Applied Methods for the Vehicle Positioning Problem

vorgelegt von

M.Sc. Carlos Henrique Cardonha

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Tag der wissenschaftlichen Aussprache: 19. Oktober 2011

Berlin 2012
D 83

Table of Contents

List of Figures

List of Tables

List of Algorithms

to Berlin.

Abstract

This dissertation is dedicated to the Vehicle Positioning Problem (VPP), a classical combinatorial optimization problem in public transport in which vehicles should be assigned to parking positions in a depot in such a way that shunting moves are minimized. We investigate several models and solution methods to solve the VPP and the VPP^P, a multi-periodic extension of the problem which was not previously studied.

In the first part of the thesis, the basic version of the problem is introduced and several formulations, theoretical properties, and concepts are investigated. In particular, we propose a mixed integer quadratic constrained formulation of the VPP whose QP relaxation produces the first known non-trivial lower bound on the number of shunting moves.

The second part of our work describes two advanced solution methods. In the first approach, a set partitioning formulation is solved by a branch-and-price framework. We present efficient algorithms for the pricing problem and in order to improve the performance of the framework, we introduce heuristics and discuss strategies to reduce symmetry. The second approach consists of an iterative technique in which we try to optimize an ILP by solving some of its projections, which are smaller and therefore easier to compute. Both techniques are able to produce satisfactory solutions for large-scale instances of the VPP^P.

In the third part, advanced aspects of the problem are investigated. We propose and analyze several solution methods for the VPP$_+$ and for the VPP$_+^P$, which are extended and more challenging versions of the VPP and of the VPP^P, respectively. Finally, the role of uncertainty in the problem is discussed. In particular, we introduce a new criteria to evaluate the robustness of assignment plans, a formulation based on this concept, and a new online algorithm for the VPP.

Zusammenfassung

Diese Dissertation beschäftigt sich mit dem Vehicle Positioning Problem (VPP), ein klassisches Problem der Kombinatorischen Optimierung, das sich mit den optimalen Zuweisungen von Fahrzeugen öffentlicher Verkehrsunternehmen zu Umläufe und zu Stellplätzen befasst. Wenn ein nicht zielorientierter Plan verwendet wird, sind häufig Umpositionierungen von Fahrzeugen erforderlich. Ziel des Optimierungsansatzes ist es, diese zu minimieren. In dieser Arbeit werden mehrere Modelle und Methoden vorgestellt, um das VPP und das Multi-Periodic Vehicle Positioning Problem (VPP^P), eine bisher nicht untersuchte multiperiodische Erweiterung des VPP, zu lösen.

Im ersten Teil der Arbeit werden das Problem aufgeführt sowie mehrere Formulierungen, theoretische Eigenschaften und Konzepte untersucht. Es wird eine gemischt-ganzzahlige quadratische Formulierung des Problems vorgestellt, deren QP Relaxierung nicht-triviale untere Schranken für das VPP erzeugen kann.

Der zweite Teil der Arbeit beschreibt zwei spezielle Verfahren. In dem ersten Ansatz wird eine Set Partitioning Formulierung mit einem Branch-and-Price Verfahren gelöst. Für die Pricing Probleme werden effiziente Algorithmen beschrieben. Um die Leistung des Verfahrens zu verbessern, wurden Heuristiken entwickelt und Strategien zur Reduktion der Symmetrie des Problems analysiert. Der zweite Ansatz ist eine iterative Technik, die ein gemischt-ganzzahliges Problem durch die Lösung einiger ihrer Projektionen optimiert. Beide Techniken sind in der Lage, Lösungen für große Instanzen vom VPP^P zu erzeugen, die wenige Umpositionierungen benötigen.

Im dritten Teil werden vertiefende Aspekte des Problems untersucht. Vorgestellt werden die Analyse verschiedener Lösungswege für das VPP$_+$ und für das VPP$^P_+$, die erweiterte und schwierigere Versionen des VPP bzw. VPP^P sind. Abschließend wird ein neues Konzept diskutiert, mit dem die Robustheit eines Planes ausgewertet wird. Es werden eine auf diesem Konzept basierende robuste Formulierung vorgestellt, und zuletzt ein neuer Online-Algorithmus für das VPP.

Acknowledgment

I would like to thank Prof. Dr. Dr. h.c. mult. Martin Grötschel for giving me the opportunity to work in his research group at ZIB with such an interesting and challenging problem. I am also very grateful to PD Dr. habil. Ralf Borndörfer, with whom I had the opportunity to work during my doctoral studies and whose support and suggestions were indispensable.

Other people were also important for the development of this work. In particular, Nam Dũng Hoàng, Ivan Dovica, and Fernando Mario de Oliveira, who read early versions of my thesis and gave very useful suggestions; Anke Fröbel and Elmar Swarat, who reviewed and corrected the "Zusammenfassung"; Bettina Kasse, who helped me a lot with bureaucratic issues; and Annie Raymond, who prepared a very nice buffet for my defense. Also essential were CNPq-DAAD and the Zuse Institute Berlin, who provided the financial support for this project.

Without the assistance and the encouragement given by Yoshiko Wakabayashi, Cristina Gomes Fernandes, and Carlos Eduardo Ferreira, from IME-USP, I would not have come to Germany, so they were also important and I am very grateful to them.

It was a pleasure to spend these years working at the Zuse Institute Berlin, and I am very thankful to all colleagues and staff that I met there. In particular, I would like to thank the 11:30 lunch-group for the very nice conversations and for the German lessons.

I met a lot of people during my doctoral studies who played (and still play) a significant role in my life. Thank you Francisco, Brand, Moisés, João Paulo, Napo, Miguel, Rodrigo, Mariela, Martin, Juliano, Dani, Gevson, Ednilson, Lorena, Guarani, Flaviano, Igor, Roberta, Luciane, Bob, Pawel, Irene, Evgenia, Nadya, Guillermo, Anke, Alexis, Diana, Jitka, Luzia, Laura, Thomas, Ivan, Tereza, and Dũng for so many good moments. My life in Germany would have been much harder without you.

It was nice to stay in touch with some good old friends as well, like Marcel, Marcelo, Fabricio, Ciré, Salem, Domingos, Daniel, and Fernando. Events like the night in White Trash with Salem, Domingos, Daniel, Fernando, and Guillermo and the clubbing night in Berghain with Ciré made these old connections even stronger.

Finally, I am who I am today mainly because of my family (in particular, my parents, Regina and José) and because of Berlin. The final result is suboptimal, but "das ist auch gut so", so I wish to express my sincere thanks and warm gratitude to them.

Chapter 1

Introduction

The Vehicle Positioning Problem (VPP) is about the assignment of vehicles of a transport company to parking positions in a depot. The problem is nontrivial because the parking lots used by these companies usually have rigid space constraints and because they have to deal with vehicles of different types. If bad assignment decisions are taken, a vehicle may be blocked by other units at the moment it is supposed to depart. Rearrangements are necessary to solve such issues, but these operations are costly and should be avoided. The designation of decent parking positions to vehicles is a central problem in depot management and a key to the smooth operation of a public transport or railway company (see Eisenberg & Prigge (2002) [32]).

The problem appears in different contexts. While bus positioning is basically about the order of vehicles in parking rows, tram and train positioning deal with rail-bound traffic, where operations with units are more restricted. Moreover, tram and train positioning may consider compositions of more than one vehicle, which might be split up or put together. There are several further aspects that can play a role in the Vehicle Positioning Problem, and as a consequence, the number of different scenarios is considerable. In his PhD thesis, Hansmann (2010) [54] presents an extensive classification of problems related to the VPP and claims that there are at least 100 different versions of real-world applications.

The Vehicle Positioning Problem can be seen as a subclass of the Stack Management Problem, in which a buffer consisting of a set of stacks receives and redistributes items, possibly subject to additional operational constraints. Other interesting problems of this type are railway and airline delay management, container stowage in harbors and ships, and high rack warehouse operation (see Günther & Kim (2005) [50] and Froyland et al. (2006) [38]).

It is interesting to observe that the Vehicle Positioning Problem has been studied almost exclusively by research groups in Europe (especially Germany, Holland, and Italy, while Canada is one of the few exceptions). This fact suggests that the problem does not receive similar attention in other locations. In particular, we can mention the cases of USA, where the public transport system in most cities

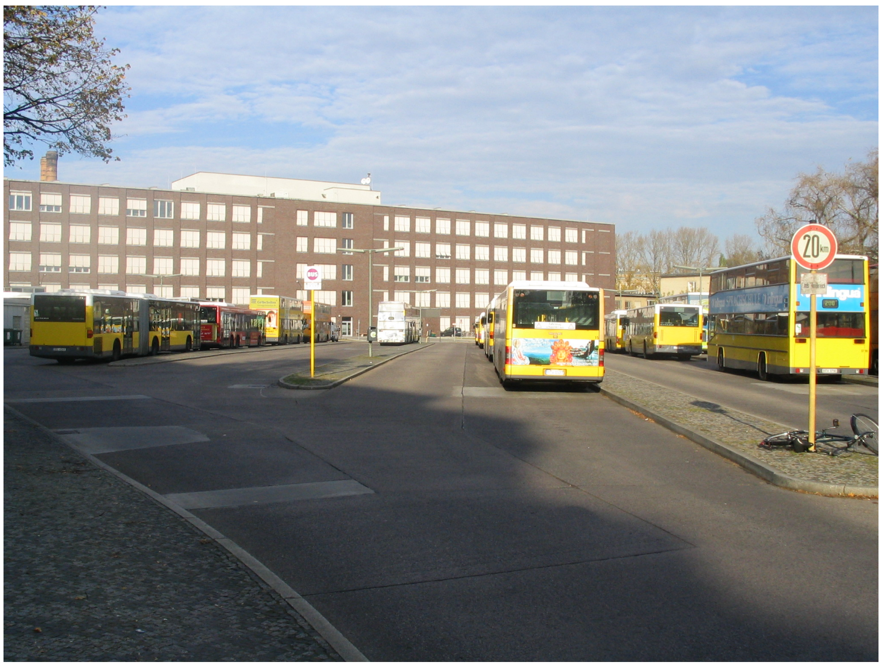


Figure 1: Depot in Zoologischer Garten, Berlin

is not as well-developed as in Europe, and of Latin America, where quality of service, good equipment, good planning, and good management are features that can hardly be found simultaneously.

1.1 The Basic Problem

We present now a more detailed description of the Vehicle Positioning Problem for buses, which is the main object of study of this thesis.

A bus company must have units of different types in order to provide satisfactory services for its list of scheduled trips. Single-deck vehicles, for example, are usually acceptable for less popular routes. Conversely, for itineraries with a large number of passengers, articulated buses may be more appropriate. Double-decker vehicles are also suitable for highly demanded trips, but their "cult status" may make their use in touristic routes commercially more interesting. In Figure 1, it is possible to see buses of different types parked in a depot close to the Zoologischer Garten Station, in Berlin. Because transport companies usually have several units of each type, *the assignment of vehicles to itineraries* is one of the tasks of a depot manager.

The vehicles of a transport company do not work uninterruptedly. Typically, in certain intervals of time (usually at night), there is no trip to be serviced, and during these periods the buses stay parked in the depot. Therefore, a depot manager must also *assign vehicles to parking positions* as soon as they complete their previ-

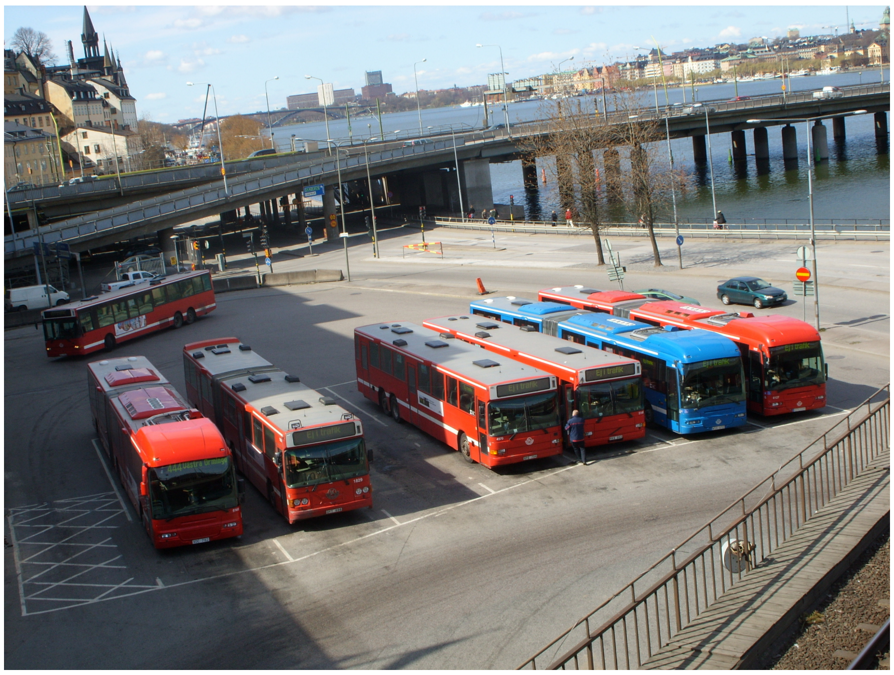


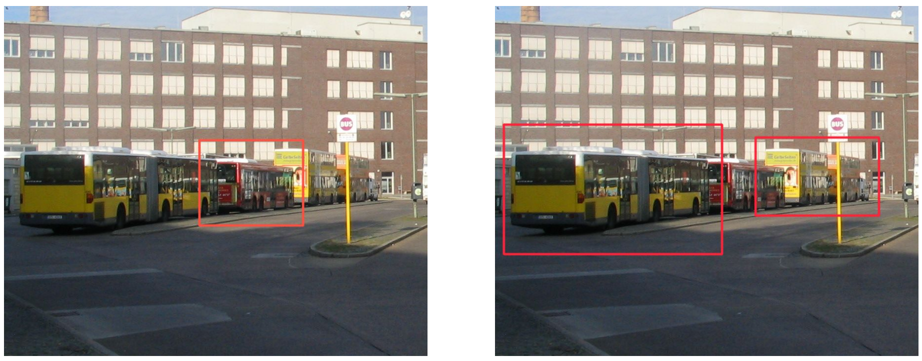
Figure 2: Parking lot for buses in Stockholm, Sweden

ously assigned itineraries. If the number of vehicles is small and/or if the depot is sufficiently big, the depot can work like a parking lot for cars, i.e., the arrival and the departure of one bus is not affected by the arrivals and the departures of the others. Figure 2 shows that such scenarios can be found in the real world.

If space restrictions make this trivial organization of the depot impossible, though, the problem becomes significantly more challenging. In these cases, parking positions usually form rows that can only be accessed at one of their two ends. Moreover, the space between two rows is so limited that it does not allow a bus to pass others parked in front of it. As a consequence, the assignment of vehicles to parking positions has a strong influence on the assignment of vehicles to trips in such scenarios. In particular, bad parkings may lead to situations where units must be rearranged. This happens if we have the situation depicted in Figure 3 at the moment that the red single-deck bus should depart, for example. In these cases, we say that a *shunting move* is necessary, which is an undesired operation whose number should be minimized.

While such severe space restrictions of bus depots may seem surprising, this situation is clearly unavoidable in rail-bound traffic, where the parking rows are track segments that must be operated as one or two-sided stacks or queues (see Figure 4 for an example of train depot).

A theoretical idealization of the problem is described in Figure 5, where we can see a sequence of six arriving buses, two tracks containing three parking positions each, and a sequence of six itineraries. Finally, the types of vehicles are represented



(a) Red bus has to depart

(b) Yellow buses are blocking both paths

Figure 3: Situation where a vehicle is blocked

Figure 4: Train depot in Panierai, Lithuania

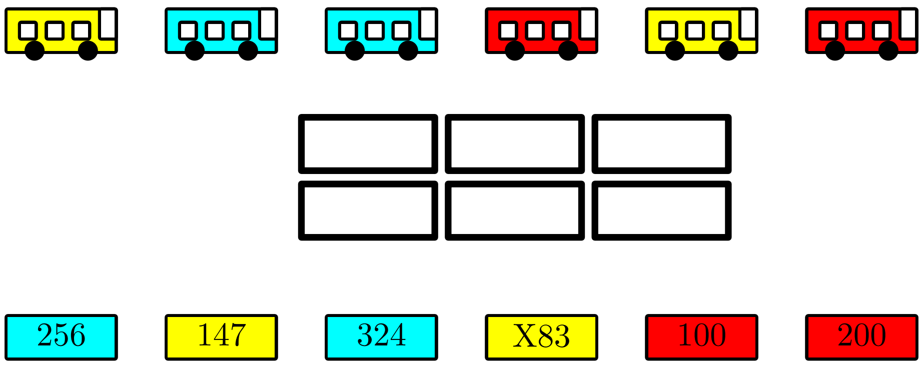
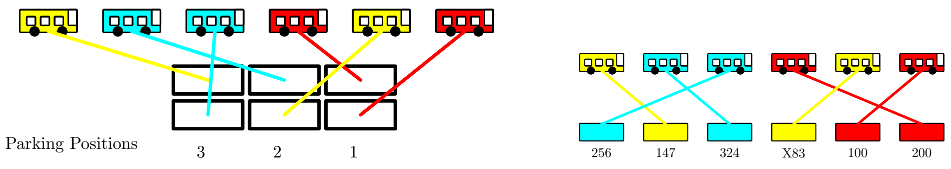


Figure 5: Example of theoretical idealization of the problem



(a) Assignment of buses to parking positions (b) Assignment of buses to trips

Figure 6: Feasible solution

by the colors. If we assume that the sequences are ordered from the left to the right, i.e., both the last arrival and the last departure are of type red, and that the vehicles arrive at the rightmost positions and depart from the leftmost ones, then the assignments depicted in Figure 6 represent a feasible solution for this instance.

The final product of a decision support software for depot management is an assignment plan similar to the one presented in Figure 7. Each line of the table represents a track, and each cell contains three informations (from top to bottom): the arrival time (in our example, the vehicles arrived in the previous day), the departure time, and the type of the vehicle parked in this position.

1.2 Formal Aspects

Based on discussions with PSI Trans, a German software company which is interested in the development of computational solutions for depot management, we identified the most important practical aspects of the Vehicle Positioning Problem. We present now a brief overview of these features in a more formal way.

