLINE-P)

Input: σ_A^h, $\sigma^h(t')$, $\Gamma^1, \ldots, \Gamma^k$

Output: modified σ_A^h and modified tours $\Gamma^1, \ldots, \Gamma^k$

1: determine VIPA start positions $P^h(t')$ and start times $S^h(t')$ from $\Gamma^1, \ldots, \Gamma^k$
2: create the time-expanded network $G_T^h(t')$
3: solve FLOW-HEURISTIC using $G_T^h(t')$
4: update σ_A^h, and $\Gamma^1, \ldots, \Gamma^k$ accordingly and return them

Example 6.9. Consider the instance $(M, \sigma, p, 10, 2, 1)$ of the P-TaxiMP from Example 6.3. hREPLAN-P proceeds with this request sequence σ as follows. At the beginning, hREPLAN-P initializes $\sigma_A^h = \emptyset$, 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. hREPLAN-P computes the partial heuristic offline solution for $\sigma_A^h(0) = \emptyset$, $\sigma_C^h(0) = \emptyset$, $\sigma_N^h(0) = \{r_1\}$, $S^h(0) = (0, 0)$ and $P^h(0) = (v_0, v_0)$, computes a transportation schedule with the time-expanded network G_T^h and G_H constructed using FLOW-HEURISTIC (6.2.1.3), obtains

$$\begin{aligned}\Gamma^1 =& (v_0, 0) \xrightarrow{\quad r_1 \quad} (a, 1) \to (b, 2) \to (c, 3) \to (v_0, 4)\\ \Gamma^2 =& (v_0, 0) \to (v_0, 4)\end{aligned}$$

accepts r_1 and moves VIPA 1 towards a.
At time $t' = 1$, $r_2 = (1, c, f, 5, 9, 1)$ and $r_3 = (1, e, g, 2, 5, 1)$ are released. hREPLAN-P computes the partial offline solution for $\sigma_A^h(1) = \{r_1\}$, $\sigma_C^h(1) = \emptyset$, $\sigma_N^h(1) = \{r_2, r_3\}$, $S^h(1) = (1, 1)$ and $P^h(1) = (a, v_0)$, recomputes a transportation schedule with the time-expanded network G_T^h and G_H constructed using FLOW-HEURISTIC (6.2.1.3), obtains

$$\begin{aligned}\Gamma^1 =& (a, 1) \xrightarrow{\quad r_1 \quad} (b, 2) \to (c, 3) \to (c, 5) \xrightarrow{\quad r_2 \quad} (e, 6) \to (f, 7) \to (v_0, 9)\\ \Gamma^2 =& (v_0, 1) \xrightarrow{\quad r_3 \quad} (e, 2) \to (v_0, 3) \to (g, 4) \to (v_0, 9)\end{aligned}$$

accepts r_2 and r_3 and moves VIPA 1 towards c (serving r_1) and VIPA 2 towards e.
At time $t' = 3$, r_1 is served, r_3 is currently being served and $r_4 = (3, b, f, 4, 7, 1)$, $r_5 = (3, b, g, 4, 7, 1)$ are released. hREPLAN-P computes the partial offline solution for $\sigma_A^h(3) = \{r_2\}$, $\sigma_C^h(3) = \emptyset$, $\sigma_N^h(3) = \{r_4, r_5\}$, $S^h(3) = (3, 3)$ and $P^h(3) = (c, v_0)$ hREPLAN-P recomputes a transportation schedule with the time-expanded network

G_T^h and G_H constructed using FLOW-HEURISTIC (6.2.1.3), obtains

$$\Gamma^1 = (c,3) \to (b,4) \xrightarrow{r_4} (a,5) \xrightarrow{r_5} (g,6) \to (f,7) \to (v_0,10)$$
$$\Gamma^2 = (v_0,3) \to (v_0,5) \to (c,6) \xrightarrow{r_2} (e,7) \to (f,8) \to (v_0,10)$$

accepts r_4, r_5. hREPLAN-P moves VIPA 1 towards b then towards a then towards g (serving r_5) then towards f (serving r_4) and VIPA 2 towards c (note that r_2 is replanned to be served by VIPA 2 with pickup time 6).
At time $t' = 5$, r_3 is served, r_4 and r_5 are currently being served and $r_6 = (5, b, c, 6, 8, 1)$ and $r_7 = (5, b, e, 6, 9, 1)$ are released. hREPLAN-P computes the partial optimal offline solution for $\sigma_A^h(5) = \{r_2\}$, $\sigma_C^h(5) = \{r_4, r_5\}$, $\sigma_N^h(5) = \{r_6, r_7\}$, $S^h(5) = (5,5)$ and $P^h(5) = (a, v_0)$, recomputes a transportation schedule with the time-expanded network G_T^h and G_H constructed using the flow based heuristic (6.2.1.3), obtains

$$\Gamma^1 = (a,5) \xrightarrow{r_4} (g,6) \to (f,7) \to (v_0,8)$$
$$\Gamma^2 = (v_0,5) \to (c,6) \xrightarrow{r_2} (e,7) \to (f,8) \to (v_0,10)$$

and rejects r_6 and r_7, in order to serve r_2, r_4 and r_5 from $\sigma_A^h(5)$ and $\sigma_C^h(5)$.
In total, hREPLAN-P accepts 5 requests with $\sigma_A^h = \{r_1, r_2, r_3, r_4, r_5\}$ and serves them by the tours

$$\Gamma^1 = (v_0,0) \to (a,1) \xrightarrow{r_1} (b,2) \to (c,3) \to (b,4) \xrightarrow{r_4} (a,5) \xrightarrow{r_5} (g,6) \to (f,7) \to (v_0,9)$$
$$\Gamma^2 = (v_0,0) \to (v_0,1) \xrightarrow{r_3} (e,2) \to (f,3) \to (g,4) \to (a,5) \to (c,6) \xrightarrow{r_2} (e,7) \to (f,8) \to (v_0,10)$$

with a total tour length of 14 and an objective function value of 8. ◊

Computing an upper bound We may solve the uncapacitated Taxi Mode Problem or the uncapacitated PDP with time windows by using the same time-expanded network $G_T = (V_T, A_T)$ with a VIPA Flow F and a set of commodities f_j, $j \in \{1 \cdots |\sigma|\}$ for the requests. The only difference is that we remove the coupling constraint:

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

and replace it by

$$F(a) \geq f_j(a), \ \forall a \in A_M$$

Therefore by relaxing the coupling constraint and supposing that the VIPAs have infinite capacity so that we serve as much requests as possible, it is possible to directly compute an upper bound on the maximal profit as well as the maximal number of customer requests which can be theoretically served in the Preemptive Taxi Mode Problem.

6.3 Competitive Analysis

Concerning the competitive analysis of the Online Non-Preemptive and the Preemptive Taxi Mode Problem, we in fact, obtain the more general result that no (deterministic) online algorithm for the Online TaxiMP is competitive against two common types of adversaries. As the examples provided below prove that there is no finite number bounding the ratio between the optimal offline algorithm OPT and any online algorithm ALG neither for the Online NP-TaxiMP nor for the P-TaxiMP, therefore in the sequel of the theorems and proofs we do not distinguish between the preemptive and non-preemptive problems but we only consider the Online Taxi Mode Problem (TaxiMP).

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

We show that an oblivious adversary can force any (deterministic) online algorithm ALG for the Online TaxiMP to reject all requests of a sequence while the adversary can accept and serve all requests, implying that ALG is not competitive.

Theorem 6.10. *There is no competitive (deterministic) online algorithm for the Online TaxiMP 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 time 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 σ. For that, we consider an instance $(M, \sigma, p, T, 1, 1)$ of the Online TaxiMP with

- the network G with depot v_0 from Figure 6.13

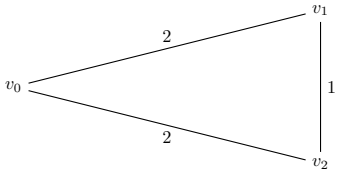


Figure 6.13: This figure illustrates the network G of the instance $(M, \sigma, p, T, 1, 1)$ of the Online TaxiMP with an oblivious adversary.

- 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 σ 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

$$ALG(\sigma) = 0.$$

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 σ (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 σ. At the end of the sequence the adversary returns its VIPA to the depot to close its tour. Thus we obtain

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

This shows

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

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

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

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

Theorem 6.11. *There is no competitive (deterministic) online algorithm for the Online TaxiMP against a non-abusive adversary.*

Proof. The idea for a worst-case sequence for an online algorithm ALG is as follows. The adversary releases a first request r_0 that is accepted by ALG and brings the VIPA operated by ALG to a station y_0 in the network whose distance $d(y_0, x_j)$ to the origins x_j of all further requests r_j is larger than the delay between the release time t_j and the latest possible pickup time $q_j - d(x_j, y_j)$. Thus, ALG has to reject all remaining requests, whereas the adversary rejects r_0 but can accept and serve all other requests in the sequence. For that, we consider an instance $(M, \sigma, p, T, 1, 1)$ of the Online TaxiMP with

- the network G with depot v_0 from Figure 6.14

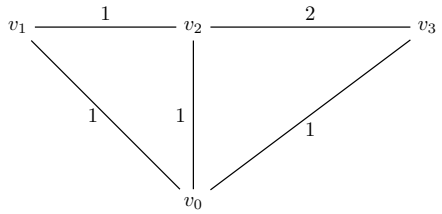


Figure 6.14: This figure illustrates the network G of the instance $(M, \sigma, p, T, 1, 1)$ of the Online TaxiMP with a non-abusive adversary.

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

$$r_0 = (0, v_2, v_3, 1, 3, 1)$$
$$r_j = (j+2, v_1, v_2, j+3, j+4, 1) \text{ for all odd } j \text{ with } 1 \leq j \leq \ell$$
$$r_j = (j+2, v_2, v_1, j+3, j+4, 1) \text{ for all even } j \text{ with } 2 \leq j \leq \ell$$

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

The online algorithm ALG treats the sequence σ as follows. At time $t = 0$, r_0 is the only released request. As the origin $x_0 = v_2$ of r_0 is reachable from the depot v_0, ALG accepts r_0 and serves it by moving its VIPA from $(v_0, 0)$ to $(v_2, 1)$ and then to the destination $(v_3, 3)$ of r_0. At time $t = 3$, request $r_1 = (3, v_1, v_2, 4, 5, 1)$ is released. As the VIPA operated by ALG is situated in v_3, ALG cannot arrive in the origin $x_1 = v_1$ of r_1 before or at the latest possible pickup time $4 = q_1 - d(v2, v_1)$ due to $d(v_3, v_1) = 2$. Thus, ALG rejects r_1 and the VIPA operated by ALG stays in v_3. At time $t = 4$, request $r_2 = (4, v_2, v_1, 5, 6, 1)$ is released. Again, ALG cannot arrive in the origin $x_2 = v_2$ of r_2 before or at the latest possible pickup time $5 = q_2 - d(v_1, v_2)$ due to $d(v_3, v_2) = 2$. Thus, ALG rejects also r_2 and the VIPA operated by ALG stays in v_3. This is repeated for any further request $r_j \in \sigma$ so that all r_j with $j > 0$ are rejected by ALG. At the end of the sequence, ALG returns its VIPA to the depot in order to close its tour. Thus we obtain

$$ALG(\sigma) = p(r_0) - (d(v_0, v_2) + d(v_2, v_3) + d(v_3, v_0)) = 2 \cdot 2 - (1 + 2 + 1) = 0$$