Technical Details of the Basic Problem. In certain scenarios, there are itineraries that can be serviced by several types of vehicles. As we mentioned in the last section, for example, both articulated and double-decker vehicles may be

19:03:00h 15:21:00h T_1	18:15:00h 15:14:00h T_2	17:54:00h 15:11:00h T_1	17:34:00h 15:04:00h T_1	17:22:00h 11:26:00h T_1
18:54:00h 16:10:00h T_2	18:27:00h 12:47:00h T_2	18:15:00 12:45:00h T_1	17:27:00h 12:05:00h T_3	17:25:00h 11:10:00h T_4
19:23:00h 16:07:00h T_2	18:55:00h 15:44:00h T_2	18:54:00h 14:38:00h T_1	18:17:00h 14:08:00h T_2	18:03:00h 12:37:00h T_2
19:45:00h 16:41:00h T_4	19:13:00h 15:38:00h T_4	18:57:00h 11:46:00h T_4	18:47:00h 11:45:00h T_3	18:32:00h 11:39:00h T_3

Figure 7: Planned assignment

suitable for trips with a high number of passengers. However, in the basic version of the problem, we will assume that *each trip can only be serviced by one type of vehicle.*

There are works in the literature that do not incorporate some important real-world restrictions, as limitations on the number and on the size of tracks. Such simplifications lead to new problems with slightly different purposes. For example, the objective may be the minimization of the number of tracks used to park all the vehicles. Problems of this nature are interesting if one wants to plan the topology of a depot, but we assume here that the physical layout already exists and must be respected. Therefore, the *depots have a limited numbers of tracks* and *all the tracks have the same (finite) size* in every version of the problem that we investigate in this thesis.

Our goal is the *minimization of the number of shunting moves*, which is an estimation of the volume of work that has to be performed in a depot. Shuntings are traditionally evaluated as the number of pairs of vehicles parked in the same track that must have their relative order switched. We say that there is a *crossing* involving a pair of units in this situation. This estimation is reasonable, but we remark that it is not always completely accurate, as one change of positions between two vehicles may require as much work as several changes in the same track. For example, if we have the situation described in Figure 3, the articulated bus is the next to leave, and the vehicles are not allowed to go backwards in the track, then all the buses of the track will have to be moved in the shunting operation. Therefore, it may be more interesting to consider only crossings of vehicles parked in consecutive positions. We will refer to such events as *first crossings.*

Another important parameter is the operation policy of the tracks. A track is LIFO (or stack) if vehicles can only arrive and leave from the same (and typically unique) entrance, and it is FIFO (or queue) if vehicles arrive at one entrance and leave from the other. It is possible to say that a shunting will occur in a given track if its policy is not followed. We assume in this text that *all the tracks are FIFOs*. Other policies are discussed by Hansmann (2010) [54].

We also assume that *buses are independent atomic units*, i.e., that they can be neither composed nor decomposed, and that *the assignment of vehicles to trips is not fixed*.

A combinatorial description of the basic problem is presented below:

Definition 1.1. *The basic version of the* Vehicle Positioning Problem, *to which we refer as the* VPP, *is defined as follows:*

***Input**: We are given a sequence $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ of arrival times ($n \in \mathbb{N}$), a sequence $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$ of departure times, a set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t\}$ of vehicle types ($t \in \mathbb{N}$), a function $\tau : \{\mathcal{A} \cup \mathcal{D}\} \to \mathcal{T}$, and a set $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ of tracks ($m \in \mathbb{N}$), where each track represents a queue of β parking positions ($\beta \in \mathbb{N}, n = m\beta$). Sequence $\mathcal{A}$ ($\mathcal{D}$) is such that $a_i \leq a_j$ ($d_i \leq d_j$) if and only if $i \leq j$.*

***Task**: Determine a set M of triples $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ such that each element of $\mathcal{A}$ and $\mathcal{D}$ appears exactly once; each element of $\mathcal{S}$ appears exactly β times; (a_i, s_j, d_k) belongs to M only if $\tau(a_i) = \tau(d_k)$; and the number of pairs of elements $(a_i, s_j, d_k), (a_p, s_j, d_r)$ in M such that $i < p$ and $k > r$ is minimized.*

The practical objective of the problem is to find solutions without crossings, which is the trivial lower bound valid for every instance of the VPP, as they represent plans that do not require shuntings and, therefore, are operationally easier and cheaper to be implemented. The following fact is known about the computational complexity of the decision version of the VPP:

Theorem 1.2 (Winter (1998) [91]). *It is $\mathcal{NP}$-complete to decide if there is a solution for an instance of the* VPP *with at most k crossings for any fixed k.*

In order to prove this result, Winter reduced the Three-Dimensional Matching Problem, a classical combinatorial problem, to the VPP.

In this work, we investigate different models and approaches to the VPP and compare them from a theoretical and from a computational point of view. Our results show that it is not possible to obtain satisfactory solutions for large-scale instances of the problem just by formulating them as mixed integer programs and by solving them with generic commercial software unless an unacceptable amount of computation time is dedicate to these tasks. We also remark that we were able to design a mixed integer quadratic constrained program whose continuous relaxation always delivers non-trivial lower bounds for instances which require shuntings.

Finally, we will also consider a version of the Vehicle Positioning Problem where first crossings are to be minimized as VPP. In the cases where the distinction between these two versions is relevant, the correct interpretation is clearly indicated by the context.

Multiperiodiocity. A transport company usually has a schedule that describes the departure time of each trip and the type of vehicle it requires. Moreover, it is possible to assign groups of itineraries to *periods*, which are defined according to the weekday (Monday until Sunday) and to the shift (typically morning or afternoon) when they have to be serviced .

After servicing a trip in some period, a vehicle returns to the depot, where it is assigned to a parking position and to an itinerary of the next period. Therefore, based on the arrival times of period p, on the description of the depot, and on the departure times of period $p+1$, we can define an instance of the VPP associated with period $p+1$. Finally, because the schedule is usually repeated every week, what we actually have is a *cyclic sequence of periods of activities.*

On the weekends, the number of timetabled itineraries is smaller, and in these situations the transport companies usually partition their fleet in two groups. The first one contains the vehicles that will service trips during the weekend periods, and the second contains buses that will stay parked until the first weekday period of the next week. This division is defined in the last weekday period, and as it affects the assignments that will be made on the next week, a decomposition strategy may produce unsatisfactory plans.

As far as we know, this multi-periodic version of the problem was not yet explicitly defined and investigated in the literature. Therefore, we present below a combinatorial description of the problem.

Definition 1.3. *The basic version of the* Multi-Periodic Vehicle Positioning Problem*, to which we will refer as the* VPP^P*, is defined as follows:*

***Input**: We are given a sequence of $P = p + w$ instances of the* VPP *($P, p, w \in \mathbb{N}$), where the first p periods are* weekday periods *and the last w periods are* weekend periods*; a sequence $\mathcal{A}_h = \{a_1, a_2, \ldots, a_n\}$ of arrival times, a sequence $\mathcal{D}_h = \{d_1, d_2, \ldots, d_n\}$ of departure times, and a set $\mathcal{S}_h = \{s_1, s_2, \ldots, s_m\}$ of tracks for $1 \le h \le p$ ($h, n, m \in \mathbb{N}$); a sequence $\mathcal{A}_h = \{a_1, a_2, \ldots, a_e\}$ of arrival times, a sequence $\mathcal{D}_h = \{d_1, d_2, \ldots, d_e\}$ of departure times, and a set $\mathcal{S}_h = \{s_1, s_2, \ldots, s_b\}$ of tracks for $p+1 \le h \le P, e \le n$, and $b \le m$ ($h, e, b \in \mathbb{N}$); and a set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_t\}$ of vehicle types and a function $\tau : \{\mathcal{A}_h \cup \mathcal{D}_h\} \to \mathcal{T}$ ($1 \le h \le P$). Each track represents a queue of β parking positions ($\beta \in \mathbb{N}$). For every h, $1 \le h \le P$, sequence $\mathcal{A}_h$ ($\mathcal{D}_h$) is such that $a_i \le a_j$ ($d_i \le d_j$) if and only if $i \le j$.*

***Task**: Determine a set M of triples in $\mathcal{A} \times \mathcal{S} \times \mathcal{D}$, where $\mathcal{A} = \bigcup_h^P \mathcal{A}_h$, $\mathcal{S} = \bigcup_h^P \mathcal{S}_h$, and $\mathcal{D} = \bigcup_h^P \mathcal{D}_h$, such that: each element of $\mathcal{A}$ and $\mathcal{D}$ appears exactly once; each element of $\mathcal{S}$ appears exactly β times; (a_i, s_j, d_k) belongs to M only if $\tau(a_i) = \tau(d_k)$ and if either a_i belongs to the last weekday period and d_k and s_j belong to the first weekday period or if a_i, s_j, and d_k belong to the same period; and the number of pairs of elements $(a_i, s_j, d_k), (a_p, s_j, d_r)$ in M such that $i < p$ and $k > r$ is minimized.*

We will also refer to the version of the VPP^P where first crossings are to be minimized (instead of crossings) as the VPP^P, and again, eventual ambiguities are eliminated by the context.

The VPP^P can only be addressed by advanced solution methods. In this thesis, we present two approaches. The first is a branch-and-price algorithm, whose pricing problem can be solved in polynomial time if we want to minimize first crossings and all the vehicles have the same size. We also present heuristics (one based on simulated annealing), inequalities, and symmetry-breaking techniques to speed up the procedure. Our second approach, called *progressive method*, implements an iterative way to solve ILPs with certain characteristics. In this method, variables and constraints are progressively incorporated into the problem, making the computation of feasible solutions easier and faster. With both techniques, we were able to obtain satisfactory solutions for large-scale scenarios of the problem.

Disruptions. One aspect that plays an important role in real-world instances of the VPP and of the VPP^P is the occurrence of disruptions. More precisely, there are many uncontrollable events that can retard the journey of the vehicles and eventually change the order of their arrival in the depot. When such modifications happen, unexpected crossings might happen if the original assignments are maintained. Methods to avoid such events and to react to them play a crucial role in real-world applications of the Vehicle Positioning Problem.

We introduce in this text a new online version of the VPP, present a competitive analysis, and suggest an algorithm for the problem. We also define a new criterion to evaluate the robustness of solutions for the VPP and present an ILP formulation based on this concept.

Additional Aspects. Some extra conditions should be considered if one is interested in a more realistic description of certain applications of the VPP.

When the difference between the departure times of two trips is too small, it may be hard or even physically impossible to dispatch two vehicles leaving from the same track to service them. In this situation, it is interesting to assume that the minimum headway between the times of departure of any pair of units assigned to the same track is given by a parameter Δ.

It is typically assumed that each vehicle can be assigned to any track in a depot, but this is not always the case. Restrictions on the assignment of vehicles of certain types to tracks occur mostly due to physical limitations. Tall units can not be assigned to tracks where the ceiling is too low, for example. Conversely, as we already mentioned, in some cases, certain trips can be serviced by vehicles of different types. These generalizations make the problem more challenging, especially when we consider multi-periodic scenarios.

The single-periodic version of the Vehicle Positioning Problem with these extra features can be described as follows:

Definition 1.4. *The extended version of the* Vehicle Positioning Problem, *to which we refer as the* VPP$_+$, *is defined as follows:*

Input: *We are given a sequence $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ of arrival times ($n \in \mathbb{N}$), a sequence $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$ of departure times, a set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t\}$ of vehicle types ($t \in \mathbb{N}$), a set $\mathcal{T}(d_i) \subseteq \mathcal{T}$ for each $d_i \in \mathbb{D}$, a function $\tau : \mathcal{A} \to \mathcal{T}$, a set $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ of tracks ($m \in \mathbb{N}$), where each track represents a queue of β parking positions ($\beta \in \mathbb{N}, n = m\beta$), and a set $\mathcal{T}(s_i) \subseteq \mathcal{T}$ for each track s_i. Sequence $\mathcal{A}$ ($\mathcal{D}$) is such that $a_i \leq a_j$ ($d_i \leq d_j$) if and only if $i \leq j$.*

Task: *Determine a set M of triples $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ such that each element of $\mathcal{A}$ and $\mathcal{D}$ appears exactly once; each element of $\mathcal{S}$ appears exactly β times; (a_i, s_j, d_k) belongs to M only if $\tau(a_i) \in \mathcal{T}(s_j) \cap \mathcal{T}(d_k)$; and the number of pairs of elements $(a_i, s_j, d_k), (a_p, s_j, d_r)$ in M such that $i < p$ and $k > r$ is minimized.*

Below we present the combinatorial description of the multi-periodic version of the problem.

Definition 1.5. *The extended version of the* Multi-Periodic Vehicle Positioning Problem, *to which we will refer as the* VPP$^P_+$, *is defined as follows:*

Input: *We are given a sequence of $P = p + w$ instances of the* VPP$_+$ *($P, p, w \in \mathbb{N}$), where the first p periods are* weekday periods *and the last w periods are* weekend periods*; a sequence $\mathcal{A}_h = \{a_1, a_2, \dots, a_n\}$ of arrival times, a sequence $\mathcal{D}_h = \{d_1, d_2, \dots, d_n\}$ of departure times, and a set $\mathcal{S}_h = \{s_1, s_2, \dots, s_m\}$ of tracks for $1 \leq h \leq p$ ($h, n, m \in \mathbb{N}$); a sequence $\mathcal{A}_h = \{a_1, a_2, \dots, a_e\}$ of arrival times, a sequence $\mathcal{D}_h = \{d_1, d_2, \dots, d_e\}$ of departure times, and a set $\mathcal{S}_h = \{s_1, s_2, \dots, s_b\}$ of tracks for $p+1 \leq h \leq P, e \leq n$, and $b \leq m$ ($h, e, b \in \mathbb{N}$); a set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t\}$ of vehicle types ($t \in \mathbb{N}$); t_i units of vehicles of type $\mathcal{T}_i$ ($1 \leq i \leq t$, $t_i \in \mathbb{N}$); a set $\mathcal{T}(a_i) \subseteq \mathcal{T}$ for each $a_i \in \mathcal{A}$, where $\mathcal{A} = \bigcup_h^P \mathcal{A}_h$; a set $\mathcal{T}(d_i) \subseteq \mathcal{T}$ for each $d_i \in \mathcal{D}$, where $\mathcal{D} = \bigcup_h^P \mathcal{D}_h$; and a set $\mathcal{T}(s) \subseteq \mathcal{T}$ for each $s \in \mathcal{S}$, where $\mathcal{S} = \bigcup_h^P \mathcal{S}_h$. Each track represents a queue of β parking positions ($\beta \in \mathbb{N}$). For every h, $1 \leq h \leq P$, sequence $\mathcal{A}_h$ ($\mathcal{D}_h$) is such that $a_i \leq a_j$ ($d_i \leq d_j$) if and only if $i \leq j$.*

Task: *Determine a set M of triples in $\mathcal{A} \times \mathcal{S} \times \mathcal{D}$ such that: each element of $\mathcal{A}$ and $\mathcal{D}$ appears exactly once; each element of $\mathcal{S}$ appears exactly β times; each vehicle type $\mathcal{T}_i$ is associated to t_i arrivals and departures, $1 \leq i \leq t$; (a_i, s_j, d_k) belongs to M only if $\mathcal{T}(a_i) \cap \mathcal{T}(s_j) \cap \mathcal{T}(d_k) \neq \emptyset$ and if either a_i belongs to the last weekday period and d_k and s_j belong to the first weekday period or if a_i, s_j, and d_k belong to the same period; and the number of pairs of elements $(a_i, s_j, d_k), (a_p, s_j, d_r)$ in M such that $i < p$ and $k > r$ is minimized.*

These extensions are more challenging then the basic versions of the problems, but our results show that it is possible to compute satisfactory solutions for them if we use an algorithm based on the progressive method.

1.3 Brief Literature Overview

The Vehicle Positioning Problem has become a “hot topic” over the last decade, but Hansmann (2010) [54] identified articles from the middle of the 19th century which investigate solution methods for early versions of the VPP. Heuristic approaches were considered adequate in that period, and the situation remained like this until the 1980s. However, due to the significant growth of the number of vehicles and trips employed by transport companies over the last decades, such approaches became unsatisfactory for modern scenarios. As a consequence, several articles describing more sophisticated strategies to solve the problem appeared in the last years.

Winter (1998) [91] introduced the first mixed-integer quadratic programming and the first mixed-integer linear programming formulations for the problem. For computational purposes, though, only the linear models were analyzed. The author used heuristics to obtain warm start solutions for the ILPs, and these solutions could only be improved by exact solutions methods after long periods of computation. Winter also investigated online and real-time versions of the problem.

Gallo & Di Miele (2001) [39] presented a model for the Bus Dispatching Problem and used some advanced techniques to solve it, like Lagrangean Relaxation, Bundle Method, and a decomposition method. As a result, the authors were able to compute solutions for small- to medium-sized instances, i.e., for scenarios with less than fifty vehicles.

Hamdouni et al. (2004) [51] also deal with the Bus Dispatching Problem. They presented the first discussion regarding robustness of solutions for the VPP and introduced the important concept of *uniform tracks*, i.e., tracks whose assigned vehicles belong to the same type. The authors managed to solve medium-scale scenarios, but their assumptions and their cost functions are slightly different from the ones employed in this thesis.

Two successful methodologies for the Train Unit Shunting Problem were proposed by Lentink (2006) [72], but in both cases optimality was sacrificed. The first strategy consists of a decomposition of the problem in two parts. Initially, the assignments of arrivals to departures are defined, and after this the resulting units are assigned to the tracks. The problem of the first step is modeled (and solved) as a mixed-integer linear program, while a column generation approach is employed in the second step. In the other strategy, the author computes both assignments in an integrated way, but it is assumed that the solutions must contain at least a certain (fixed) number of uniform tracks. This hypothesis simplifies the problems, but typically it does not correspond to a realistic operational constraint. As we will show, the integrated approach has the best performance of all the solutions proposed in the literature.

A branch-and-cut approach is proposed by Føns (2006) [37] based on the work of Di Stefano & Koči (2004) [29], but the author reports a very poor computational performance of this method.

Jacob (2007) [58] investigates a version of the problem for trains where shunting moves have a different interpretation. In these scenarios, a vehicle composed of n units arrives at the depot (also organized in tracks), has its elements rearranged, and departs. A shunting is an operation where all the units parked in one of the tracks are pushed to another area of the depot, called *hump*. From the hump, the units can be redistributed to the tracks or redirected to the area where the new train is being composed. A robust approach to this problem is presented by Cicerone et al. (2007) [20] and by Cicerone et al. (2009) [21], which are based on the work of Liebchen et al. (2007) [73] on *recoverable robustness*.

More detailed overviews of these early works are provided by Winter (1998) [91] and Føns (2006) [37], while references to more recent results are presented by Hansmann (2010) [54] in his Ph.D. thesis.

1.4 Outline of the Thesis

The thesis is organized as follows. In Chapter 2, we give an overview of the VPP and present the concepts that will be used in this text. Initially, we recall some classic models and introduce new formulations based on aspects that, to the best of our knowledge, were not explored in the literature. We define *first crossings* and *sequential matchings*, two concepts that are used in several parts of this text and that can be considered as important as *uniform tracks* for the study of the VPP. We also introduce generic families of inequalities, discuss possible solutions for some symmetry issues, and analyze the computational complexity of the VPP^P and of some special cases of the VPP. Finally, the computational results presented in this chapter show that large-scale scenarios can only be solved by advanced methods.

Significant advances have been made in Mixed Integer Quadratic Constrained Programming, and nowadays there are satisfactory tools to solve these problems. For this reason, we present in Chapter 3 nonlinear approaches to the VPP. We revisit the first model for the VPP, proposed by Winter (1998) [91], introduce new formulations, and apply a convexification technique to some of them. With the incorporation of penalties for inconsistent assignments, the QP relaxation of one of these models is able to produce nontrivial lower bounds for all the instances of the problem that require crossings. There is no linear model with this theoretical property, and it is also important to notice that this technique seems to be applicable in other contexts.

In Chapter 4, we introduce a set partitioning formulation of the VPP and of the VPP^P. Basically, arrivals and departures are the elements to be partitioned, while configurations of tracks are the sets. These models have a large number of variables, so we apply branch-and-price approaches to solve them. For the minimization of first crossings, we present a pricing algorithm that runs in polynomial time if all the vehicles have the same size. Heuristics for the generation of columns and symmetry breaking techniques are suggested for the improvement of the computational performance. With these formulations, it is possible to produce high-quality solutions for medium and large-scale scenarios of the VPP and of the VPP^P. We also investigate several families of clique inequalities for our models.

A generic iterative methodology to deal with mixed-integer linear programs is introduced in Chapter 5. Basically, we eliminate several variables of an ILP and optimize the resulting program. If the solution is not optimal for the original problem, then we re-incorporate some of the variables and repeat the procedure using the computed solution as a warm start. The method stops as soon as an optimal solution for the original program is obtained. We propose algorithms for the VPP and for the VPP^P based on this technique and show that they are able to compute optimal solutions for large-scale scenarios after less than one hour of consumption.

In Chapter 6, we present formulations of the VPP$_+$ and exact and heuristic solutions methods for the VPP$_+^P$. These extensions incorporate aspects that have not been considered before in the literature. Both the VPP$_+$ and the VPP$_+^P$ are more challenging from a computational point of view, but with an algorithm based on the methodology described in Chapter 5 we were able to obtain very satisfactory solutions for large-scale instances of the VPP$_+^P$.

In Chapter 7, we analyze how disruptions in the arrival sequence affect pre-determined plans for instances of the VEHICLE POSITIONING PROBLEM and investigate strategies to avoid and to react to such events. Initially, we describe an online algorithm used to modify a solution in response to changes in the input data. In order to produce solutions which are less sensitive to disruptions, we introduce a novel criterion to evaluate the robustness of feasible matchings and suggest an ILP formulation based on this concept.

We believe that the solution method described in Chapter 5, the strategies proposed in Chapter 7 to deal with uncertainty, and the algorithms suggested in Chapter 6 for more complex scenarios compose a satisfactory theoretical framework for decision-support software dedicated to real-world scenarios of the VEHICLE POSITIONING PROBLEM.

Chapter 2

The Vehicle Positioning Problem

In this chapter, we give an introduction to the Vehicle Positioning Problem (VPP). Some early models are recalled and new formulations based on aspects that were not explored in the literature are presented. We define *first crossings* and *sequential matchings* - two concepts that play an important role in this thesis - and investigate the similarities between the VPP and the 3-Dimensional Matching Problem. We also analyze the computational complexity of the Multi-Periodic Vehicle Positioning Problem (VPP^P) and of some special cases of the VPP, introduce families of valid inequalities, and discuss strategies to deal with certain symmetry issues.

The chapter is organized as follows. Section 2.1 gives a brief overview of the relevant theoretical framework used in this thesis and introduces IP models for the VPP and for the VPP^P. In Section 2.2 we present several formulations and analyze them from a theoretical and from a computational point of view. Section 2.3 contains models for the minimization of first crossings. Finally, we investigate some theoretical aspects of the problem in Section 2.4.

2.1 Background

In this dissertation, our main idea is to employ mixed integer programming and combinatorial optimization methods in order to solve some versions of the Vehicle Positioning Problem.

Optimization problems play a very important role in Mathematics. In these problems we are given a set S and a computable function $f: S \to \mathbb{R}$, and the objective is to determine an element s in S such that $f(s)$ is maximum (or minimum). If s is such an element, we say that it is an *optimal solution* for the problem. If S is a finite set, we say that this is a *Combinatorial Optimization* problem.

In many cases the set S, although finite, is extremely large. Thus, computing the value of f for each of its elements would consume an unacceptable amount of time. The main goal of research in this area is to develop efficient algorithms to find optimal solutions. Efficiency is usually measured according to the required resources, considering the size of the input as the parameter for such analyses. Generally, efficient algorithms evaluate the function f explicitly for a small subset of elements of S, discarding the other ones by implicit but rigorous arguments.

Combinatorial optimization problems have been studied since the 18th century, but only in the 50's, after the development and establishment of linear and integer programming, a unifying theory emerged. The history of combinatorial optimization in its early years is described by Aardal, Nemhauser & Weismantel (2005) [1, chapter 1]. In the 60's, important contributions were given by Edmonds (1965) [31], and in the 70's the articles of Cook (1971) [22] and Karp (1972) [65] introduced the foundations of *Computational Complexity*, an area that provides frameworks to analyze the hardness of problems from a theoretical point of view. However, we remark that it is possible nowadays to produce software systems which are able to compute optimal solutions with acceptable consumption of time and memory for many problems that are considered theoretically intractable.

As we already mentioned, *Linear Programming* plays a prominent role in the field of optimization. We can say that the area was properly founded with the works of Kantorovich (1960) [61] and Dantzig (1951) [26]. Dantzig introduced the simplex method and was inspired by the work of von Neumann (1947) [90]. Khachiyan (1979) [67] presented the Ellipsoid Algorithm, showing that linear programs can be solved in polynomial time, and Karmarkar (1984) [64] introduced the Interior Point Method. Grötschel (1991) [44] shows how the developments in linear programming increased significantly the applicability of combinatorial optimization.

The Vehicle Positioning Problem is a *Discrete Optimization* problem. For many problems of this nature, the best framework is provided by *Graph Theory*. The first article of this area was written in 1736 by Euler (1741) [33]. Also important is *Mixed Integer Programming*, introduced by Gomory (1958) [41]. Mixed integer programs are systems of linear equations with integrality constraints for some or all their variables.

Uncertainty and imprecision in data are some of the biggest challenges for the practical application of theoretical models. Problems of this nature are addressed by *Stochastic Optimization*, *Robust Optimization*, and *(Combinatorial) Online Optimization*. Dantzig (1955) [27] presented the first discussion on the use of linear programming to deal with uncertainty.

Good texts about linear and integer programming are the books of Schrijver (1986) [82], Wolsey (1998) [94], and Chvátal (1983) [19]. Some of the best references for combinatorial optimization are the collection of Schrijver (2003) [83], the book of Grötschel, Lovász & Schrijver (1988) [45], and the collection of Grötschel, Graham & Lovász (1995) [46]. For computational complexity, we refer to the book of Garey

& Johnson (1979) [40]. For graph theory, the classical reference is the book of Bondy & Murty (1976) [15]. Finally, we give a more detailed literature overview of combinatorial online optimization, stochastic optimization, and robust optimization in Chapter 7.

2.1.1 Notation

We present here concepts and some notation that will be used in this text. It is assumed that undefined elements follow the definitions which are already well-established in the literature.

For the different types of families of optimization problems, we use the following abbreviations: LP is a linear program; ILP is an integer linear program; MINLP is a mixed integer nonlinear program; and MIQCP is a mixed integer quadratic-constrained program.

For any program or model M, we denote its optimal objective value by $V(M)$. If M is an ILP, the optimal objective value of its LP relaxation is $V_{LP}(M)$. If M is an MIQCP, $V_{QP}(M)$ is the optimal objective value of its fractional quadratic programming relaxation. $P_{LP}(M)$ denotes the polytope associated with the LP relaxation of formulation M. We say that two models M and M' are equivalent if, for every solution of M, there is a solution of M' with the same objective value and vice-versa.

2.1.2 The Vehicle Positioning Problem

The first and simplest version of the Vehicle Positioning Problem (VPP) was introduced by Winter (1998) [91], who originally referred to it as the Tram Dispatching Problem. We keep our notation as similar to Winter's as possible. Moreover, several elements have been already introduced in Chapter 1, but we will repeat their description here in order to make this overview complete and self-contained.

Some number n of vehicles *arrive* in a sequence $\mathcal{A} = \{a_1, \dots, a_n\}$. They must be assigned to μ *parking positions* $\mathcal{P} = \{p_1, \dots, p_\mu\}$ in some number $m \leq n$ of *tracks* $\mathcal{S} = \{s_1, \dots, s_m\}$ in a depot; from these parking positions, the vehicles *depart* to service a sequence of timetabled trips $\mathcal{D} = \{d_1, \dots, d_n\}$. We refer to an element $a \in \mathcal{A}$ as an *arrival* and to an element $d \in \mathcal{D}$ as a *departure*. A *unit* is a pair in $\mathcal{A} \times \mathcal{D}$. We denote by $s(p)$ the track to which position p belongs. Given arrivals a_i and a_j (departures d_i and d_j), we say that $a_i < a_j$ ($d_i < d_j$) if $i < j$, i.e., to indicate that a_i (d_i) comes (starts) before a_j (d_j).

We denote by (a, p, d) or by (a, s, d) the *assignment* of arrival a to parking position p or track s and to departure d. Note that the parking positions can be implicitly determined by the assignments involving the preceding vehicle arrivals.

Consequently, we will use (a, s, d) more frequently than (a, p, d). We also say that an arrival a is assigned to a departure d, that an arrival a is assigned to a tracks s (or to a position p), or that a departure d is assigned to a tracks s (or to a position p) if the third element of the assignment (the track or the parking position, the departing trip, and the arriving vehicle, respectively) is irrelevant in the context.

Assignments are restricted by a number of constraints. We consider t vehicle types $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$ and assume that there are t_i elements of type $\mathcal{T}_i$, $i = 1, \ldots, t$, and that $\sum_{i=1}^{t} t_i = n$. Each arriving vehicle a is of *type* $\tau(a)$ and each departing trip d can only be serviced by vehicles of type $\tau(d)$. An assignment (a, s, d) is feasible if $\tau(a) = \tau(d)$; we will denote the set of feasible assignments by $\mathcal{F}$, i.e., $(a, s, d) \in \mathcal{F}$ if and only if $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ and $\tau(a) = \tau(d)$. We denote by $\mathcal{A}^i$ $(\mathcal{D}^i)$ the subsequence of arrivals in $\mathcal{A}$ (departures in $\mathcal{D}$) which contains all the arriving vehicles (departing trips) of type $\mathcal{T}_i$. Clearly, $|\mathcal{A}^i| = |\mathcal{D}^i|$, $1 \leq i \leq t$.

Each arriving vehicle a also has a *size* (or length) $l(a)$ and each track $s \in \mathcal{S}$ has size $\beta > 1$ (if $\beta = 1$, the problem is trivial). We assume that there is enough space to park all vehicles. Note that in the basic version of the problem, the tracks are all identical. If all vehicle types have the same size, which is also the case in the basic version, we have $l(a) = 1$ for all $a \in \mathcal{A}$ and $m\beta \geq n$, i.e., β is the number of parking positions in each track. We assume that $m\beta = n = \mu$.

We assume that the first departure starts after the last arrival and that each track is operated as a FIFO queue, i.e., vehicles enter the track at one end and leave at the other. The mathematical treatment of the problem does not change if the tracks are supposed to operate as a LIFO stack, i.e., if vehicles enter and leave at the same end. Namely, we can use exactly the same models in order to solve instances where all the tracks are LIFOs (stacks) if we invert the order of the elements in the departure sequence. Adjustments for scenarios where different tracks may have different policies are also straightforward.

Consider assignments (a, s, d) $((a, p, d))$ and (a', s', d') $((a', p', d'))$; if either $a < a'$ and $d > d'$ or $d < d'$ and $a > a'$, we say that these assignments are in *conflict* and that a *semi-crossing* occurs between (a, s, d) $((a, p, d))$ and (a', s', d') $((a', p', d'))$ (or simply between (a, d) and (a', d')). If (a, s, d) $((a, p, d))$ and (a', s', d') $((a', p', d'))$ are in conflict and $s = s'$ $(s(p) = s(p'))$, then we say that there is a *crossing* between these assignments. Crossings are denoted by: $(a, s, d) \dagger (a', s, d')$ or by $(a, p, d) \dagger (a', p', d')$ for assignments; $(a, d) \dagger (a', d')$ for units; $(a, p) \dagger (a', p')$ for arrivals and positions belonging to the same track; and $(d, p) \dagger (d', p')$ for departures and positions belonging to the same track. Finally, if $(a, s, d) \dagger (a', s, d')$ and there is no assignment (a'', s, d'') such that $a < a'' < a'$, or if $(a, p, d) \dagger (a', p', d')$ and $p = p + 1$, i.e., if a and a'' are parked in consecutive positions in the same track and are involved in a crossing, then we say that these assignments are also involved in a *first crossing*.

A *configuration* q is a set of assignments (a, s, d) describing the arrivals and departures assigned to a given track s. We denote by $q(\mathcal{A})$ $(q(\mathcal{D}))$ the set of arriving vehicles (departing trips) of configuration q. Given two configurations q and q', we denote by $q \cap q'$ their set of common elements (both arrivals and departures). A pair of configurations q and q' such that $|q \cap q'| > 0$ is said to be *intersecting*. We say that a configuration (or a track) is *uniform* if all the vehicles that it contains (or which are assigned to it) are of the same type.

A *matching* is a set of configurations q_s, one for each track $s \in \mathcal{S}$, such that each arriving vehicle and each departing trip is assigned to a parking position in exactly one track (or configuration). The objective of the Vehicle Positioning Problem is to find a matching that minimizes the total number of shuntings, which can refer to the number of crossings or to the number of first crossings. Note that a matching without crossings is also free of first crossings and vice versa.

2.1.3 The Multi-Periodic Vehicle Positioning Problem

The Multi-Periodic Vehicle Positioning Problem (VPP^P) involves a cyclic sequence of p weekday and w weekend single period instances of VPP, where $P = p + w$. Denote the arrival sequence for period h by $\mathcal{A}_h = \{a_1^h, \ldots, a_{n'}^h\}$ and the departure sequence by $\mathcal{D}_h = \{d_1^h, \ldots, d_{n'}^h\}$, $1 \leq h \leq P$. Departures in $\mathcal{D}_h$ have an expected return time to the depot, and the return times of the trips define the arrival sequence $\mathcal{A}_{(h+1)}$. There is a function $sync : \mathcal{D}_h \rightarrow \mathcal{A}_{h+1}$ for each period h that describes which arrival a_j in $\mathcal{A}_{(h+1)}$ is associated with departure d_i in $\mathcal{D}_h$. Finally, $\mathcal{S}_h$ is the set of tracks available in period h, and $\mathcal{A} = \bigcup_h \mathcal{A}_h$, $\mathcal{S} = \bigcup_h \mathcal{S}_h$, and $\mathcal{D} = \bigcup_h \mathcal{D}_h$ if the distinction of periods is not relevant.

Typically, the number e of timetabled trips on weekend periods is smaller than the number n of trips on regular (weekday) periods. These trips must be serviced by a reduced number of vehicles (also equal to e), and these vehicles can only be assigned to $b \leq m$ tracks of the depot. The idle $n - e$ vehicles stay parked in the remaining $m - b$ tracks during the weekend periods and can only be assigned to departures when the next weekday period starts, i.e., after the end of the cycle (which usually represents a week). We denote the set of configurations for a track s in period h by $\mathcal{Q}_s^h$.

The VPP^P has not been considered in the literature before. For this reason, in this chapter we only consider formulations for the VPP. Finally, we leave the introduction of some special aspects and generalizations of both problems to the chapters where they are investigated.

2.1.3.1 Generation of Instances

In order to perform computational experiments for our models and algorithms, we generated random instances. The schemes of generation employed in different chapters vary according to the specifications of the problems that were being solved, but all of them follow the guidelines that we describe below.

Let n, m, and t (denoting the number of vehicles, the number of tracks, and the number of types, respectively) be the parameters of the instance I of the VPP that we want to generate, and let $random(a, b)$ be a procedure that chooses uniformly a random integer between a and b (inclusively). Initially, we generate a random arrival sequence. For this, we select uniformly at random a type $\mathcal{T}_i$ for each arrival a, i.e., $\tau(a) = \mathcal{T}_i$ with probability $\frac{1}{t}$, $a \in \mathcal{A}$ and $1 \leq i \leq t$.

The departure sequence $\mathcal{D}$ of I can be seen as a permutation of the arrival sequence $\mathcal{A}$. Consequently, we use an algorithm that receives a vector v as input and returns a random permutation of v. Algorithm 1, also known as *Fisher-Yates shuffle* (see Fisher (1948) [36]), presented by Cormen et al. (2001) [23], performs this task. Procedure $swap(a, b)$ indicates that a and b exchange their positions in $\mathcal{D}$.

Algorithm 1 Randomize Sequence

```
$\mathcal{D} := \mathcal{A}$
$n := size(\mathcal{A})$
**for** $i := 1$ to $n$ **do**
    $swap(\mathcal{D}[i], \mathcal{D}[random(i, n)])$
**end for**
```

The sequence $\mathcal{D}$ generated by Algorithm 1 is a uniformly random permutations of $\mathcal{A}$ (see, e.g., Cormen et al. (2001) [23] for a proof).

For the VPP^P, we also have to determine a bijection between departures of period h and arrivals of period $h + 1$. More precisely, we have to define the function $sync : \mathcal{D}_h \to \mathcal{A}_{h+1}$ for each period h. Using the same idea of Algorithm 1, we suggest Algorithm 2, which generates a mapping of $\mathcal{D}_h$ to $\mathcal{A}_{h+1}$ uniformly at random.

Algorithm 2 Randomize Synchronization

```
$n := size(\mathcal{A}_h)$
**for** $i := 1$ to $n$ **do**
    $sync(d_i^h) = a_i^{h+1}$
**end for**
**for** $i := 1$ to $n$ **do**
    $swap(sync(d_i^h), sync(d_{random(i,n)}^h))$
**end for**
```

2.2 Models for the VPP

In this section, we recall classic models for the VPP and their most important properties. We also introduce new inequalities to count crossings and discuss the similarities between the VPP and the 3-Dimensional Matching Problem.