In contrary, the adversary rejects r_0 but is able to accept and serve all other requests by moving its VIPA from $(v_0, 3)$ to $(v_1, 4)$ as origin of r_1, then to $(v_2, 5)$ as destination of r_1 and origin of r_2, then to $(v_1, 6)$ as destination of r_2 and origin of r_3, and so on. After all requests are served, the adversary returns its VIPA (from v_1 or v_2) to the depot to close its tour. Thus, we obtain

$$OPT(\sigma) = \ell \cdot 2 \cdot d(v1, v_2) - (\ell \cdot d(v_1, v_2) + 2) = \ell \cdot d(v_1, v_2) - 2 = \ell - 2$$

This shows

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

so that there is no finite number c bounding this ratio between $OPT(\sigma)$ and $ALG(\sigma)$ for all possible sequences σ. □

Since in both cases, 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 6.12. *None of the algorithms REPLAN-NP and hREPLAN-P is competitive for the Online TaxiMP against an oblivious or non-abusive adversary.*

6.4 Computational Results

This section deals with computational experiments for the optimal offline solutions of the TaxiMP (Non-Preemptive and Preemptive) and the replan strategies (REPLAN-NP and hREPLAN-P) 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 can expect a better behavior of the proposed replan strategies for the Online TaxiMP in average.

The computational results presented in this section support this expectation. They compare the total number of accepted (and thus served) requests by REPLAN-NP and by hREPLAN-P with the optimal offline solutions OPT-NP and OPT-P. 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 [112].

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

In the first set of 180 instances, the requests have a random load between 4 and 10 (10 is the capacity of the VIPA), in the second set of 180 instances, the requests have a random load between 1 and 10 (with the same capacity of the VIPA). For each set,

we run its instances to compare the total number of accepted (and thus served) requests by REPLAN-NP (respectively hREPLAN-P) with the optimal offline solution OPT-NP (respectively OPT-P). Due to the long computation time of finding an optimal solution for the P-TaxiMP using the instances with 300 requests, an upper bound is shown by computing the uncapacitated preemptive TaxiMP (marked in the tables by (UB)), in order to get this upper bound we fixed a time limit of four hours for the solver while solving the LP corresponding to the uncapacitated preemptive TaxiMP). The results are summarized in Table 6.2 and Table 6.4.

REPLAN-NP (respectively hREPLAN-P) computes solutions for each replanning step within a short time, and a reasonable ratio w.r.t. the total number of accepted requests between the optimal offline solution OPT-NP and REPLAN-NP (respectively OPT-P and hREPLAN-P) in average around 53% for the first set and 45% for the second (respectively 49% for the first and 44% for the second). Note that hREPLAN-P takes more time to compute a solution for each replanning step, but in average the time is reasonable (less than 30 seconds).

In the first set of instances, we observe that the percentage of improvement between the optimal offline non-preemptive solution OPT-NP and the preemptive offline solution OPT-P is not very high (in average around 10%) and the percentage of improvement between the REPLAN-NP and hREPLAN-P is also not very high (in average around 13%) (see Table 6.3). The reason why there is no remarkable improvement is that the load of the requests is greater than Cap/2 most of the times. As in OPT-P and hREPLAN-P we allow vehicle preemption but not load preemption, therefore most of the times the requests cannot be accumulated together to be served.

In the second set of instances, we observe that the percentage of improvement between the optimal offline non-preemptive solution OPT-NP and the preemptive offline solution OPT-P increased. It is in average around 43% and the percentage of improvement between the REPLAN-NP and hREPLAN-P also increased in average around 74% (see Table 6.5). The reason why there is a remarkable improvement is that the load of the requests varies between 1 and 10. As in OPT-P and hREPLAN-P we allow vehicle preemption but not load preemption, therefore most of the times OPT-P and hREPLAN-P can serve more than one request simultaneously and therefore accept more requests than OPT-NP and REPLAN-NP.

Table 6.2: This table shows the computational results for the first set of 180 test instances of REPLAN-NP respectively hREPLAN-P in comparison to OFFLINE-NP entitled by OPT-NP respectively to the optimal preemptive offline solution OPT-P. The instances are grouped by the number of requests (1st column), the time horizon (2nd column) and the number of VIPAs (3rd column) with 30 instances per parameter set. Average values are shown for the total number $|\sigma_A|$ of accepted requests and for the total tour length TTL needed to serve the accepted requests. Finally, we provide the time needed to compute the optimal offline solution, the average runtime of REPLAN-NP respectively hREPLAN-P per recomputation step and the maximum runtime of the recomputation steps of REPLAN-NP respectively hREPLAN-P.