For the sake of clarity, the computational comparisons are presented initially in a "local" way, that is, models which are closely related are directly compared. "Global" comparisons will involve only the formulations with the best performances.

2.2.1 The First Models

We classify the models here according to the number of elements of the tuples used to index the variables of the formulations.

2.2.1.1 Two-index Model

The first model for the VPP was proposed by Winter & Zimmermann (2000) [93] and consists of the following quadratic integer programming formulation for the Tram Dispatching Problem:

$$
\begin{array}{lll}
(\mathbf{W}) & \min \displaystyle\sum_{(a,p)\dagger(a',q)} x_{a,p}x_{a',q} + \sum_{(d,p)\dagger(d',q)} y_{d,p}y_{d',q} & \\
\text{(i)} & \displaystyle\sum_{a\in\mathcal{A}} x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \displaystyle\sum_{p\in\mathcal{P}} x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \displaystyle\sum_{d\in\mathcal{D}} y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \displaystyle\sum_{p\in\mathcal{P}} y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \leq 1 & (a,p,d) \in \mathcal{A}\times\mathcal{P}\times\mathcal{D}, \tau(a) \neq \tau(d) \\
 & x_{a,p}, y_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}
$$

The model uses binary variables $x_{a,p}$, with $a \in \mathcal{A}$ and $p \in \mathcal{P}$, and $y_{d,p}$, with $d \in \mathcal{D}$ and $p \in \mathcal{P}$. If $x_{a,p} = 1$ ($y_{d,p} = 1$), arrival a (departure d) is assigned to parking position p. Equations $(\mathbf{W})$(i)-$(\mathbf{W})$(iv) define the assignment constraints and Inequalities $(\mathbf{W})$(v) enforce the coherence of these assignments by allowing only arrivals and departures of the same type to be assigned to a given parking position. Finally, the quadratic cost function calculates the number of crossings.

In his work, Winter did not solve the quadratic program directly. Instead, he tried several linearization methods on $(\mathbf{W})$. The best resulting formulation is reproduced below and is based on the technique of Kaufmann & Broeckx (1978) [66]:

$$
\begin{array}{lll}
(\mathbf{LW}) & \min \sum_{(a,p)} w_{a,p} + \sum_{(d,p)} u_{d,p} & \\
\text{(i)} & \sum_{a} x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \sum_{p} x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \sum_{d} y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \sum_{p} y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \le 1 & \substack{(a,p,d)\in\mathcal{A}\times\mathcal{P}\times\mathcal{D} \\ \tau(a)\neq\tau(d)} \\
\text{(vi)} & d^x_{a,p} x_{a,p} - w_{a,p} + \sum_{(a,p)\dagger(a',q)} x_{a',q} \le d^x_{a,p} & p \in \mathcal{P}, a \in \mathcal{A} \\
\text{(vii)} & d^y_{d,p} y_{d,p} - u_{d,p} + \sum_{(d,p)\dagger(d',q)} y_{d',q} \le d^y_{d,p} & p \in \mathcal{P}, d \in \mathcal{D} \\
 & x_{a,p}, y_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P} \\
 & w_{a,p}, u_{d,p} \in \mathbb{N} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}
$$

In this model, the integer variables $w_{a,p}$ and $u_{d,p}$ count the number of crossings involving the assignments (a,p) and (d,p) in Inequalities (**LW**)(vi) and (**LW**)(vii), respectively. Constants $d^x_{a,p}$ and $d^y_{d,p}$ are upper bounds for $w_{a,p}$ and $u_{d,p}$, respectively, and are computed a priori.

The following facts are known about these models:

Remark 2.1. *Model (**W**) has $2n^2$ variables and $O(n^3)$ constraints.*

Remark 2.2. *Model (**LW**) has $4n^2$ variables and $O(n^3)$ constraints.*

Theorem 2.3 (Winter (1998) [91]). *For each feasible solution of (**W**), there is a feasible solution for (**LW**) with the same objective value.*

Theorem 2.4 (Winter (1998) [91]). *The linear relaxation of model (**LW**) always has optimal objective value equal to zero, i.e., $V_{LP}(\boldsymbol{LW}) = 0$.*

Table 1 gives the results of a computational comparison between models (**W**) and (**LW**) on a test set containing ten artificial instances of small and medium sizes. These computational experiments were performed on an Intel(R) Core 2 Quad 2660 MHz with 4GB RAM, running under openSUSE 11.1 (64 bits). We used `CPLEX` 11.2 (ILOG (2010) [57]) to solve linear programs, `SCIP` 1.0 (Achterberg (2007) [2]) to solve integer programs, `SNIP` 1.0 (Vigerske (2009) [88]) to solve integer non-linear programs, and ZIMPL (Koch (2004) [68]) to generate these models.

The first column in this table gives the name t-m-β of the instance. Following our notation, t is the number of vehicle types, n is the number of tracks, and β is the number of parking positions per track. Based on these parameters, the arrival sequences $\mathcal{A}$ were randomly generated (i.e., the type of each vehicle was uniformly chosen among the t possibilities), while their respective departure sequences $\mathcal{D}$ were obtained by the application of Algorithm 1 in $\mathcal{A}$. The columns labeled Row, Col,

			(**LW**)					(**W**)		
Name	Row	Col	NZ	Nod	T	Row	Col	NZ	Nod	T
3-6-4	10465	2305	43741	1343	58	9325	1165	21889	215	142
4-6-4	11617	2305	46045	12849	265	10477	1165	24193	816	214
5-6-4	12289	2305	47389	32870	654	11149	1165	25537	1010	237
3-7-3	7141	1765	25257	234	18	6273	897	14995	590	58
4-7-3	7897	1765	26769	17220	15	7029	897	16507	523	52
5-7-3	8359	1765	27693	114	19	7491	897	17431	651	64
3-7-4	16297	3137	68391	17220	124	14743	1583	33937	480	121
4-7-4	18145	3137	72087	7393	574	16591	1583	37633	1609	251
5-7-4	19209	3137	74215	60590	2171	17655	1583	39761	113997	11845
3-7-5	31151	4901	152125	59992	3251	28715	2465	64471	6612	76685

Table 1: Comparing models (**LW**) and (**W**).

and NZ give the number of constraints, variables, and non-zeros of the respective program. Columns Nod give the number of nodes in the search tree generated by the corresponding solver and columns T give the running time in seconds. The objective value is equal to zero for all the instances.

As expected, the performance of the linear model is better than the performance of the quadratic formulation, especially when we consider larger instances. Winter (1998) [91] proposed some inequalities to strengthen (**LW**), but his computational results showed that they were not able to make (**LW**) suitable to solve large-scale scenarios.

2.2.1.2 Three-Index Models

Gallo & Di Miele (2001) [39] improved Winter's model by noting that assignments (a, s) and (s, d) implicitly determine the assignment of arrivals and departures to parking positions. For the Train Unit Shunting Problem, which can be seen as an extension of the VPP where vehicles can be composed and decomposed, Kroon, Lentink & Schrijver (2006) [70] proposed a model which explores the same idea and uses triples of $\mathcal{A} \times \mathcal{S} \times \mathcal{D}$ as indices for the decision variables. Based on this approach, we suggest the following new formulation of the VPP:

$$
\begin{array}{lll}
(\mathbf{LU}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(iv)} & \sum_{a'<a} x_{a',s,d} + \sum_{d' \leq d} x_{a,s,d'} - r_{a,s,d} \leq 1 & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F} \\
 & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

This model uses binary variables $x_{a,s,d}$, with $a \in \mathcal{A}$, $s \in \mathcal{S}$, $d \in \mathcal{D}$, and $\tau(a) = \tau(d)$, where $x_{a,s,d} = 1$ if and only if arrival a is assigned to departure d and to track s. Equalities (**LU**)(i) and (**LU**)(ii) are assignment constraints for arrivals and departures, while Equalities (**LU**)(iii) are capacity restrictions for each track in $\mathcal{S}$. Inequalities (**LU**)(iv) count crossings using binary variables $r_{a,s,d}$. More precisely, $r_{a,s,d} = 1$ if and only if $x_{a,s,d'} = 1$, $x_{a',s,d} = 1$, $a' < a$, $d' \leq d$, and $(a, s, d') \dagger (a', s, d)$.

In any feasible solution of (**LU**), n variables $x_{a,s,d}$ are equal to one, while the others are equal to zero. Each arrival and departure belongs to the index of exactly one of these variables, while each track appears exactly β times. It is clear that every solution of (**LU**) can be represented as a solution of (**W**), but the opposite does not hold. This happens because (**LU**) can only represent solutions where the units are distributed in the tracks according to their arrival times, i.e., as we are assuming that all tracks are queues, if a_i and a_j are assigned to the same track and $a_i < a_j$, then a_i is parked in front of a_j (and the trip assigned to a_i ideally starts earlier than the trip assigned to a_j). But Winter (1998) [91] showed that there is always an optimal matching for the problem such that each configuration has this property, so we can conclude that (**LU**) describes and optimizes the same problem as (**W**).

This formulation has the following property:

Remark 2.5. *Model (**LU**) has $O(mn^2)$ variables and $O(mn^2)$ constraints.*

One of the most important differences between (**LU**) and (**LW**) is the necessity of constraints (**LW**)(v). Namely, a triple $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ composes the index of a variable in (**LU**) if and only if $\tau(a) = \tau(d)$. Consequently, assignments (a, s, d) with $\tau(a) \neq \tau(d)$ can be removed from the model, which makes (**LU**) stronger than (**LW**). However, as the following theorem shows, the linear relaxation of (**LU**) remains weak from the practical point of view.

Theorem 2.6. *The linear relaxation of* (***LU***) *always has optimal objective value equal to zero for instances with more than one track, i.e.,* $V_{LP}(\boldsymbol{LU}) = 0$ *if* $m > 1$.

Proof. Let M be any feasible assignment of $\mathcal{A}$ to $\mathcal{D}$. We construct a solution x for (**LU**) as follows:

$$x_{a,s,d} = \begin{cases} \frac{1}{m}, & \text{if } (a,d) \in M \\ 0, & \text{else} \end{cases}, r_{a,s,d} = 0.$$

Constraints (**LU**)(i) and (**LU**)(ii) are clearly satisfied by x, as

$$\sum_s x_{a,s,d} = \sum_s \frac{1}{m} = 1$$

for each $a \in \mathcal{A}$ and $d \in \mathcal{D}$. Moreover, as $|M| = n$,

$$\sum_{(a,d)} x_{a,s,d} = n\frac{1}{m} = \beta$$

for each $s \in \mathcal{S}$, satisfying (**LU**)(iii). Finally, because each arrival is assigned to only one departure, we have

$$\sum_{a'<a} x_{a',s,d} + \sum_{d' \leq d} x_{a,s,d'} \leq \frac{2}{m} \leq 1,$$

and consequently, Constraints (**LU**)(iv) hold with $r_{a,s,d} = 0$ for each (a, s, d) in $\mathcal{A} \times \mathcal{S} \times \mathcal{D}$. We conclude that (x, r) is a feasible solution of $P_{LP}(\mathbf{LU})$ with cost zero. □

It is possible to define conflict graphs for formulations of the VPP such that each vertex represents an assignment and each edge connects a pair of vertices representing assignments which can not appear simultaneously in a feasible solution. Therefore, we refer to clique inequalities for models of the VPP based on this definition of conflict graphs.

Balas & Saltzman (1989) [5] presented four families of clique inequalities for a model for the 3-Dimensional Matching Problem. We can adapt one of them for the (**LU**) as follows:

Proposition 2.7. *Let e be an element of $\mathcal{A}$ or $\mathcal{D}$ of type $\mathcal{T}_k$. Let M^e be the set containing all triples (a, s, d) in $\mathcal{F}$ such that $e = a$ or $e = d$ if $e \in \mathcal{A}$ or $e \in \mathcal{D}$, respectively. Then*

$$\sum_{(a,s,d)\in M^e} x_{a,s,d} \leq 1$$

is a maximal clique inequality for (***LU***) *and $|M^e| = mt_k$.*

Proof. Let e be be an arrival a of $\mathcal{A}$ of type $\mathcal{T}_k$. From the set of triples $M^a \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ containing a, exactly one of its elements appears in a feasible solution of VPP. We conclude that

$$\sum_{(s,d)} x_{a,s,d} \leq 1,$$

which is implied by (**LU**)(i). The cardinality of M^a is equal to the number t_k of departures of type $\mathcal{T}_k$ multiplied by the number of tracks, i.e., $|M^a| = mt_k = O(mn)$.

For any triple (a', s, d), $a \neq a'$, if $m \geq 2$ and if $t_k \geq 2$, then there is at least one triple (a, s', d') in M^a such that $s' \neq s$ and $d' \neq d$, i.e., (a', s, d) and (a, s', d') can belong simultaneously to a solution of VPP. Consequently, a clique inequality containing variables representing the triples of $M^a \cup \{(a', s, d)\}$ is not valid for (**LU**).

Finally, the same argument holds for each departure d. $\square$

Table 2 gives the results of a computational test of (**LU**) on the same machine and on the same dataset used in Subsection 2.2.1.1 plus three additional instance, whose optimal objective values are also equal to zero and which could not be solved with (**W**) after many hours of computation. The comparison shows a clear superiority of the three-index model.

2.2.2 Alternative Inequality to Count Crossings

As we have seen in Theorem 2.4 and in Theorem 2.6, the optimal objective values of the linear relaxation of the models presented so far are always equal to zero. Because deciding if an instance of the VPP needs crossings is $\mathcal{NP}$-complete (Theorem 1.2), it is probably impossible to create an ILP formulation with polynomial dimensions whose linear relaxation is always able to yield non-trivial lower bounds.

Name	Row	Col	NZ	Nod	T
3-6-4	3511	4609	38017	1	1
4-6-4	3511	4321	30241	1	0
5-6-4	3511	4153	25675	59	15
3-7-3	3137	4117	30871	12	8
4-7-3	3137	3865	24816	1	1
5-7-3	3137	3711	21274	54	6
3-7-4	5552	7323	67803	1	1
4-7-4	5552	6861	53509	41	29
5-7-4	5552	6595	45389	1	1
3-7-5	8653	11439	126099	1	4
4-7-5	8653	10725	98582	59	44
5-7-5	8653	10291	82321	26	38
6-7-6	12440	14407	117307	227	200

Table 2: Solving the VPP using (**LU**).

However, each Inequality (**LU**)(iv) considers just one potential crossing between units containing a given arrival and a given departure, i.e., $r_{a,s,d} \in \{0,1\}$ for every $(a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$. Conversely, all the crossings involving an arriving vehicle (a departing trip) and a parking position are considered simultaneously in Inequalities (**LW**)(vi) ((**LW**)(vii)). It is natural to suppose that an inequality considering several potential crossings can be more "sensitive" and, consequently, more capable to produce non-trivial lower bounds if the integrality constraints are relaxed.

For this reason, we modify (**LU**) by replacing Inequalities (**LU**)(iv) with inequalities that count all the crossings involving each assignment (a,s,d), obtaining the model below.

$$
\begin{array}{lll}
(\mathbf{LUE}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d}/2 & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(iv)} & \displaystyle\sum_{(a,d)\dagger(a',d')} x_{a',s,d'} \leq \beta(1 - x_{a,s,d}) + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F} \\
 & r_{a,s,d} \in \mathbb{R} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

Because (**LUE**)(iv) counts all the crossings involving an assignment (a, s, d), variables $r_{a,s,d}$ can not be binary. The amount of crossings is always integer, but as all the other terms of (**LUE**)(iv) are integer, we can allow variables $r_{a,s,d}$ to be real. Besides, each crossing involving assignments (a, s, d) and (a', s', d') is computed twice, and this the reason why the cost function of (**LUE**) is divided by two.

The following remark shows that the programs generated from (**LUE**) and (**LU**) have similar sizes.

Remark 2.8. *Model (**LUE**) has $O(mn^2)$ variables and $O(mn^2)$ constraints.*

The following proposition shows that Inequalities (**LUE**)(iv) bring improvements from a theoretical point of view.

Proposition 2.9. *The linear relaxation of (**LUE**) provides non-trivial lower bounds for some instances of* VPP.

Proof. We present a family of instances $\mathcal{I}$ of VPP for which $V_{LP}(\mathbf{LUE}) > 0$. Let us assume that each instance of $\mathcal{I}$ is such that the first arrival a_1 can only be assigned to the last departure d_n. In this case, an optimal solution of such an instance contains at least $\beta - 1$ crossings.

Let s be any track such that $x_{a_1,s,d_n} = p > 0$. In this case, we know that

$$\sum_{(a',d') \neq (a_1,d_n)} x_{a',s,d'} = \beta - p.$$

As a consequence, Inequality (**LUE**)(iv) for (a_1, s, d_n) is as follows:

$$\begin{aligned}
\sum_{(a',d')\dagger(a_1,d_n)} x_{a',s,d'} &\leq \beta(1 - x_{a_1,s,d_n}) + r_{a_1,s,d_n} \implies \\
\beta - p &\leq \beta(1-p) + r_{a_1,s,d_n} \implies \\
-p &\leq -p\beta + r_{a_1,s,d_n} \implies \\
p(\beta - 1) &\leq r_{a_1,s,d_n}.
\end{aligned}$$

As $p > 0$ and $\beta > 1$, we have a non-trivial lower bound for r_{a_1,s,d_n} and, consequently, for the cost function of each instance of $\mathcal{I}$. $\square$

We did some computational experiments with (**LUE**) and the performance of the solvers was extremely unsatisfactory, as they spent a prohibitive amount of time to generate poor feasible solutions and to close the optimality gaps even for small instances.

2.2.3 A 3-Dimensional Matching Model

Winter's model considers the assignment of arrivals to parking positions and to departures. In other words, the problem consists of a search for n triples in $\mathcal{A} \times \mathcal{P} \times \mathcal{D}$, $|\mathcal{A}| = |\mathcal{P}| = |\mathcal{D}|$, such that each element of the three sets belongs to exactly one of these triples. This is exactly the definition of the 3-Dimensional Matching Problem (**3DMP**), a classical combinatorial optimization problem which is known to be $\mathcal{NP}$-complete (Karp (1972) [65]). Winter (1998) [91] explored this similarity in order to show that the VPP is $\mathcal{NP}$-complete. For a comprehensive overview of the **3DMP** and other assignment problems, we refer to Burkard, Dell'Amico & Martello (2009) [18].