NP-TaxiMP										
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	OPT-NP	REPLAN-NP	gap %	OPT-NP	REPLAN-NP	OPT-NP	AVG	MAX
94	180	10	77	39	49,35	667,5	424	11,9	0,49	1,6
188	180	10	112	55	50,9	831	580	151	3	10,83
295	180	10	146,86	75,85	48,35	1005	750,57	76867,86	13,56	45,54
97	240	5	62,04	25,19	59,4	527,16	298,82	1650,94	0,29	1,23
194	240	5	93,76	45,84	51,1	680,44	490	229,76	1,8	7,85
290	240	5	115,94	47,64	58,9	759,94	500,6	121985,32	7,18	29,8
P-TaxiMP										
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	UB	hREPLAN-P	gap %	OPT-P	hREPLAN-P	OPT-P	AVG	MAX
94	180	10	82,8	41,8	46,78	658,7	406,74	2543,76	2,06	9,62
188	180	10	121,08	59,12	48,82	835,3	540,76	85732,2	6,47	38,63
295	180	10	160	90,8	62,78	1067,8	762,4	89547(UB)	21,7	68,4
97	240	5	66,48	29,7	44,67	514,65	267,75	5469,5	1,22	12,92
194	240	5	104,34	51,2	49,52	705,42	501,5	115678,6	3,79	24,4
290	240	5	129	54,4	43,44	792,75	496,6	132576(UB)	19,5	58,25

Table 6.3: This table shows the percentage of improvement of the average number of accepted requests between the non-preemptive and preemptive optimal solutions and between REPLAN-NP and hREPLAN-P for the first set of instances.

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

Table 6.4: This table shows the computational results for the second set of instances.

NP-TaxiMP										
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	OPT-NP	REPLAN-NP	gap %	OPT-NP	REPLAN-NP	OPT-NP	AVG	MAX
94	180	10	65,31	36,54	55,95	667,5	416,7	12,6	0,68	5,32
188	180	10	107,48	47,16	43,888	831	596,35	181,23	2,69	12,45
295	180	10	153,2	79,14	51,66	1005	726,86	73456,5	12,67	27,42
97	240	5	61,76	24,1	39,02	527,16	279,15	697,75	0,86	6,45
194	240	5	100,32	45,38	45,24	680,44	504,7	846,5	3,7	14,6
290	240	5	123,67	46,21	37,37	759,94	527,45	116875,85	14,1	22,46
P-TaxiMP										
			$\|\sigma_A\|$			TTL		runtime (s)		
req	T	k	UB	REPLAN-P	gap %	UB	REPLAN-P	UB	AVG	MAX
94	180	10	86,70	47,54	54,83	727,56	460,16	43824,50	2,58	11,58
188	180	10	158,65	77,80	49,04	930,75	649,67	125849,75	7,42	45,45
295	180	10	283(UB)	124,10	43,77	1175,25	878,25	97849(UB)	26,45	82,42
97	240	5	84,40	32,50	38,51	558,45	358,75	90470,67	1,36	14,36
194	240	5	154,23	72,67	47,12	825,74	609,40	156752,58	4,26	27,42
290	240	5	275(UB)	88,50	32,13	957,52	630,86	128417(UB)	21,78	65,80

Table 6.5: This table shows the percentage of improvement of the average number of accepted requests between OPT-NP and OPT-P and between REPLAN-NP and hREPLAN-P for the second set of instances.

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

6.5 Conclusion

Regarding the quality of the solutions obtained by REPLAN-NP, we summarize from this chapter that

- in theory, REPLAN-NP is not competitive since there is no finite c s.t. for all instances σ we have that REPLAN-NP$(\sigma) \geq c$ OPT(σ), but
- in practice, REPLAN-NP provides solutions of reasonably quality within short time for each recomputation step, see again Table 2.4.
- in theory, hREPLAN-P is also not competitive since there is no finite c s.t. for all instances σ we have that REPLAN-NP$(\sigma) \geq c$ OPT(σ), but
- in practice, hREPLAN-P leads to a higher rate of accepted requests and, therefore, to a higher quality-of-service level for the fleet management than REPLAN-NP. However, sometimes the transportation schedule contains preemptive tours.

Therefore, in order to handle the Online Taxi Mode Problem in the studied VIPAFLEET management system it is up to the operator to decide whether it is worth to have preemptive tours in order to increase the number of accepted requests.

Chapter 7

Conclusion

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 work w.r.t. different objective functions in Table 7.1.

Table 7.1: This table shows the competitive ratios obtained in this work. For each case, we show the competitive ratio for the algorithm on which type of graphs, using which mode w.r.t. which objective function. In some cases only lower bounds LB are shown.

Mode	metric space	Algorithm	General	Lunch	Morning	Evening
		w.r.t. minimizing the total tour length				
$TramMP$	Circuit or Line	SIR	$\text{Cap} \cdot \lvert C \rvert$	2Cap	Cap	Cap
$TramMP$	Circuit or Line	SIF_M	-	-	1	-
$TramMP$	Circuit or Line	SIF_E	-	-	-	1
$TramMP$	Circuit or Line	SIF_L	-	2	-	-
EMP	Line	MAIN	$2\text{Cap} \cdot \lvert L \rvert$	2Cap	Cap	Cap
		w.r.t. minimizing the total number of stops				
$TramMP$	Circuit or Line	SIR	$2 \cdot \text{Cap}$	$\frac{4}{3} \cdot \text{Cap}$	Cap	Cap
$TramMP$	Circuit or Line	SIF_M	-	-	$\frac{\text{Cap}+1}{2}$	-
$TramMP$	Circuit or Line	SIF_E	-	-	-	$\frac{\text{Cap}+1}{2}$
$TramMP$	Circuit or Line	SIF_L	-	$\text{Cap}+1$	-	-
EMP	Line	MAIN	2Cap	$\frac{4}{3} \cdot \text{Cap}$	Cap	Cap
		w.r.t. minimizing the makespan				
$TramMP$	Circuit or Line	SIR	$LB = 2$	$LB = 2$	$LB = 2$	$LB = 2$
$TramMP$	Circuit or Line	SIF_M	-	-	$LB = \frac{3}{2}$	-
$TramMP$	Circuit or Line	SIF_E	-	-	-	$LB = 2$
$TramMP$	Circuit or Line	SIF_L	-	$LB = 2$	-	-
EMP	Line	MAIN	$LB = 2$	$LB = 2$	2	2
		w.r.t. minimizing the total waiting time				
$TramMP$	Circuit or Line	SIR	∞			
$TramMP$	Circuit or Line	SIF_M	∞			
$TramMP$	Circuit or Line	SIF_E	∞			
$TramMP$	Circuit or Line	SIF_L	∞			
EMP	Line	MAIN	∞			
w.r.t. maximizing the number of accepted requests (serving them with minimum costs)						
NP-$TaxiMP$	general metric space	REPLAN-NP	∞			
P-$TaxiMP$	general metric space	hREPLAN-P	∞			