The similarities between these two problems bring us to the following model for the VPP, which is based on a similar well-known formulation of the **3DMP** (see, e.g., Burkard, Dell'Amico & Martello (2009) [18]).

$$
\begin{array}{lll}
(\mathbf{3DM1}) & \min \sum\limits_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & \sum\limits_{(p,d)} x_{a,p,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \sum\limits_{(a,p)} x_{a,p,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum\limits_{(a,d)} x_{a,p,d} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \sum\limits_{\substack{a'<a \\ s(p)=s}} x_{a',p,d} + \sum\limits_{\substack{d'\leq d \\ s(p)=s}} x_{a,p,d'} - r_{a,s(p),d} \leq 1 & (a,p,d) \in \mathcal{A}\times\mathcal{P}\times\mathcal{D} \\
 & x_{a,p,d} \in \{0,1\} & \substack{(a,p,d)\in\mathcal{A}\times\mathcal{P}\times\mathcal{D} \\ \tau(a)=\tau(d)} \\
 & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}.
\end{array}
$$

This model employs binary variables $x_{a,p,d}$, with $a \in \mathcal{A}$, $p \in \mathcal{P}$, $d \in \mathcal{D}$, and $\tau(a) = \tau(d)$, where $x_{a,p,d} = 1$ if and only if arriving vehicle a is assigned to departing trip d and parked in position p. Equations (**3DM1**)(i), (**3DM1**)(ii), and (**3DM1**)(iii) are assignment constraints for arrivals, departures, and parking positions, respectively, while Inequalities (**3DM1**)(iv) count crossings using binary variables $r_{a,s,d}$.

Model (**3DM1**) is basically a mixture of model (**LU**), as it is a three-index model, with model (**W**), as arrivals and departures are assigned to parking positions, and not to tracks. This model has the following property:

Remark 2.10. *Model (**3DM1**) has $O(n^3)$ variables and $O(mn^2)$ constraints.*

The following corollary shows that Inequalities (**3DM1**)(iv) bring the same weakness to (**3DM1**) that they bring to (**LU**):

Corollary 2.11. *The linear relaxation of (**3DM1**) always has optimal value equal to zero for instances with more than one track, i.e., $V_{LP}(\mathbf{3DM1}) = 0$ if $m > 1$.*

Proof. Let M be any set of feasible assignment of $\mathcal{A}$ to $\mathcal{D}$ where each element appears exactly once. If we set $x_{a,p,d} = \frac{1}{\mu}$ for every $p \in \mathcal{P}$, the same arguments employed in the proof of Theorem 2.6 show that it is always possible to build a fractional solution of cost zero for any instance of the VPP containing at least two tracks. □

Finally, the idea proposed in Subsection 2.2.2 leads to the following model for the VPP:

$$
\begin{array}{lll}
(\mathbf{3DM2}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & \displaystyle\sum_{(p,d)} x_{a,p,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,p)} x_{a,p,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(a,d)} x_{a,p,d} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \displaystyle\sum_{\substack{(a,d)\dagger(a',d') \\ s(p)=s}} x_{a',p,d'} \le \beta - \beta \sum_{s(p)=s} x_{a,p,d} + r_{a,s(p),d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & x_{a,p,d} \in \{0,1\} & \substack{(a,p,d)\in\mathcal{A}\times\mathcal{P}\times\mathcal{D} \\ \tau(a)=\tau(d)} \\
 & r_{a,s,d} \in \mathbb{R} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

This model uses real variables $r_{a,s,d}$ that count all the crossings involving the assignment of an arrival a to a departure d in a track s. Consequently, these variables can not be binary, as they can assume any integer value between 0 and β.

Remark 2.12. *Model (**3DM2**) has $O(n^3)$ variables and $O(mn^2)$ constraints.*

Finally, as expected, (**3DM2**) is stronger than (**3DM1**):

Corollary 2.13. *The linear relaxation of (**3DM2**) provides non-trivial lower bounds for some instances of* VPP*.*

Proof. The same family of instances presented in the proof of Proposition 2.9 always produces non-trivial lower bounds when solved with (**3DM2**). □

From the families of inequalities proposed by Balas & Saltzman (1989) [5] for the **3DMP**, two can be adapted to (**3DM1**) and (**3DM2**).

Proposition 2.14. *Let e be an element of $\mathcal{A}, \mathcal{P}$, or $\mathcal{D}$. Let M^e be the set containing all triples (a,p,d) in $\mathcal{A} \times \mathcal{P} \times \mathcal{D}$, $\tau(a) = \tau(d)$, such that $e = a$, $e = p$, or $e = d$ if $e \in \mathcal{A}$, $e \in \mathcal{P}$, or $e \in \mathcal{D}$, respectively. Then*

$$\sum_{(a,p,d)\in M^e} x_{a,p,d} \le 1$$

*is a maximal clique inequality for (**3DM1**) and (**3DM2**) for any triple (a',p',d') in $\mathcal{A} \times \mathcal{P} \times \mathcal{D} \setminus M^e$ and $|M^e| = O(n^2)$.*

Proof. Similar to the proof of Proposition 2.7. □

Proposition 2.15. *For every triple $(a, p, d) \in \mathcal{A} \times \mathcal{P} \times \mathcal{D}$, let*

$$\begin{aligned} T(a,p,d) = \{(a',p',d') : \quad & (a = a' \wedge p = p' \wedge d \neq d') \vee \\ & (a = a' \wedge p \neq p' \wedge d = d') \vee \\ & (a \neq a' \wedge p = p' \wedge d = d')\}. \end{aligned}$$

Then

$$\sum_{(a,p,d) \in \{(a,p,d)\} \cup T(a,p,d)} x_{a,p,d} \leq 1$$

*is a maximal clique inequality for (**3DM1**) and (**3DM2**) for any triple (a', p', d') in $\mathcal{A} \times \mathcal{P} \times \mathcal{D} \setminus \{(a,p,d)\} \cup T(a,p,d)$ and $|\{(a,p,d)\} \cup T(a,p,d)| = O(n)$.*

Proof. It follows directly from the Pigeonhole Principle that any pair of triples in $T(a,p,d) \cup \{(a,p,d)\}$ contains at least one element in common.

Let (a', p', d') be a triple in $\mathcal{A} \times \mathcal{P} \times \mathcal{D} \setminus T(a,p,d) \cup \{(a,p,d)\}$. If it has one element in common with (a,p,d) (let us assume w.l.o.g. that a is this element) and if there are at least 2 arrivals of type $\tau(a)$, then there is an element (a'', p, d) in $T(a,p,d) \cup \{(a,p,d)\}$ such that $a'' \neq a$, $p' \neq p$, and $d' \neq d$. Consequently, a clique inequality containing variables representing the triples of $T(a,p,d) \cup \{(a,p,d)\} \cup \{(a',p',d')\}$ is not valid for (**3DM1**) and (**3DM2**).

Finally, there are $O(n)$ triples sharing two elements with (a,p,d). Because there are 3 such pairs, $|\{(a,p,d)\} \cup T(a,p,d)| = O(n)$. □

Some preliminary computational experiments show that (**3DM2**) is useless for practical applications. Formulation (**3DM1**) is better than (**LUE**) and (**3DM2**), but we saw that it would take us a prohibitively large amount of time to optimize even small-scale scenarios of the problem with this model. For this reason, we did not present computational results in this section.

2.3 Minimization of First Crossings

Traditionally, shunting movements are computed as the number of crossings in a solution for the VPP. However, this should not be considered the ultimate criterion of evaluation. For example, if the rearrangement of vehicles must be performed only when all the units are parked (i.e., after the last arrival and before the first departure) and two buses parked in the last

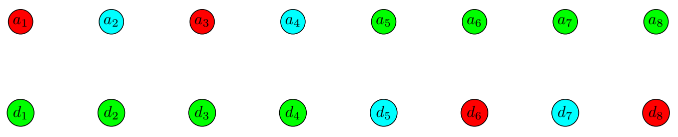


Figure 1: Example with 8 arrivals and departures

positions of some track are involved in a crossing, then all the vehicles assigned to this track will be involved in the shunting operation. Therefore, in this case, changing the position of the last two buses may require the same amount of work as changing the positions of all the units in this track.

Winter (1998) [91] showed that any solution containing crossings also contains first crossings (the second concept was neither explicitly defined nor explored by the author, though). This fact is clear, as every first crossing is also a crossing. Also clear is the following remark:

Remark 2.16. *The minimum number of first crossings is a lower bound for the minimum number of crossings for any instance of* VPP*.*

These facts suggest that the minimization of first crossings and the minimization of crossings are closely related. However, the following result shows that these two minimization problems do not necessarily share optimal solutions.

Proposition 2.17. *A solution that minimizes the number of crossings does not necessarily minimize the number of first crossings and vice-versa.*

Proof. Consider the following family of instances of VPP; the set $\mathcal{T}$ contains three types (say, $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$), sequence $\mathcal{A}$ is such that

$$\tau(a_i) = \begin{cases} \mathcal{T}_1, & \text{if } i = 1 \text{ or } 3 \\ \mathcal{T}_2, & \text{if } i = 2 \text{ or } 4, \\ \mathcal{T}_3, & \text{if } 5 \leq i \leq n, \end{cases}$$

sequence $\mathcal{D}$ is such that

$$\tau(d_i) = \begin{cases} \mathcal{T}_1, & \text{if } i = n-2 \text{ or } n \\ \mathcal{T}_2, & \text{if } i = n-3 \text{ or } n-1, \\ \mathcal{T}_3, & \text{if } 1 \leq i \leq n-4, \end{cases}$$

and $\beta = 4$. The instance of this family for $n = 8$ (and, consequently, $m = 2$) is presented in Figure 1.

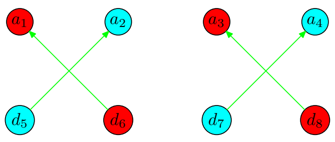


Figure 2: Minimizing crossings

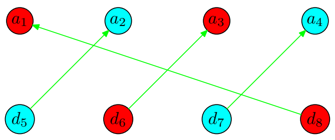


Figure 3: Minimizing first-crossings

Clearly, there is semi-crossing involving any unit of type $\mathcal{T}_1$ or $\mathcal{T}_2$ and all the units of type $\mathcal{T}_3$. Consequently, a solution minimizing first crossings or crossings has all the vehicles of types $\mathcal{T}_1$ and $\mathcal{T}_2$ assigned to the same track. As we can see in Figures 2 and 3, the solution that minimizes the number of first crossings is not the same as the one that minimizes the amount of crossings and vice-versa. $\square$

In the following subsections, we modify some of the models we presented on the last sections in order to obtain formulations that minimize the number of first crossings.

2.3.1 Two-index Model

Because the parking positions appear explicitly in the model, we just need to make a simple modification in the cost function of (**W**) in order to obtain a quadratic integer programming formulation that minimizes first crossings:

$$
\begin{array}{lll}
(\mathbf{WF}) & \min \displaystyle\sum_{\substack{(a,p)\dagger(a',q)\\ p=q+1\\ s(p)=s(q)}} x_{a,p}x_{a',q} + \sum_{\substack{(d,p)\dagger(d',q)\\ p=q+1\\ s(p)=s(q)}} y_{d,p}y_{d',q} & \\
\text{(i)} & \displaystyle\sum_a x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \displaystyle\sum_p x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \displaystyle\sum_d y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \displaystyle\sum_p y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \leq 1 & \substack{(a,p,d)\in\mathcal{A}\times\mathcal{P}\times\mathcal{D}\\ \tau(a)\neq\tau(d)} \\
 & x_{a,p}, y_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}
$$

In the cost function of (**WF**), p and q are consecutive positions that belong to the same track. As we saw already, the quadratic model is not very efficient from a computational point of view. Consequently, an adaptation of the linear model (**LW**) is more interesting. For this, we can modify inequalities (**LW**)(vi) and (**LW**)(vii) in order to obtain the following linear model for VPP:

$$
\begin{array}{lll}
(\mathbf{LWF}) & \min \displaystyle\sum_{(a,p)} w_{a,p} + \sum_{(d,p)} u_{d,p} & \\
\text{(i)} & \displaystyle\sum_a x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \displaystyle\sum_p x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \displaystyle\sum_d y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \displaystyle\sum_p y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \leq 1 & \substack{(a,p,d)\in\mathcal{A}\times\mathcal{P}\times\mathcal{D}\\ \tau(a)\neq\tau(d)} \\
\text{(vi)} & x_{a,p} - w_{a,p} + \displaystyle\sum_{\substack{(a,p)\dagger(a',q)\\ q=p-1\\ s(p)=s(q)}} x_{a',q} \leq 1 & p \in \mathcal{P}, a \in \mathcal{A} \\
\text{(vii)} & y_{d,p} - u_{d,p} + \displaystyle\sum_{\substack{(d,p)\dagger(d',q)\\ q=p-1\\ s(p)=s(q)}} y_{d',q} \leq 1 & p \in \mathcal{P}, d \in \mathcal{D} \\
 & x_{a,p}, y_{d,p}, w_{a,p}, u_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}
$$

Inequalities (**LWF**)(vi) and (**LWF**)(vii) are similar to (**LW**)(vi) and to (**LW**)(vii), respectively. The difference lies in the fact that we only count crossings of consecutive parking positions. Because there is at most one possible crossing involving each pair of positions p and $p-1$, the integer constants $d^x_{a,p}$ ($d^y_{d,p}$) are always equal to one, and consequently, both $w_{a,p}$ and $u_{d,p}$ are binary variables.

Models (**WF**) and (**LWF**) are also similar to (**W**) and (**LW**), respectively, when we consider their dimensions and the properties of their linear relaxations.

Remark 2.18. *Model (**WF**) has $2n^2$ variables and $O(n^3)$ constraints.*

Remark 2.19. *Model (**LWF**) has $4n^2$ variables and $O(n^3)$ constraints.*

Corollary 2.20. *For each feasible solution of (**WF**), there is a feasible solution for (**LWF**) with the same objective value.*

Proof. Similar to the proof presented in Winter (1998) [91] for Theorem 2.3. □

Corollary 2.21. *The linear relaxation of (**LWF**) always has optimal solution equal to zero, i.e., $V_{LP}(\boldsymbol{LWF}) = 0$.*

Proof. The same family of instances used in 2.4 to prove that $V_{LP}(\mathbf{LW}) = 0$ can be used to show that $V_{LP}(\mathbf{LWF}) = 0$. □

2.3.2 Three-index Model

Similarly to (**W**) and (**LW**), (**LU**) is meant to be used for the version of VPP where crossings should be minimized. However, in this case, modifications are harder, as parking positions are considered implicitly in the formulations.

The following formulation shows how one can adapt (**LU**) in order to penalize only first crossings:

$$
\begin{array}{lll}
(\mathbf{LUF}) & \min \sum\limits_{(a,s,d,a')} r_{a,s,d,a'} & \\
\text{(i)} & \sum\limits_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \sum\limits_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum\limits_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(iv)} & x_{a,s,d} + \sum\limits_{d' \leq d} x_{a',s,d'} \leq & \\
& 1 + r_{a,s,d,a'} + \sum\limits_{\substack{d'' \\ a<a'' \\ a''<a'}} x_{a'',s,d''} & \substack{(a,s,d,a') \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}\times\mathcal{A} \\ a<a'} \\
& x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F} \\
& r_{a,s,d,a'} \in \{0,1\} & \substack{(a,s,d,a') \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}\times\mathcal{A} \\ a<a'}.
\end{array}
$$

For every tuple $(a, s, d, a') \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \times \mathcal{A}$, $a < a'$, Inequality (**LUF**)(iv) checks if there is a crossing involving (a, s, d) and some assignment (a', s, d'), $d' \leq d$. This crossing is a first crossing (and, consequently, is penalized in (**LUF**)(iv)) if there is no vehicle a'', $a < a'' < a'$, assigned to s. As each vehicle and trip can be involved in at most one first crossing, it is clear that variables $r_{a,s,d,a'}$ are binary.

Remark 2.22. *Model (**LUF**) has $O(mn^3)$ variables and $O(mn^3)$ constraints.*

A comparison between Remark 2.5 and Remark 2.22 shows that (**LUF**) is bigger and potentially harder than (**LU**).

Similarly to (**LU**), the linear relaxation of (**LUF**) always yields a trivial lower bound.

Proposition 2.23. *The linear relaxation of (**LUF**) always has optimal objective value equal to zero for instances with more than one track, i.e., $V_{LP}(\mathbf{LUF}) = 0$ if $m > 1$.*

Proof. Similar to Proposition 2.6. □

2.3.3 Computational Results

We compare (**LUF**) and (**LWF**) from a computational point of view. Table 3 gives the results obtained from the computation of five instances on

	(**LWF**)					(**LUF**)				
Name	Row	Col	NZ	Sol	T	Row	Col	NZ	Sol	T
20-4-3	8140	1200	21440	0	2	5792	6364	297884	0	36
49-7-5	175714	7203	414344	12	*	83622	87080	13974331	7	*
56-8-6	262416	9408	616448	12	*	137616	142536	29129120	8	*
70-10-6	512750	14700	1198400	-	*	291830	300000	81657130	12	*
96-12-7	1542432	27648	3521088	-	*	-	-	-	-	-

Table 3: Comparing models (**LUF**) and (**LWF**).

an Intel Core2 Quad 2.83GHz processor and 16GB RAM using `CPLEX` 12.2 as solver.

The first column in this table gives the name n-m-t of the problem. Following our notation, n is the number of arrivals (and departures), m is the number of tracks, and t is the number of vehicle types. Based on these parameters, the arrival sequences $\mathcal{A}$ were randomly generated (i.e., the type of each vehicle was uniformly chosen among the t possibilities), while their respective departure sequences $\mathcal{D}$ were obtained after the application of Algorithm 1 in $\mathcal{A}$. The columns labeled Row, Col, NZ, and T give the number of constraints, variables, non-zeros, and the time consumed in seconds by the respective model. Each program had a time limit of 10800 seconds. Models that could not obtain an optimal solution in the time limit have $*$ in their column T.

As we can see, (**LUF**) is superior to (**LWF**), but actually both models have poor performances. Moreover, for the largest scenario, it was not possible to obtain a feasible solution with (**LUF**), while the ILPs generated from (**LWF**) exceeded the limit of memory of our computer.

We conclude that the minimization of first crossings is harder than the minimization of crossings when we consider non-advanced solution approaches. That is, it is easier from the computational point of view to minimize crossings than first crossings if we just model these problems as ILPs and try to optimize them using generic solvers. However, in Chapter 4, we introduce a set partitioning approach to the problem and show that this formulation is easier to solve if we minimize first crossings, which shows the usefulness of this concept for the VPP.

2.4 Theoretical Aspects

The first theoretical results related to the VPP were presented by Winter (1998) [91] and Blasum et al. (1999) [12]. Some further aspects were covered by Lentink (2006) [72]. For a comprehensive survey, we refer to the work of Hansmann (2010) [54].

In this section, we present some theoretical aspects of the VPP which were not explored in the literature. Namely, we investigate families of inequalities, symmetry issues, and the computational complexity of some special versions of the problem.

2.4.1 Inequalities

Typically, practical instances of the VPP admit crossing-free solutions. However, when this is not the case, the LP/ILP solvers usually need a considerable amount of time to find nontrivial lower bounds and to close optimality gaps. For this reason, we address families of valid inequalities that might be useful in this matter. We formulate inequalities for (**LU**), but most of the results can be adapted and employed in other models as well.

2.4.1.1 Sequential Matchings

A *sequential matching* of subsequences $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{D}' \subseteq \mathcal{D}$ is the assignment of the i-th arrival in $\mathcal{A}'$ of type $\mathcal{T}_k$ to the i-th departure in $\mathcal{D}'$ of type $\mathcal{T}_k$ for every valid i and k. It is clear that the sequential matching of $\mathcal{A}'$ and $\mathcal{D}'$ is well-defined if and only if $|\mathcal{A}'^i| = |\mathcal{D}'^i|$, $1 \le i \le t$. Finally, the sequential matching of $\mathcal{A}$ and $\mathcal{D}$ is called *complete sequential matching*.

Lemma 2.24. *Given subsequences $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{D}' \subseteq \mathcal{D}$ of size β such that $|\mathcal{A}'^i| = |\mathcal{D}'^i|$, $1 \le i \le t$, the sequential matching produces a configuration with a minimum number of crossings.*

Proof. Let q be a configuration such that $q(\mathcal{A}) = \mathcal{A}'$ and $q(\mathcal{D}) = \mathcal{D}'$ and assume that q does not describe a sequential matching. In this case, there is a crossing involving pairs (a, d) and (a', d') such that $\tau(a) = \tau(a')$. This situation is represented in Figure 4, where a_2, a_4, d_2, and d_4 represent individual elements, with $\tau(a_2) = \tau(a_4) = \tau(d_2) = \tau(d_4)$, and A_1, A_3, A_5, D_1, D_3, and D_5 represent (possibly empty) sequences of arrivals and departures.

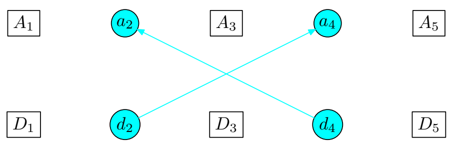


Figure 4: Assignment that does not describe a sequential matching

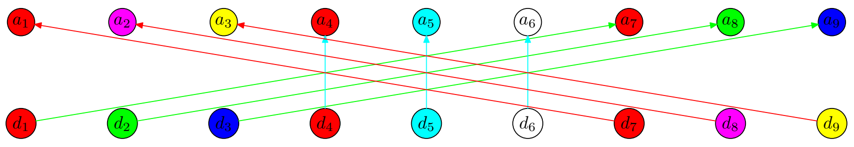


Figure 5: Example with 9 elements and 3 tracks

If we substitute for the assignments (a_2, d_4) and (a_4, d_2) the assignments (a_2, d_2) and (a_4, d_4), we obtain a configuration q' whose number of crossings is smaller than the number of crossings of q, as we are eliminating the crossings involving units (a_2, d_4) and (a_4, d_2) and units whose arrivals is in A_3 and whose departures is in D_3. We remark that the other original crossings remain and that no extra crossing appears after this substitution.

If we apply this substitution to every pair of crossing assignments of the same type, we will obtain a new configuration with less crossings. It follows directly that the sequential matching of the elements of sets $\mathcal{A}'$ and $\mathcal{D}'$ provides the configuration with the minimum number of crossings. □

Corollary 2.25. *A solution using the complete sequential matching minimizes the number of semi-crossings.*

However, the complete sequential matching does not yield an optimal solution for every instance of VPP. In Figure 5, we show an example with nine arrivals and departures and three tracks for which the complete sequential matching leads to a scenario where crossings are unavoidable, while the optimal solution (whose assignments are indicated in the figure) does not contain crossings.

Model (**LU**) counts the number of crossings by penalizing pairs $(a, d) \in \mathcal{A} \times \mathcal{D}$ that belong to assignments (a, s, d') and (a', s, d) such that $(a, s, d') \dagger (a', s, d)$. We use Lemma 2.24 in order to show that a relaxation of (**LU**) can be considered instead.

Proposition 2.26. *If we remove Inequalities (**LU**)(iv) for triples $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ such that $\tau(a) = \tau(d)$, it is still possible to find an optimal solution for the original problem with the same objective value from this relaxation.*

Proof. It is clear that the value of an optimal solution of the relaxed model is smaller than or equal to the optimal value of the complete model, as every feasible solution for the last is also a feasible solution for the first and potentially "more penalized", as it considers crossings ignored in the relaxed model. We show that we can build a solution for the complete model with the same cost value out of every solution of the relaxed model.

Let x be an optimal solution for the relaxed model. This solution might contain crossings involving pairs (a, d) and (a', d') such that $\tau(a) = \tau(a')$. However, from Lemma 2.24, if we substitute for the assignments $q(\mathcal{A}) \to q(\mathcal{D})$ of each configuration q the sequential matching of $q(\mathcal{A})$ and $q(\mathcal{D})$, we will have a solution x' where such crossings do not happen. Finally, it follows from the optimality of x and from Lemma 2.24 that the value of x in the relaxed model is equal to the value of x' in the complete model. □

2.4.1.2 Global Lower Bound

For every instance of the VPP, we can define a VPP-*graph* G whose vertices represent feasible assignments $(a, d) \in \mathcal{A} \times \mathcal{D}$ and whose edges indicate a semi-crossing between the units represented by its incident vertices. If we select a set of vertices of G such that each arrival and each departure appears exactly once, then these vertices represent a matching $\mathcal{A} \to \mathcal{D}$.

Cliques induced by subsets of vertices of G represent sets of assignments which cannot be assigned to the same track without crossings. We use this idea in order to derive a valid inequality for the VPP.

Proposition 2.27. *If c is the minimum size of a maximum clique of the subgraph of the* VPP*-graph G induced by some set of vertices representing a matching, then*

$$\sum_{a,s,d} r_{a,s,d} \geq \frac{1}{2} \left\lfloor \frac{c}{m} \right\rfloor (c + (c \mod m) - m)$$

is a valid inequality for VPP.

Proof. This fact follows directly from the Pigeonhole Principle. If $c = am+b$, $a, b \geq 0$, and the crossing assignments can be distributed among the tracks in the best way to avoid crossings, there will be b tracks with at least $a(a+1)/2$

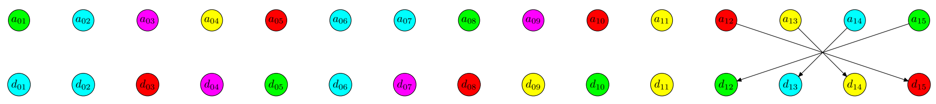

a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 a13 a14 a15
d01 d02 d03 d04 d05 d06 d07 d08 d09 d10 d11 d12 d13 d14 d15

Figure 6: Clique of size 4 in the sequential matching

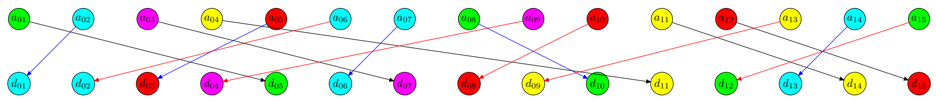

a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 a13 a14 a15
d01 d02 d03 d04 d05 d06 d07 d08 d09 d10 d11 d12 d13 d14 d15

Figure 7: Crossing-free assignment of units to 3 tracks

crossings and $m - b$ tracks with at least $a(a-1)/2$ crossings. It follows that

$$\begin{aligned}\sum_{a,s,d} r_{a,s,d} &\geq \frac{ba(a+1)}{2} + \frac{(m-b)a(a-1)}{2} = \frac{a}{2}\left(b(a+1) + (m-b)(a-1)\right) \\ &= \frac{a}{2}\left(b + ma - m + b\right) = \frac{a}{2}\left(c - m + b\right) \\ &= \frac{1}{2}\left\lfloor \frac{c}{m} \right\rfloor \left(c + (c \mod m) - m\right)\end{aligned}$$

is a valid inequality for (**LU**). □

We observe that $\frac{1}{2}\left\lfloor \frac{c}{m} \right\rfloor (c + (c \mod m) - m) \leq 0$ if $c + (c \mod m) \leq m$.

Based on Corollary 2.25, one can "intuitively" conclude that the complete sequential matching contains the smallest maximum clique of size c. However, in Figure 6, we present an example with fifteen units whose sequential matching contains a clique of size four, and in Figure 7 we show that it is possible to construct a crossing-free solution with three tracks for the same instance.

2.4.1.3 Induced Crossings

The following proposition shows that, in some cases, it is possible to conclude that if a unit (a, d) is used, there will be crossings in the solution.

Proposition 2.28. *Let $(a, d) \in \mathcal{A} \times \mathcal{D}$ be a feasible assignment. If there is no set of units which can compose a crossing-free configuration with (a, d), then any matching containing (a, d) requires at least one crossing.*

Proof. If there is no crossing-free configuration containing unit (a, d), then it is clear that any matching that assigns a to d will require crossings, even if they do not involve (a, d) directly. □

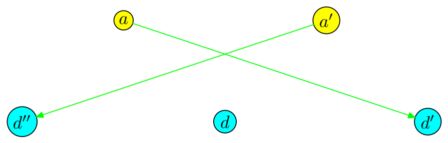


Figure 8: Departure d is involved in semi-crossings with (a, d') and (a', d'')

We say that the (a, d)-*adjusted complete sequential matching* is the matching composed by (a, d) and by the units of the sequential matching of $\mathcal{A} \setminus \{a\}$ and $\mathcal{D} \setminus \{d\}$. Let $C'(a, d)$ be the set of assignments of the (a, d)-adjusted complete sequential matching which do not cross with (a, d), and let $c'(a, d)$ be the minimum number of crossings between any subset of $\beta - 1$ elements of $C'(a, d)$.

Proposition 2.28 gives a direct lower bound for the cost function. Moreover, because each arrival is assigned to exactly one pair in an integer solution, we can consider all the lower bounds for units sharing the same arrival simultaneously, obtaining

$$\sum_{a',s',d'} r_{a',s',d'} \geq \sum_{\substack{s',d' \\ c(a,d')>0 \text{ or} \\ |C'(a,d')|<\beta-1}} x_{a,s',d'} \qquad a \in \mathcal{A}. \tag{1}$$

Analogously,

$$\sum_{a',s',d'} r_{a',s',d'} \geq \sum_{\substack{a',s' \\ c(a',d)>0 \text{ or} \\ |C'(a',d)|<\beta-1}} x_{a',s',d} \qquad d \in \mathcal{D}. \tag{2}$$

2.4.1.4 Inequalities With Variables $r_{a,s,d}$

Consider arrivals a and a', with $a < a'$, a track s, and a departure d. If a is assigned to a departure d', $d' > d$, and a' is assigned to a departure d'', $d'' < d$, there will be a crossing involving assignments (a, s, d') and (a', s, d''), and as a consequence, if departing trip d is also assigned to track s, then it will certainly be involved in a crossing as well. A graphic representation of this situation is given in Figure 8.

These observations can be translated into the following inequalities for (**LU**):

$$\sum_{d' \geq d} x_{a,s,d'} + \sum_{d'' \leq d} x_{a',s,d''} - \sum_{\substack{a^* \\ s' \neq s}} x_{a^*,s',d} \leq \ 1 + \sum_{a^*} r_{a^*,s,d}$$

for arrivals $a, a' \in \mathcal{A}$, $a < a'$, track $s \in \mathcal{S}$ and departure $d \in \mathcal{D}$. We can assume that $\tau(a) \neq \tau(a')$. Analogously,

$$\sum_{a' \geq a} x_{a',s,d} + \sum_{a'' \leq a} x_{a'',s,d'} - \sum_{\substack{d^* \\ s' \neq s}} x_{a,s',d^*} \leq \quad 1 + \sum_{d^*} r_{a,s,d^*};$$

for departures $d, d' \in \mathcal{D}$, $d < d'$, track $s \in \mathcal{S}$ and arrival $a \in \mathcal{A}$. Again, we assume that $\tau(d) \neq \tau(d')$.

2.4.2 Symmetry

One of the biggest challenges for most models of the VPP is their symmetry. In this section we investigate some sources of symmetry and propose strategies to attenuate and eventually to eliminate their effects.

2.4.2.1 Track Symmetry

Let x be a solution of an instance of VPP, and let s and s' be two tracks of this instance. Let x^* be a solution derived from x by exchanging the assignments of s with the assignments of s', i.e.,

- $x^*_{a,s',d} = x_{a,s,d}$;
- $x^*_{a,s,d} = x_{a,s',d}$;
- $x^*_{a,s'',d} = x_{a,s'',d}$ for $s'' \neq s$ and $s'' \neq s'$.

It is clear that x^* is similar to x, as the only difference between them lies in the labeling of tracks s and s'. We can apply further exchanges of this nature in order to generate other solutions identical to x. More precisely, every matching can be represented in $m!$ different ways if the tracks are all equal but their labels are not ignored.

One simple strategy employed in assignment problems to avoid this issue consists of fixing some of the assignments in a way that not every optimal solution of the original problem is removed from the new reduced version. For the VPP, if we fix the assignment of some arrival a_i of type $\mathcal{T}_1$ to track s_1, we can remove the $t_1(m-1)$ variables representing the assignment of a_i to other tracks. As a result, the number of feasible solutions is divided by $(m)!$, and, more important, at least one optimal solution for the original version clearly remains in the reduced problem.

After fixing the first arrival a_i, we can apply the same idea in order to eliminate other variables. For any $a_j \neq a_i$, we typically do not know in advance if a_i and a_j belong to the same configuration in every optimal solution, but we can assume w.l.o.g. that a_j can only be parked in s_1 or in s_2. Thus we can not fix the assignment of a_j, but we still can remove from the formulation all variables indexed by triples (a_j, s_k, d) such that $s_k > 2$. With this, we also divide the number of feasible solutions by $(m-1)!$.

The idea can be naturally extended and applied to up to m arrivals. Because the amount of variables which are removed after fixing an arrival of type $\mathcal{T}_j$ is proportional to t_j, it is more interesting to apply this procedure to arrivals whose types contain the biggest number of available units.

These ideas are summarized in Algorithm 3, where we describe a routine to avoid track symmetry in model (**LU**).

Algorithm 3 Reduction of Track Symmetry

```
Relabel types $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t$ in a way that $t_i \geq t_{i+1}$, $1 \leq i < t$
$add := 0$
$i := 1$
while $add < |\mathcal{S}|$ do
  $flag := false$
  for $a := 1$ to $n$ do
    if $\tau(a) = \mathcal{T}_i$ and $a$ was not fixed then
      $flag := true$
      for $d := 1$ to $n$ do
        if $\tau(a) = \tau(d)$ then
          for $s := i+1$ to $|\mathcal{S}|$ do
            Remove $x_{a,s,d}$ from the model
          end for
        end if
      end for
    end if
  end for
  if $flag = true$ then
    $add := add + 1$
  else
    $i := i + 1$
  end if
end while
```

2.4.2.2 Uniform Consecutive Subsequences

We say that $\mathcal{A}' \subseteq \mathcal{A}$ ($\mathcal{D}' \subseteq \mathcal{D}$) is a *uniform consecutive subsequence* if all its elements appear consecutively in $\mathcal{A}$ ($\mathcal{D}$) and belong to the same type. The following proposition shows that uniform consecutive subsequences can be sources of symmetry if crossings involving units of the same type are ignored (which can be done according to Proposition 2.26).

Proposition 2.29. *Let $\mathcal{A}'$ be a uniform consecutive subsequence. Let M be a matching and M' its subset containing only the assignments with arrivals of $\mathcal{A}'$. Let $\mathcal{D}'$ be the subset of $\mathcal{D}$ covered by M'. If we keep the assignment of $\mathcal{D}$ to $\mathcal{S}$ and if we exchange M' by any of the other $|\mathcal{A}'|! - 1$ possible assignments of vehicles of $\mathcal{A}'$ to trips in $\mathcal{D}'$, we obtain in each case a new solution with the same number of crossings involving vehicles of different types.*

Proof. Let (a_2, d_4) and (a_4, d_2) be two different assignments in M'. We refer again to Figure 4 and recall that a substitution of these two assignments for (a_2, d_2) and (a_4, d_4) changes the number of crossings among vehicles of different types if and only if there are assignments from elements of A_3 to elements of D_3 whose types are different from $\tau(a_2)$. Because each arrival in A_3 is of type $\tau(a_2)$, the number of such crossings does not change with this substitution. □

Briefly, if the assignments of departures to arrivals of a uniform consecutive subsequence are exchanged and only crossings involving units of different types are considered, the cost value of the resulting matching will be equal to the cost value of the original one.

It is clear that the same argument holds for uniform consecutive subsequences of departures.

Corollary 2.30. *Let $\mathcal{D}'$ be a uniform consecutive subsequence. Let M be a complete matching and M' its subset containing only the assignments with departures of $\mathcal{D}'$. Let $\mathcal{A}'$ be the subset of $\mathcal{A}$ covered by M'. If we keep the assignment of $\mathcal{A}$ to $\mathcal{S}$ and if we exchange M' by any of the other $|\mathcal{D}'|! - 1$ possible assignments of departures of $\mathcal{D}'$ to vehicles in $\mathcal{A}'$, we obtain a new solution with the same number of crossings involving vehicles of different types.*

In order to avoid these symmetry problems, we can use the following inequality for each uniform consecutive subsequence $\mathcal{A}'$:

$$\sum_{s,d'<d} x_{a_{i+1},s,d'} \leq 1 - \sum_{s} x_{a_i,s,d} \quad a_i, a_{i+1} \in \mathcal{A}', d \in \mathcal{D}. \tag{3}$$

Similarly, if $\mathcal{D}'$ is a uniform consecutive subsequence:

$$\sum_{s,a'<a} x_{a',s,d_{i+1}} \leq 1 - \sum_{s} x_{a,s,d_i} \quad d_i, d_{i+1} \in \mathcal{D}', a \in \mathcal{A}. \tag{4}$$

If these inequalities are used, this type of symmetry is removed completely.

While the families of valid inequalities above reduce the number of valid solutions for an instance of VPP, the following proposition shows that, in some cases, it is also possible to eliminate assignment variables from the models.