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 and all buildings are covered, 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) and all buildings are covered, one or several VIPAs on each circuit operating in tram mode using SIF_L.
- In general periods: 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.
- In "important periods": consider the whole graph where a fleet of VIPAs transport "VIPs" from their requested origins to their requested stations. Depending on the loads per request, the VIPAs operate on the network using REPLAN-P or hREPLAN-NP with the constraint that only one VIPA is allowed per arc to avoid their meetings.
- In "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,
  - consider one Hamilton cycle as shown in Figure 3.6, 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) using SIR.

Regarding the quality of the solutions obtained by the algorithms proposed for the TramMP (SIF_M, SIF_E, SIF_L, and SIR) we summarize from Chapter 4 that

- during the morning, SIF_M is suitable to apply w.r.t. all the different objective functions.
- during the evening, SIF_E is suitable to apply w.r.t. all the different objective functions.

- during the lunch, SIF_L is suitable to apply w.r.t. all the different objective functions.
- SIR is convenient to be applied in case of emergencies so that the operator does not need to take any decision.

The future works are to take into consideration the case of VIPAs running on circuit and a VIPA can overtake another one but only when arriving at stations. The circuit C may be designed to have the main arcs between stations, where overtaking is forbidden but on each station there is a loading/an unloading area, where a VIPA leaves the main track to load or unload users and does not block the other VIPAs that may pass by this station and they do not want to load or unload passengers.

Regarding the quality of the solutions obtained by the algorithm proposed for the EMP ($MAIN$) we summarize from Chapter 5 that

- MAIN leads to good ratios and reasonable solutions. Despite that MAIN can be used with only one VIPA operating on the line, this algorithm can serve more requests than SIR for the TramMP. By using MAIN in general case we may improve the quality of service.

Regarding the quality of the solutions obtained by REPLAN-NP, we summarize from Chapter 6 that

- in theory, REPLAN-NP is not competitive since there is no finite c, s.t., for all instances σ, we have that REPLAN-NP(σ) $\geq c$ OPT(σ), but
- in practice, REPLAN-NP provides solutions of reasonably quality within short time for each recomputation step, see again Tables 6.2-6.5
- in theory, hREPLAN-P is also not competitive since there is no finite c, s.t., for all instances σ, we have that REPLAN-NP(σ) $\geq c$ OPT(σ), but
- in practice, hREPLAN-P leads to a higher rate of accepted requests and, therefore, to a higher quality-of-service level for the fleet management than REPLAN-NP. However, sometimes the transportation schedule contains preemptive tours, therefore it is up to the operator to decide whether it is worth to have preemptive tours in order to increase the number of accepted requests.

We can conclude that the proposed replan strategies are promising algorithms to handle the Online NP-TaxiMP and the Online P-TaxiMP in the studied VIPAFLEET management system. FLOW-HEURISTIC is a promising heuristic that may be used not only to solve the Online P-TaxiMP but also its static version. As a perspective, we intend to apply this heuristic to static transportation problems (PDPTW and DARPTW) and to production and logistic problems such as Resource Constrained Project Scheduling Problems with convenient updates.

The main task of the VIPAfleet management is to find a good global strategy for processing the incoming requests. Hereby, the following questions have to be addressed:

- How often should the design of network be changed?
- Which algorithm we use for each mode and which subnetwork?
- Is it convenient to allow preemption? (i.e., the passenger needs to change VIPA at a certain station to be delivered to his destination?)

The online character of the problem and at the same time its main challenge comes from the impossibility to predict, where and when new requests in the (near) future will take place.

Given a "snapshot" of the situation at some moment in time, the Global VIPAFleet management problem consists in computing a transportation schedule for serving all requests with the available VIPAs w.r.t. a certain objective function.

If we take an example of one of the presented scenarios (lunch time on Michelin network), and we try to apply one algorithm on the whole network and apply the proposed algorithms SIF_L and $MAIN$ on subnetworks resulting from a partitionning of the network such that each building is reached via a shortest path from the restaurant and compare the transportation schedules obtained.

We clearly note that partitioning the network is beneficial on many levels:

- it is easier to handle the meeting of two vehicles.
- it provides better solutions in average if we know the pattern of requests in advance.
- it is easier for the VIPA to operate on a line or on a circuit than on the whole network.

Actually, we propose a design of network for each scenario with suitable algorithms for each subnetwork. The real challenge is to solve the partitioning problem behind this global fleet management problem. We know in advance that there is no hope to find theoretical results in terms of competitiveness on the global optimality because there are two much constraints that we need to take into consideration from the practical sense. However, an interesting idea would be to find a convenient partitioning heuristic and provide some analysis in practice via simulations to determine the best or the partition that gives a global good solution.

Bibliography

[1] Reducing emissions from transport. `https://ec.europa.eu/clima/policies/transport_en`, address = Nuremberg, note = Accessed: 2017-06-14, 2017.

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

[3] Luca Allulli, Giorgio Ausiello, and Luigi Laura. On the power of lookahead in on-line vehicle routing problems. In COCOON, pages 728–736. Springer, 2005.

[4] Jardar Andersen, Marielle Christiansen, Teodor Gabriel Crainic, and Roar Grønhaug. Branch and price for service network design with asset management constraints. Transportation Science, 45(1):33–49, 2011.

[5] M. Antosiewicz, G. Koloch, and B. Kamiński. Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed. Journal of Theoretical and Applied Computer Science, 7:46–55, 2013.

[6] David L Applegate, Robert E Bixby, Vašek Chvátal, William Cook, Daniel G Espinoza, Marcos Goycoolea, and Keld Helsgaun. Certification of an optimal tsp tour through 85,900 cities. Operations Research Letters, 37(1):11–15, 2009.

[7] Jay E Aronson. A survey of dynamic network flows. Annals of Operations Research, 20(1):1–66, 1989.

[8] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. J. ACM, 45(5):753–782, September 1998.

[9] N. Ascheuer, S. O. Krumke, and J. Rambau. Competitive scheduling of elevators. Preprint SC 98-34, November 1998.

[10] 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.

[11] A Assad, M Ball, L Bodin, and B Golden. Routing and scheduling of vehicles and crews. Computers & Operations Research, 10(2):63–211, 1983.

[12] Mikhail J Atallah and S Rao Kosaraju. Efficient solutions to some transportation problems with applications to minimizing robot arm travel. SIAM Journal on Computing, 17(5):849–869, 1988.

[13] Giorgio Ausiello, Luca Allulli, Vincenzo Bonifaci, and Luigi Laura. On-line algorithms, real time, the virtue of laziness, and the power of clairvoyance. Lecture notes in computer science, 3959:1, 2006.

[14] Giorgio Ausiello, Vincenzo Bonifaci, and Luigi Laura. The on-line asymmetric traveling salesman problem. In WADS, pages 306–317. Springer, 2005.

[15] Giorgio Ausiello, Marc Demange, Luigi Laura, and Vangelis Paschos. Algorithms for the on-line quota traveling salesman problem. In Computing and Combinatorics, pages 290–299. Springer, 2004.

[16] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Serving requests with on-line routing. Algorithm Theory—SWAT'94, pages 37–48, 1994.

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

[18] Kay W Axhausen. Social networks, mobility biographies, and travel: survey challenges. Environment and Planning B: Planning and design, 35(6):981–996, 2008.

[19] Barrie M. Baker and M.A. Ayechew. A genetic algorithm for the vehicle routing problem. Computers & Operations Research, 30(5):787 – 800, 2003.

[20] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In FOCS, pages 267–276, 2011.

[21] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Metrical task systems and the k-server problem on HSTs. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, Automata, Languages and Programming, volume 6198 of Lecture Notes in Computer Science, pages 287–298. Springer Berlin Heidelberg, 2010.

[22] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, pages 711–719, New York, NY, USA, 1997. ACM.

[23] Hannah Bast, Daniel Delling, Andrew V Goldberg, and Matthias Müller. Hannemann, thomas pajor, peter sanders, dorothea wagner, and renato f. werneck. route planning in transportation networks. Technical report, Technical Report MSR-TR-2014-4, Microsoft Research, 2014.

[24] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. Top, 15(1):1–31, 2007.

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

[26] Kristin Berg Bergvinsdottir, Jesper Larsen, and Rene Munk Jørgensen. Solving the Dial-a-Ride Problem using Genetic algorithms. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[27] Norman Biggs, E Keith Lloyd, and Robin J Wilson. Graph Theory, 1736-1936. Oxford University Press, 1976.

[28] Antje Bjelde, Yann Disser, Jan Hackfeld, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Kevin Schewior, Miriam Schlöter, and Leen Stougie. Tight bounds for online tsp on the line. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 994–1005. SIAM, 2017.

[29] Michiel Blom, Sven O Krumke, Willem E de Paepe, and Leen Stougie. The online TSP against fair adversaries. INFORMS Journal on Computing, 13(2):138–148, 2001.

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

[31] Natashia Boland, Mike Hewitt, Duc Minh Vu, and Martin Savelsbergh. Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks. In International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pages 254–262. Springer, 2017.

[32] Vincenzo Bonifaci, Leen Stougie, et al. Online k-server routing problems. Lecture Notes in Computer Science, 4368:83–94, 2006.

[33] Allan Borodin and Ran El-Yaniv. Online algorithms and competitive analysis, 1998.

[34] Burak Boyacı, Konstantinos G Zografos, and Nikolas Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. European Journal of Operational Research, 240(3):718–733, 2015.

[35] Rainer E. Burkard, Vladimir G. Deineko, and Gerhard J. Woeginger. The travelling salesman and the PQ-tree. In William H. Cunningham, S. Thomas McCormick, and Maurice Queyranne, editors, Integer Programming and Combinatorial Optimization, volume 1084 of Lecture Notes in Computer Science, pages 490–504. Springer Berlin Heidelberg, 1996.

[36] Roberto Wolfler Calvo and Nora Touati Moungla. A matheuristic for the dial-a-ride problem. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, INOC, Lecture Notes in Computer Science, pages 450–463. Springer, 2011.

[37] Yaw Chang and Lin Chen. Solve the vehicle routing problem with time windows via a genetic algorithm. Descrete and continous dynamical systems supplement, pages 240–249, September 2007.

[38] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming, 20(1):255–282, 1981.

[39] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. In SIAM Journal on Discrete Mathematics, pages 291–300, 1990.

[40] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k servers on trees. SIAM Journal on Computing, 20:144–148, 1996.

[41] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4):568–581, 1964.

[42] Leandro C. Coelho and Gilbert Laporte. The exact solution of several classes of inventory-routing problems. Computers & Operations Research, 40(2):558 – 565, 2013.