Proposition 2.31. *Let $\mathcal{A}' = \{a'_1, a'_2, \ldots, a'_k\}$ and $\mathcal{D}' = \{d'_1, d'_2, \ldots, d'_k\}$ be maximal uniform consecutive subsequences. We can eliminate the variables representing assignments of pairs $(a, d) \in \mathcal{A}' \times \mathcal{D}'$ that do not belong to the sequential matching of $\mathcal{A}'$ and $\mathcal{D}'$ without losing all the optimal solutions for the original program.*

Proof. We show that if the variables representing assignments of pairs $(a, d) \in \mathcal{A}' \times \mathcal{D}'$ that do not belong to the sequential matching of $\mathcal{A}'$ and $\mathcal{D}'$ are removed, at least one optimal solution from the original problem remains in the reduced version.

Let x be an optimal solution of the original (complete) model. If all the arrivals in $\mathcal{A}'$ are assigned to the departures in $\mathcal{D}'$ in x, we just have to substitute for these assignments the ones that belong to the sequential matching of $\mathcal{A}'$ to $\mathcal{D}'$ in order to obtain a solution x^* with the same number of crossings. Also trivial is the case where there is no arrival in $\mathcal{A}'$ assigned to departures in $\mathcal{D}'$.

Let us assume that at least one arrival a_j in $\mathcal{A}'$ is assigned to a trip d_i in $\mathcal{D}'$ in x and that (a_i, d_i) and (a_j, d_j) belong to the the sequential matching of $\mathcal{A}'$ and $\mathcal{D}'$. We can suppose w.l.o.g. that both a_i and d_j are assigned to $d_k \notin \mathcal{D}'$ and $a_l \notin \mathcal{A}'$, respectively. If this is not the case (say, because a'_l belongs to $\mathcal{A}'$), then we consider the arrival a_m assigned to d_l. If $a_m \notin \mathcal{A}'$, we found the element we wanted, and if not, we continue with this search until we reach an arrival which does not belong to $\mathcal{A}'$. The same idea applies if $d_k \in \mathcal{D}'$. We denote the chain of arrivals by $C(\mathcal{A}')$ and the chain of departures by $C(\mathcal{D}')$.

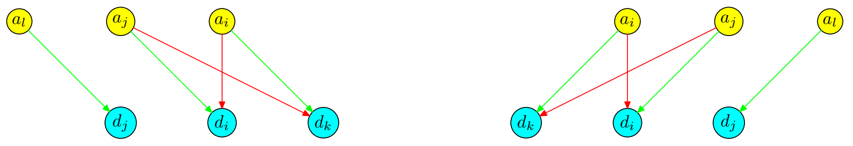


Figure 9: Using assignments from the sequential matching

It follows from our construction that there is exactly one element a_l in $C(\mathcal{A}')$ which does not belong to $\mathcal{A}'$ and exactly one element d_k in $C(\mathcal{D}')$ which does not belong to $\mathcal{D}'$. We will substitute the assignments involving $C(\mathcal{A}')$ and $C(\mathcal{D}')$ in order to obtain a solution x' with the property we want, and the substitution is done according to one of the following four possible situations:

1. a_l is the first arrival of $C(\mathcal{A}')$ and d_k is the first departure of $C(\mathcal{D}')$: We assign d_k to the first arrival of $C(\mathcal{A}') \setminus \{a_l\}$, a_l to the first departure of $C(\mathcal{D}') \setminus \{d_k\}$, and the remaining elements of $C(\mathcal{A}')$ to the remaining elements of $C(\mathcal{D}')$ according to the sequential matching.
2. a_l is the last arrival of $C(\mathcal{A}')$ and d_k is the last departure of $C(\mathcal{D}')$: We assign d_k to the last arrival of $C(\mathcal{A}') \setminus \{a_l\}$, a_l to the last departure of $C(\mathcal{D}') \setminus \{d_k\}$, and the remaining elements of $C(\mathcal{A}')$ to the remaining elements of $C(\mathcal{D}')$ according to the sequential matching.
3. a_l is the first arrival of $C(\mathcal{A}')$ and d_k is the last departure of $C(\mathcal{D}')$: We assign d_k to the first arrival of $C(\mathcal{A}') \setminus \{a_l\}$, a_l to the first departure of $C(\mathcal{D}') \setminus \{d_k\}$, and the remaining elements of $C(\mathcal{A}')$ to the remaining elements of $C(\mathcal{D}')$ according to the sequential matching.
4. a_l is the last arrival of $C(\mathcal{A}')$ and d_k is the first departure of $C(\mathcal{D}')$: We assign d_k to the last arrival of $C(\mathcal{A}') \setminus \{a_l\}$, a_l to the last departure of $C(\mathcal{D}') \setminus \{d_k\}$, and the remaining elements of $C(\mathcal{A}')$ to the remaining elements of $C(\mathcal{D}')$ according to the sequential matching.

The last two cases are presented in Figure 9. If follows from a similar argument used in the proof of Proposition 2.29 that, with these transformations, we obtain a new solution x' with the same number of crossings involving units of different types.

We conclude that we can keep only the variables representing the assignments of the sequential matching of $\mathcal{A}'$ to $\mathcal{D}'$ without losing all the feasible optimal solutions. □

These results suggest that it is not necessary to distinguish elements in a uniform consecutive subsequence. For each type $\mathcal{T}_i \in \mathcal{T}$, we can define a multi-graph $H^i = (V, E)$ where each vertex represents a uniform consecutive

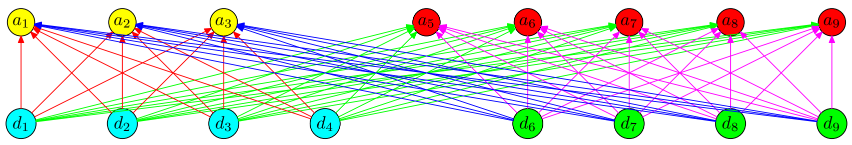


Figure 10: Original problem: 64 possible assignments

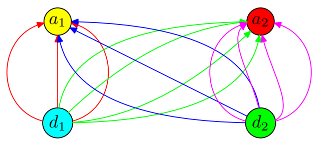


Figure 11: Graph H^i: 14 possible assignments

subsequence (of arrivals or departures) of type $\mathcal{T}_i$. An edge of H^i represents a possible assignment of a vertex a representing a sequence of arrivals to a vertex d representing a sequence of departures. If we denote the size of these sequences by $|a|$ and $|d|$, there will be $e(a,d) = min(|a|,|d|)$ edges connecting a to d. Figures 10 and 11 present an example of this construction. The first figure shows the parts of the original sequences $\mathcal{A}$ and $\mathcal{D}$ containing elements of type $\mathcal{T}_i$, and the second shows how the respective graph H^i looks like.

It is possible to use graphs H^i, $1 \leq i \leq t$, in order to construct a new ILP formulation for the VPP. For each pair (a,d), we have variables $x^1_{a,d}, x^2_{a,d}, \ldots,$ $x^{e(a,d)}_{a,d}$ representing the assignments of arrivals in a to departures in d. Clearly, if there is no sequence of consecutive units, the problem remains the same.

An integral solution can describe the assignment of $k' < e(a,d)$ elements of a to d in $\binom{e(a,d)}{k'}$ different ways, which brings more symmetry to the problem. In order to avoid this problem, we can use the following family of inequalities:

$$x^{j+1}_{a,d} \leq x^j_{a,d} \qquad a,d \in V(H^i), 1 \leq j < e(a,d), 1 \leq i \leq t. \tag{5}$$

Moreover, this technique also makes the the linearization of the cost function more challenging. The formulation used in (**LU**) is valid because each a and each d is assigned to exactly one d and a, respectively, and as a consequence, the sum on the left term is not greater than 2. However, because a variable in the new model represents a sequence, the number of assignments involving a pair (a,d) lies between 0 and $min(|a|,|d|)$, so the sums $\sum_{a'<a} x_{a',s,d}$ and $\sum_{d' \leq d} x_{a,s,d'}$ can be greater than one.

For this reason, we have to use the following constraints in this problem, which are based on the set of inequalities proposed in Section 2.2.2:

$$\sum_{\substack{(a',d')\dagger(a,d)\\1\le j\le e(a',d')}} x^j_{a',s,d'} \le \beta(1-x^i_{a,s,d}) + r_{a,s,d,i} \qquad \substack{(a,s,d)\in\mathcal{A}\times\mathcal{S}\times\mathcal{D}\\1\le i\le e(a,d)}.$$

We remark that variables $r_{a,s,d,i}$ are positive and integer. Unfortunately, as we saw in Section 2.2.2, these inequalities are very challenging from a computational point of view.

Due to several symmetry issues appearing in this formulation, this model does not seem to be a competitive option for the VPP.

2.4.2.3 General Consecutive Subsequences

Let x be a solution of an instance of VPP. Let us suppose that (a_i, d_j) is the u-th unit assigned to track s_m and that (a_k, d_l) is the v-th unit assigned to track s_n in x. In some cases, it is possible to modify x by assigning (a_i, d_j) to s_n and (a_k, d_l) to s_m, obtaining a solution x' with the same number of crossings as x. Therefore, x and x' are basically equal, so it is interesting to remove one of these possible scenarios from the space of feasible solutions.

For the minimization of crossings, there seems to be no easy way to formulate inequalities that avoid such scenarios. For the minimization of first crossings, though, the situation is more simple. Let $c(M)$ be the number of crossings involving units of a set $M \subseteq \mathcal{A} \times \mathcal{D}$. Suppose that $M_1 = \{(a_i, d_j), (a_k, d_l), (a_m, d_n)\}$ contains units assigned to track s_1 in consecutive positions, that $M_2 = \{(a_o, d_p), (a_q, d_r), (a_s, d_t)\}$ contains units assigned to track s_2 in consecutive positions, and that $a_i, a_o < a_k, a_q < a_m, a_s$. We have the situation described in the first paragraph of this section if

$$c(M_1)+c(M_2) = c(\{(a_i, d_j), (a_q, d_r), (a_m, d_n)\})+c(\{(a_o, d_p), (a_k, d_l), (a_s, d_t)\}).$$

It is possible to create families of inequalities to eliminate such scenarios, but they are clearly too numerous and too weak to make their incorporation to any ILP formulation worthy. However, we will see in Chapter 4 that this idea can be incorporated to a branch-and-price algorithm, bringing a significative improvement in its performance.

2.4.2.4 Applicability

The symmetry breaking techniques presented in this section do not yield significative gains in performance if directly applied and/or implemented in the formulation of the VPP presented in this chapter. For example, we applied Algorithm 3 to remove the track symmetry from some instances of the VPP and noticed that the resulting programs were slower than the original ones.

However, as we will see in future chapters (specially Chapter 4), these observations are useful if we want to enhance the performance of more advanced "tailor-made" techniques for the VPP.

2.4.3 Computational Complexity

In this section, we explore some aspects of the computational complexity of the VPP which were not addressed in the literature so far to the best of our knowledge.

2.4.3.1 Hardness of VPP^P

The VPP^P is an extension of the VPP, which is known to be a hard problem (Winter (1998) [91]). The following results show that the VPP^P, not surprisingly, is also hard.

Proposition 2.32. *It is $\mathcal{NP}$-complete to decide if an instance of the* VPP^P *needs shuntings.*

Proof. Let I be an instance of VPP with arrival sequence $\mathcal{A}$, departure sequence $\mathcal{D}$, and m tracks. Based on I, we can construct an instance I' of VPP^P with $k > 1$ periods. The first period is such that $\mathcal{A}_1 = \mathcal{A}$, $\mathcal{D}_1 = \mathcal{D}$, and the number of tracks is also equal to m. In other words, the first period of I' is equal to I. The other periods are such that both the arrival and the departure sequences are equal to $\mathcal{A}$. We are assuming that all the periods have the same number of scheduled trips, i.e., $e = n$. Figure 12 shows an example of this reduction where $k = 3$, $n = 6$, and $m = 2$.

It is clear that there are crossing-free solutions for the periods k such that $\mathcal{A}_k = \mathcal{D}_k = \mathcal{A}$. Conversely, the first period, which is similar to I, is non-trivial. Consequently, we are able to obtain an optimal solution for I' if and only if we are able to obtain an optimal solution for I. Moreover, both solutions will have the same objective value. Therefore, as we know that the VPP is $\mathcal{NP}$-complete, it follows that the VPP^P is also $\mathcal{NP}$-complete. $\square$

Corollary 2.33. *The* VPP^P *is $\mathcal{NP}$-hard.*

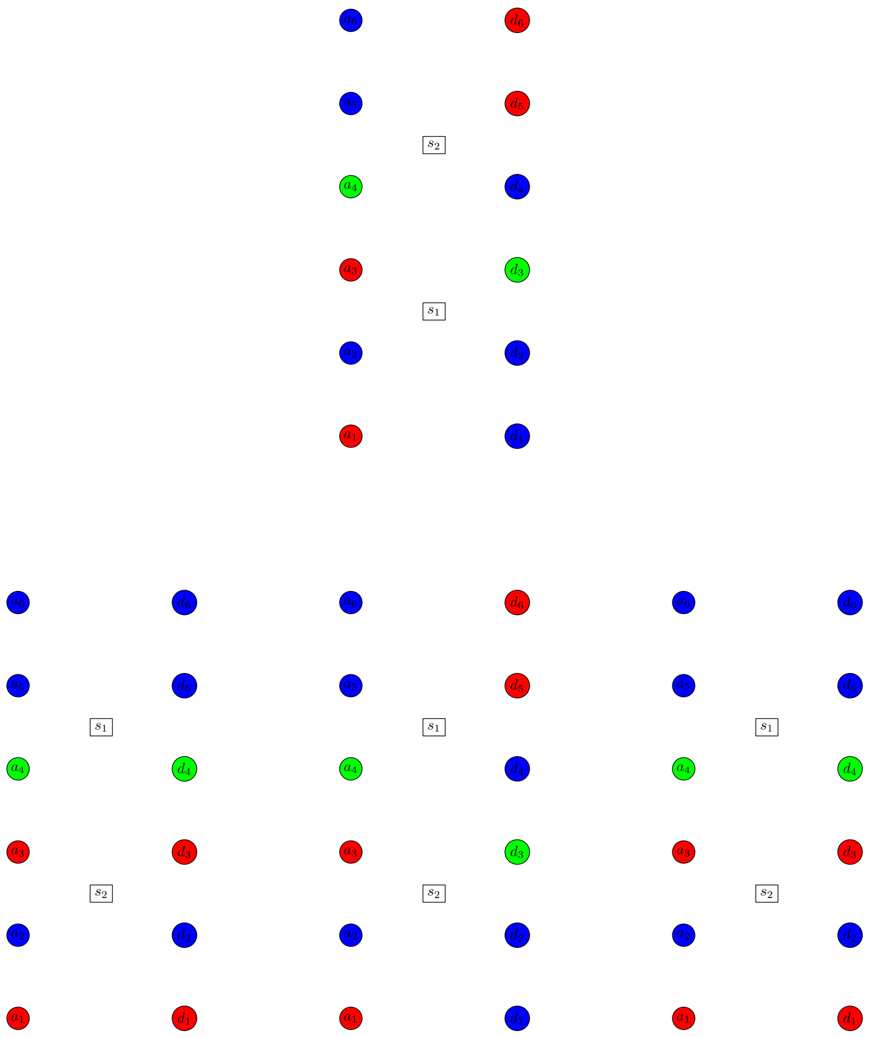


Figure 12: Reducing an instance of VPP to an instance of VPP^P

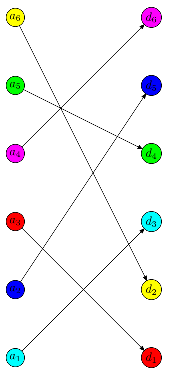


Figure 13: Original VPP instance

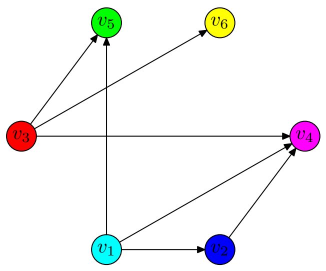


Figure 14: Resulting digraph

Proof. It follows from Proposition 2.32. □

Positive results regarding special scenarios of the VPP also hold for the VPP^P.

Corollary 2.34. *It can be decided in polynomial time if the* VPP^P *needs shuntings when all tracks have size two and the assignments of vehicles to trips are fixed.*

Proof. The problem can be reduced to the Maximum Matching Problem (as in Winter (1998) [91]), where each unit is represented by a vertex and each edge represents a crossing-free configuration composed by the units associated to the vertices. In Figures 13 and 14, we show how this reduction works for an instance of VPP.

Initially, we notice that almost every period can be solved independently of the others, and in these cases, it is clear that there is a crossing-free assignment if and only if there is a perfect matching in the graph described above. The only exception involves the first weekday period and the first weekend period. Some departures of the first period (more precisely, exactly $n-e$ of them) must be serviced by arrivals of the second period. Fortunately, the only implication of this fact is that the assignment of both periods must be considered simultaneously, so it is also clear in this case that there is a crossing-free assignment if and only if there is a perfect matching.

Each graph in this construction has $O(n)$ vertices, and it is a well-known fact that it is possible to decide in polynomial time if a graph admits a perfect matching (see Edmonds (1965) [31]). As a consequence, we conclude that it is possible to decide in polynomial time if the VPP^P needs shuntings when all tracks have size two and the assignment of vehicles to trips is fixed. □

2.4.3.2 Fixed Number of Types and Non-uniform Tracks

Hamdouni, Soumis & Desaulniers (2005) [52] investigated the Bus Dispatching Problem, and in order to define and compute robust solutions, they introduced the concept of *uniform tracks*. If a track is uniform, it clearly does not contain crossings. Kroon, Lentink & Schrijver (2006) [70] explored this idea in the context of the Train Unit Shunting Problem and obtained integer programs which are smaller and significatively easier from a computational point of view. These works were the first to show the importance of uniform tracks for the VPP.

The following results show that the VPP is easier from a theoretical point of view if the number t of vehicle types and the number of non-uniform tracks are fixed.

Theorem 2.35. *It is possible to decide in polynomial time if an instance of the* VPP *needs crossings if t is constant and if only solutions containing at most one non-uniform track are allowed.*

Proof. Initially, we observe that $\sum_{i=1}^{t}(t_i \mod \beta)$ is a multiple of β. If $t_i \mod \beta = 0$ for every type $\mathcal{T}_i$, $1 \leq i \leq t$, then it is possible to assign all the vehicles to uniform tracks and the problem is trivial. If $\sum_{i=1}^{t}(t_i \mod \beta) > \beta$, then more than one non-uniform track is necessary. Consequently, we can assume that $\sum_{i=1}^{t}(t_i \mod \beta) = \beta$.