[43] Don Coppersmith, Peter Doyle, Prabhakar Raghavan, and Marc Snir. Random walks on weighted graphs, and applications to on-line algorithms (extended. In Journal of the ACM, pages 369–378, 1990.

[44] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research, 54(3):573–586, 2006.

[45] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(2):89–101, 2003.

[46] Jean-François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological, 37(6):579–594, 2003.

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

[48] Aaron Coté, Adam Meyerson, and Laura Poplawski. Randomized k-server on hierarchical binary trees. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC '08, pages 227–234, New York, NY, USA, 2008. ACM.

[49] Teodor Gabriel Crainic, Mike Hewitt, Michel Toulouse, and Duc Minh Vu. Service network design with resource constraints. Transportation Science, 50(4):1380–1393, 2014.

[50] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. Management Science, 6(1):80–91, 1959.

[51] Alexa Delbosc. The role of well-being in transport policy. Transport Policy, 23:25–33, 2012.

[52] Samuel Deleplanque and Alain Quilliot. Insertion techniques and constraint propagation for the darp. In FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS-WCO 2012, volume 2012, pages 393–400, 2012.

[53] Samuel Deleplanque and Alain Quilliot. Robustness tools in dynamic dial-a-ride problems. In Recent Advances in Computational Optimization - Results of the Workshop on Computational Optimization WCO 2013 [FedCSIS 2013, Kraków, Poland], pages 35–51, 2013.

[54] 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.

[55] Xiaotie Deng and Sanjeev Mahajan. Server problems and resistive spaces. Information Processing Letters, 37(4):193 – 196, 1991.

[56] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research, 40(2):342–354, 1992.

[57] Alan Erera, Michael Hewitt, Martin Savelsbergh, and Yang Zhang. Improved load plan design through integer programming based local search. Transportation Science, 47(3):412–427, 2013.

[58] 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.

[59] Anke Fabri and Peter Recht. Online dial-a-ride problem with time windows: An exact algorithm using status vectors. In Karl-Heinz Waldmann and UlrikeM. Stocker, editors, Operations Research Proceedings 2006, volume 2006 of

Operations Research Proceedings, pages 445–450. Springer Berlin Heidelberg, 2007.

[60] Esteban Feuerstein and Leen Stougie. On-line single-server dial-a-ride problems. Theoretical Computer Science, 268(1):91–105, 2001.

[61] Marshall L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. Operations Research, 42(4):626–642, 1994.

[62] LR Ford and DR Fulkerson. Flows in networks princeton university press. Princeton, New Jersey, 276, 1962.

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

[64] Greg N. Frederickson and Dih Jiun Guan. Nonpreemptive ensemble motion planning on a tree. Journal of Algorithms, 15(1):29–60, 1993.

[65] Xiaobing Gan, Yan Wang, Shuhai Li, and Ben Niu. Vehicle routing problem with time windows and simultaneous delivery and pick-up service based on mcpso. Mathematical Problems in Engineering, vol. 2012, 2012.

[66] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. Management Science, 40(10):1276–1290, October 1994.

[67] Paul C Gilmore and Ralph E Gomory. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. Operations research, 12(5):655–679, 1964.

[68] Fatma Pinar Goksal, Ismail Karaoglan, and Fulya Altiparmak. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. Computers and Industrial Engineering, 65(1):39–53, May 2013.

[69] Martin Charles Golumbic. Algorithmic graph theory and perfect graphs, volume 57. Elsevier, 2004.

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

[71] Dih Jiun Guan. Routing a vehicle of capacity greater than one. Discrete Applied Mathematics, 81(1-3):41–57, 1998.

[72] Joachim Gudmundsson and Christos Levcopoulos. A fast approximation algorithm for TSP with neighborhoods and red-blue separation. In Takano Asano, Hideki Imai, D.T. Lee, Shin-ichi Nakano, and Takeshi Tokuyama, editors, Computing and Combinatorics, volume 1627 of Lecture Notes in Computer Science, pages 473–482. Springer Berlin Heidelberg, 1999.

[73] Gregory Gutin and Anders Yeo. Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number. Discrete Applied Mathematics, 119(1–2):107 – 116, 2002. Special Issue devoted to Foundation of Heuristics in Combinatoria l Optimization.

[74] Gregory Gutin, Anders Yeo, and Alexey Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Applied Mathematics, 117(1–3):81 – 86, 2002.

[75] Alex Hall, Steffen Hippler, and Martin Skutella. Multicommodity flows over time: Efficient algorithms and complexity. Theoretical Computer Science, 379(3):387–404, 2007.

[76] Dorit S Hochbaum and Dan Landy. Scheduling semiconductor burn-in operations to minimize total flowtime. Operations research, 45(6):874–885, 1997.

[77] AJ Hoffman and JB Kruskal. Linear inequalities and related systems. Annals of Mathematics Studies, pages 223–246, 1956.

[78] Saman Hong. A linear programming approach for the traveling salesman problem. 1973.

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

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

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

[82] Brady Hunsaker and Martin Savelsbergh. Efficient feasibility testing for dial-a-ride problems. Operations Research Letters, 30(3):169 – 173, 2002.

[83] Stefan Irnich, Paolo Toth, and Daniele Vigo. The family of vehicle routing problems. Vehicle Routing: Problems, Methods, and Applications, 18:1–36, 2014.

[84] P. Jaillet and M. Wagner. Online routing problems: value of advanced information as improved competitive ratios. Transportation Science, 40(2), 2006.

[85] Patrick Jaillet and Michael R Wagner. Online vehicle routing problems: A survey. The Vehicle Routing Problem: Latest Advances and New Challenges, pages 221–237, 2008.

[86] Siddhartha Jain and Pascal Van Hentenryck. Large neighborhood search for dial-a-ride problems. In Jimmy Lee, editor, Principles and Practice of Constraint Programming – CP 2011, volume 6876 of Lecture Notes in Computer Science, pages 400–413. Springer Berlin Heidelberg, 2011.

[87] Ahmad I Jarrah, Ellis Johnson, and Lucas C Neubert. Large-scale, less-than-truckload service network design. Operations Research, 57(3):609–625, 2009.

[88] Nicolas Jozefowiez, Fred Glover, and Manuel Laguna. Multi-objective meta-heuristics for the traveling salesman problem with profits. Journal of Mathematical Modelling and Algorithms, 7(2):177–195, 2008.

[89] Richard M Karp. Reducibility among combinatorial problems. In Complexity of computer computations, pages 85–103. Springer, 1972.

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

[91] Elias Koutsoupias and Christos Papadimitriou. On the k-server conjecture. Journal of the ACM, 42:507–511, 1995.

[92] S. O. Krumke, L. Laura, M. Lipmann, A. Marchetti-Spaccamela, W. E. de Paepe, D. Poensgen, and L. Stougie. Non-abusiveness helps: An $O(1)$-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In Proceedings of the 5th, volume 2462, pages 200–214. Springer, 2002.

[93] S. O. Krumke, A. Quilliot, A. Wagler, and J.-Th. Wegener. Models and algorithms for carsharing systems and related problems. In Proceedings of the VII Latin-American Algorithms, Graphs, and Optimization Symposium, 2013.

[94] Sven O Krumke. Online optimization: Competitive analysis and beyond. ZIB, 2006.

[95] Sven O Krumke, Alain Quilliot, Annegret K Wagler, and Jan-Thierry Wegener. Relocation in carsharing systems using flows in time-expanded networks. In International Symposium on Experimental Algorithms, pages 87–98. Springer, 2014.

[96] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. European journal of operational research, 59(3):345–358, 1992.

[97] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. North-Holland Mathematics Studies, 132:147–184, 1987.

[98] Hoong Chuin Lau, Melvyn Sim, and Kwong Meng Teo. Vehicle routing problem with time windows and a limited number of vehicles. European Journal of Operational Research, 148(3):559 – 569, 2003.

[99] M Lipmann. The online traveling salesman problem on the line. Master's thesis, Department of Operations Research, University of Amsterdam, The Netherlands, 1999.

[100] Ying Luo and Paul Schonfeld. A rejected-reinsertion heuristic for the static dial-a-ride problem. Transportation Research Part B: Methodological, 41(7):736–755, 2007.

[101] Thomas L Magnanti. Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. Networks, 11(2):179–213, 1981.

[102] Mark Manasse, Lyle McGeoch, and Daniel Sleator. Competitive algorithms for on-line problems. In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC '88, pages 322–333, New York, NY, USA, 1988. ACM.

[103] Karl Menger. Das botenproblem. Ergebnisse eines mathematischen kolloquiums, 2:11–12, 1932.

[104] Stephan Olariu. An optimal greedy heuristic to color interval graphs. Information Processing Letters, 37(1):21–25, 1991.

[105] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM review, 33(1):60–100, 1991.

[106] Christos H Papadimitriou. The euclidean travelling salesman problem is np-complete. Theoretical Computer Science, 4(3):237–244, 1977.

[107] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft, 58(2):81–117, 2008.

[108] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems. Journal für Betriebswirtschaft, 58(1):21–51, 2008.

[109] Victor Pimenta, Alain Quilliot, Hélène Toussaint, and Daniele Vigo. Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles. European Journal of Operational Research, 257(2):601–613, 2017.

[110] Warren B Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. Handbooks in operations research and management science, 8:141–295, 1995.

[111] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. The geography of transport systems. Taylor & Francis, 2016.

[112] 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.

[113] 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.

[114] L Rudolph, AR Karlin, MS Manasse, and DD Sleator. On-line file caching. In Proc. 9th ACM–SIAM SODA, pages 82–86, 1988.

[115] Martin Skutella. An introduction to network flows over time. In Research trends in combinatorial optimization, pages 451–482. Springer, 2009.

[116] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. Communications of the ACM, 28(2):202–208, 1985.

[117] K.C. Tan, L.H. Lee, and K. Ou. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. Engineering Applications of Artificial Intelligence, 14(6):825 – 837, 2001.

[118] K.C. Tan, L.H. Lee, Q.L. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. Artificial Intelligence in Engineering, 15(3):281 – 295, 2001.

[119] Paolo Toth and Daniele Vigo. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science, 31(1):60–71, 1997.

[120] Annegret Wagler. Decisions under uncertainty II. ISIMA, September 2016.

[121] E Willem. de paepe, jan karel lenstra, jiri sgall, rené a. sitters, leen stougie, computer-aided complexity classification of dial-a-ride problems. INFORMS Journal on Computing, 16(2):120–132, 2004.

[122] James Woodcock, Phil Edwards, Cathryn Tonne, Ben G Armstrong, Olu Ashiru, David Banister, Sean Beevers, Zaid Chalabi, Zohir Chowdhury, Aaron Cohen, et al. Public health benefits of strategies to reduce greenhouse-gas emissions: urban land transport. The Lancet, 374(9705):1930–1943, 2009.

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

[124] Fanglei Yi and Lei Tian. On the online dial-a-ride problem with time-windows. In Algorithmic Applications in Management: First International Conference, AAIM 2005, volume 3521 of LNCS, pages 85–94. Springer-Verlag Berlin Heidelberg, 2005.

[125] Alkın Yurtkuran and Erdal Emel. A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems. Expert Syst. Appl., 37(4):3427–3433, April 2010.