If only one non-uniform track is allowed, then the decision problem can be reduced to the identification of a crossing-free configuration containing $t_i \mod \beta$ vehicles of each type $\mathcal{T}_i$, $1 \leq i \leq t$. Let a and d be strings of characters representing sequences $\mathcal{A}$ and $\mathcal{D}$, respectively, where character a_i (d_i) represents type $\tau(a_i)$ ($\tau(d_i)$). The identification of a crossing-free configuration in $\mathcal{A}$ and $\mathcal{D}$ covering $t_i \mod \beta$ vehicles of each type $\mathcal{T}_i$ is similar to the identification of a common subsequence of a and d containing $t_i \mod \beta$ symbols representing type $\mathcal{T}_i$, $1 \leq i \leq t$. This problem is similar to the Longest Common Subsequence Problem, which is known to be solvable in polynomial time (Cormen et al. (2001) [23]). We give a direct proof that this variant can be solved efficiently with a dynamic program.

The state space of our dynamic program is the following:

$$\mathcal{S} := \{0, \dots, n\}^2 \times \{0, \dots, \beta\}^t.$$

The first two coordinates represent the position on strings a and d, respectively, while the other t positions represent how many of the $t_i \mod \beta$ units of type $\mathcal{T}_i$ still have to be inserted in the common subsequence.

The algorithm has to check which states are accessible and bookkeep the sequence by saving an antecessor for each state. Initially, the only reachable state is $(0, 0, t_1 \mod \beta, t_2 \mod \beta, \ldots, t_t \mod \beta)$. If tuple $(i, j, t'_1, t'_2, \ldots, t'_t)$ is reached, then, for every $k \geq i$ and $l \geq j$, the same holds for $(k, l, t'_1, t'_2, \ldots, t'_t)$. In addition to that, if $\tau(a_i) = \tau(d_j)$ and $t'_{\tau(a_i)} > 0$, the pair (a_i, d_j) can be added to the current configuration, and state $(i + 1, j + 1, t'_1, t'_2, \ldots, t'_{\tau(a_i)} - 1, \ldots, t'_t)$ is marked as reached.

Each state might be reached several times, but only one of the predecessors needs to be saved. Moreover, we just have to examine the successors of a given reached state once. Finally, the iteration of the algorithm follows the order of the indices, i.e., the more external loop iterates on the indices for arrivals, the next on the indices for departures and so on.

Finally, there is a configuration which contains all the $t_i \mod \beta$ elements, $1 \leq i \leq t$, if and only if the state $(n, n, 0, 0, \ldots, 0)$ is reached by the end of the algorithm.

Because each possible state will be analyzed at most once, t is constant, and there are $|\{0, ..., n\} \times \{0, ..., n\} \times t_1 \times t_2 \times \ldots \times t_t| = O(n^2 \beta^t)$ different states, the algorithm consumes polynomial time. □

2.4.3.3 Fixed Assignment $\mathcal{A} \to \mathcal{D}$

It is possible to identify in the literature versions of the VPP where the authors assume that the assignments of arrivals to departures are already fixed, i.e., that $t_i = 1$ for each vehicle type $\mathcal{T}_i$, $1 \leq i \leq t$.

Cornelsen & Di Stefano (2007) [24] present a theoretical investigation of some versions of the VPP with this characteristic. Lentink (2006) [72] proposes a two-step method to solve the Train Unit Dispatching Problem where all the assignments $\mathcal{A} \to \mathcal{D}$ are determined in the first phase, which makes the second phase (the assignments of units to tracks) easier.

It is possible to make an intuitive graphical description of the VPP-graph $G = (V, E)$ for such scenarios. If we draw the arrivals and departures in two parallel lines according to the order in which they appear in $\mathcal{A}$ and $\mathcal{D}$ and draw lines between elements composing a unit, each line will represent a vertex in V, and for each pair of crossing lines, there will be an edge in E.

We refer to this representation as the *$\mathcal{AD}$-drawing* of G. In Figures 15 and 16, we give an example of an $\mathcal{AD}$-drawing of an instance of the VPP and its VPP-graph, respectively. Graphs admitting this representation are called *permutation graphs*.

There is a connection between the number of crossings in this special case of the VPP and the *crossing number* of the associated $\mathcal{AD}$-drawings. The crossing number of a graph G denotes the lower bound for the number of edge crossings for every planar drawing of G. In our case, the drawing is fixed (it is the $\mathcal{AD}$-drawing) and two edges cross if and only if there is a semi-crossing involving the associated units.

Proposition 2.36. *Given an instance I of the* VPP *with fixed assignment $\mathcal{A} \to \mathcal{D}$, if the crossing number of its $\mathcal{AD}$-drawing is smaller than the number of tracks, then I admits a crossing-free matching.*

Proof. We prove by induction that it is possible to assign the set C of conflicting units to tracks in such a way that crossings are avoided. If $m = 1$, the problem is trivial, and if $m = 2$, we have at most one crossing. In this case, we just have to assign each one of the involved units in a track and we are done.

Suppose that $m > 2$. Let u be the unit involved in the largest number of semi-crossings. If we assign u to the first track together with all the units of C which are not in conflict with u, we just have to assign the elements involved in at most $m - 2$ crossings in the remaining $m - 1$ tracks, which is possible from the induction hypothesis. □

The proposition above provides an interesting result, but its application is limited, as instances of the VPP whose $\mathcal{AD}$-drawings have small crossing number are not common. Moreover, we remark that a crossing-free assignment may be possible even for instances whose associated crossing number is large.

Proposition 2.37. *There are instances of the* VPP *with fixed assignment $\mathcal{A} \to \mathcal{D}$ which admit crossing-free matchings and whose associated $\mathcal{AD}$-drawing have $\frac{n(n-\beta)}{2} = \frac{n(m-1)\beta}{2}$ crossings.*

Proof. Consider the family of instances of the VPP such that the i-th arrival is assigned to the j-th departure for $j = (m - \lceil i/\beta \rceil + 1)\beta + i \mod \beta$. In other words, if $\mathcal{A}$ is divided into m subsequences U_i of size β and the order of appearance of these subsequences is inverted, we obtain the description of $\mathcal{D}$.

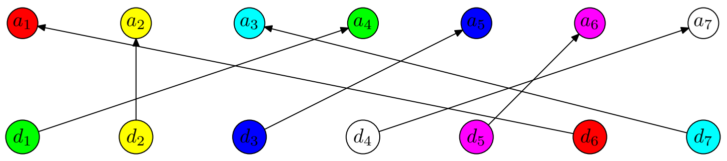


Figure 15: Example of $\mathcal{AD}$-drawing

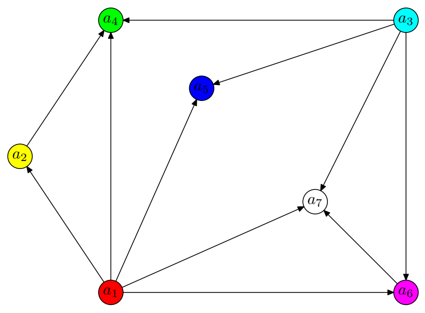


Figure 16: Permutation graph

It is clear that instances of this family admit a crossing-free matching, as we just have to assign units belonging to the same subsequence U_i to the same track. It is also clear that if unit u belongs to the subsequence U_i, then u is involved in a semi-crossing with every unit belonging to any other subsequence $U_j, j \neq i$. Therefore, each unit is involved in $(n-\beta) = (m-1)\beta$ crossings and the crossing number of the $\mathcal{AD}$-drawing is equal to $\frac{n(n-\beta)}{2} = \frac{n(m-1)\beta}{2}$. □

Deciding if there is an assignment of units to tracks without crossings is similar to deciding if it is possible to partition the set of vertices V into m independent sets of size β. This problem was introduced in the literature by Bodlaender & Jansen (1995) [13] and it is known as the Partition into Bounded Independent Sets Problem, which is defined for every type of graph. Another correlated problem is the Scheduling with Incompatible Jobs Problem, which was investigated by Bodlaender, Jansen & Woeginger (1992) [14].

The Partition into Bounded Cliques Problem is $\mathcal{NP}$-complete (Bodlaender & Jansen (1995) [13]) for cographs (or P_4-free graphs), and the following result is a corollary of this fact.

Corollary 2.38. *The* VPP *is $\mathcal{NP}$-complete even if the assignment $\mathcal{A} \to \mathcal{D}$ is fixed.*

Proof. Since the complement of a cograph is also a cograph, the Partition into Bounded Independent Sets Problem is also $\mathcal{NP}$-complete. As cographs constitute a subclass of permutation graphs, we can conclude that the Partition into Bounded Independent Sets Problem for permutation graphs, which is equivalent to the VPP with fixed assignment $\mathcal{A} \to \mathcal{D}$, is $\mathcal{NP}$-complete. $\square$

Some results suggest that fixing additional parameters can make the VPP easier from a theoretical point of view. For example, Bodlaender & Jansen (1995) [13] show that the Partition into Bounded Independent Sets Problem can be solved in polynomial time for cographs if one of the parameters (i.e., the size of the sets or the number of sets) is fixed. Also interesting is the result from Grohe (2004) [43], which says that it is possible to decide in polynomial time whether a given graph has crossing number at most k for every fixed $k > 0$.

For the VPP, if the number of tracks is fixed, then so should be the number of independent sets in the associated permutation graph. The following proposition shows that, in this case, the VPP can be solved in polynomial time.

Proposition 2.39. *It is possible to decide in polynomial time if an instance of* VPP *requires crossings if m is constant and the assignment $\mathcal{A} \to \mathcal{D}$ is fixed.*

Proof. We present a dynamic program to solve this problem whose state space is the following:

$$\mathcal{S} := \{\{0, \dots, n\} \times \{0, \dots, \beta\}\}^m.$$

Each state represents the number of vehicles parked in each track and the last unit assigned to it, which gives us a total of $O((n\beta)^m)$ different possibilities. With these informations, it is possible to know the next unit that will be assigned and if a first crossing will occur if we insert it in any track.

In this dynamic program, each state is visited at most once. We start with the "empty space", which represents the scenario where all the tracks are still empty. In the i-th iteration, we have to insert unit u_i (the order of the units is defined by the order of theirs arrivals) in some track whose last assigned unit does not cross with u_i. Because there might be many possibilities in each step and we do not know in advance which track should be chosen, we have to consider each state that was reached after the insertion of unit u_{i-1} and try to assign u_i to each track. The resulting crossing-free states are marked and we proceed with the next units. The procedure stops when the last unit is parked or when we identify a vehicle u_i that can not be assigned to a track without crossing.

It is clear that the last unit can be assigned to some track in the last iteration if and only if the instance of VPP does not require shuntings. Finally, the time consumption is $O(n^{4m})$, so it is possible to decide in polynomial time if an instance of VPP such that $\mathcal{A} \to \mathcal{D}$ and m are fixed requires shuntings. □

Corollary 2.40. *It is possible to find in polynomial-time a solution for an instance of the* VPP *with the minimum number of first crossings if m is constant and the assignment $\mathcal{A} \to \mathcal{D}$ is fixed.*

Proof. We can use an algorithm similar to the one employed in the proof of Proposition 2.39 to compute an optimal solution for such an instance of the VPP.

For each state c in $\{\{0, \dots, n\} \times \{0, \dots, \beta\}\}^m$, we bookkeep a preceding state $ant(c')$ that minimizes the number $f(c)$ of first crossings necessary to reach c, which is computed based on $f(c')$ as follows:

$$f(c) = \begin{cases} f(c') & \text{if } u_i \text{ is not involved in a first crossing;} \\ f(c') + 1 & \textit{else.} \end{cases}$$

The algorithm has n iterations. In the i-th iteration, we take each of the states representing scenarios where the first $i - 1$ units were assigned to parking positions and generate the states where u_i is inserted in each possible track. After the completion of the i-th step, we will have a pool of states that will be visited in the next iteration, and for each of these states c we can use the predecessors determined so far in order to determine a feasible set of (incomplete) configurations with $f(c)$ first crossings.

After the last iteration, we clearly have an optimal solution for the problem. The time consumption is also bounded by $O(n^{4m})$, which gives us a polynomial time algorithm to find a solution for an instance of VPP such that $\mathcal{A} \to \mathcal{D}$ and m are fixed with the minimum number of first crossings. □

We remark that the previous results are interesting only from a theoretical point of view, as the state space and the time consumptions are way too large for practical applications.

If β and $\mathcal{A} \to \mathcal{D}$ are fixed, the VPP becomes equivalent to the Mutual Exclusion Problem (**MES**) restricted to permutation graphs. The **MES** is a classical scheduling problem, which is basically the minimum coloring of vertices in a graph such that each color is used at most k times for fixed k. For permutation graphs, deciding **MES** for $k = 3$ is similar to deciding if a sequence of $3n$ distinct positive integers can be partitioned into n increasing subsequences of size 3. If $k = 2$, we have a matching problem that can be solved in polynomial time. Jansen (2003) [59] shows that the **MES** is $\mathcal{NP}$-complete if k is a fixed constant bigger or equal to 6, which implies the following corollary.

Corollary 2.41. *The* VPP *is $\mathcal{NP}$-complete if $\beta \geq 6$ and the assignments $\mathcal{A} \to \mathcal{D}$ are fixed.*

The problem remains open for k between 3 and 5.

2.4.3.4 Enlarging the Depot

We have assumed so far that the number of parking positions is equal to the number of vehicles, i.e., $\mu = n$. However, there are scenarios in the real world where depots can still have free positions after the arrival of all the vehicles.

Winter (1998) [91] shows how to modify (**W**) and (**LW**) in order to consider scenarios where $\mu > n$. Basically, Equalities (**W**)(i), (**W**)(iii), (**LW**)(i), and (**LW**)(iii) just have to become inequalities. Clearly, the same procedure can be applied to Equality (**LU**)(iii) in order to adjust (**LU**).

There is a number of interesting questions arising when we consider possible expansions of the depot. The following proposition shows that the addition of one parking position to one of the tracks of the depot can reduce the number of crossings and first crossings in an optimal solution for an instance of the VPP.

Proposition 2.42. *There are instances of the* VPP *for which the addition of one parking position to one of the tracks leads to a new instance that admits solutions with less crossings and less first crossings.*

Proof. We present a family of instances of the VPP for which the addition of a parking position to one of the tracks eliminates β crossings and 1 first crossing from the optimal solution. Assume that the first β arrivals belong to pairwise different types, and that the remaining $n - \beta$ arriving vehicles belong to type $\tau(a_\beta)$. Assume that the first $n - \beta + 1$ departures are of type

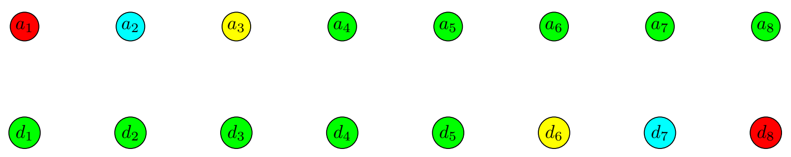


Figure 17: Example with 8 arrivals and departures and $\beta = 4$

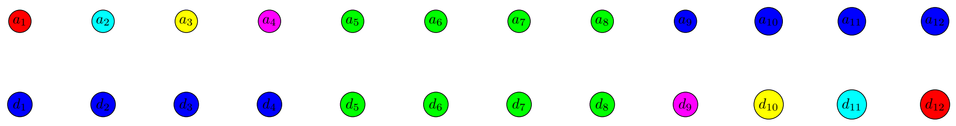


Figure 18: Example with 12 arrivals and departures and $\beta = 4$

$\tau(a_\beta)$, while $\tau(d_{n-k+1}) = \tau(a_k)$, $1 \leq k \leq \beta$. Figure 17 shows an example for $\beta = 4$ and $n = 8$.

If $n = \mu$, an optimal solution for both cases is composed of $\frac{n-\beta}{\beta}$ uniform tracks of type $\tau(a_\beta)$ and one non-uniform track containing the remaining units. This solution has $\binom{\beta}{2}$ crossings and β first crossings.

If a parking position is added to one of the tracks, arrival a_β can be assigned to a uniform track, i.e., the track with $\beta+1$ parking positions will be uniform with units of $\tau(a_\beta)$. This modification leads to a solution with $\binom{\beta}{2} - \beta$ crossings and $\beta - 1$ first crossings. □

Conversely, the following proposition shows that the expansion of existent tracks may not help to reduce the number of crossings or first crossings.

Proposition 2.43. *There are instances of the* VPP *for which the addition of an arbitrary number of parking positions to the existing tracks leads to new instances whose optimal solutions require the same number of crossings and first crossings.*

Proof. We present a family of instances of VPP for which the addition of parking positions does not reduce the number of crossings or first crossings. Let $t = m - 1 + \beta$. Assume that the first β arrivals belong to pairwise different types (say, types $\mathcal{T}_1, \ldots, \mathcal{T}_\beta$) and that $\tau(a_i) = \mathcal{T}_{\beta + \lfloor i/\beta \rfloor}$, $\beta + 1 \leq i \leq n$. The last β departures will be such that $\tau(d_{n-\beta+i}) = \tau(a_{\beta-i+1})$ and $\tau(d_i) = \mathcal{T}_{m+\beta-1-\lfloor i/\beta \rfloor}$, $1 \leq i \leq n - \beta$. Figure 18 shows an example for $\beta = 4$ and $n = 12$.

If $n = \mu$, an optimal solution is composed of $m - 1$ uniform tracks, one for each type $\mathcal{T}_i, i > \beta$, and one non-uniform track containing the remaining units. This solution has $\binom{\beta}{2}$ crossings and β first crossings.

If one or more vehicles of the non-uniform track are moved to one of the uniform tracks, the number of crossings and first crossings do not decrease. The same holds if we assign units from the uniform tracks to other tracks.

Consequently, the addition of parking positions to the existing tracks does not lead to instances that admit solutions with less crossings or first crossings. □

The addition of tracks is more powerful. With an additional track, we can always reduce the number of crossings and the number of first crossings. For example, if we add a track to the instances of the family described in the proof of Proposition 2.43, then it is possible to reduce the number of first crossings to $\beta - 1$ and the number of crossings to $\binom{\lfloor \beta/2 \rfloor}{2} + \binom{\lceil \beta/2 \rceil}{2}$ by distributing the units assigned to the non-uniform track in two tracks, one containing $\lfloor \beta/2 \rfloor$ vehicles and the other $\lceil \beta/2 \rceil$.

One natural question that arises is: How many tracks should be added to a depot in order to make an instance of the problem trivial? In other words, how many tracks should we add to the problem in order to make solutions without crossings possible?

Clearly, if we add $n - m$ tracks to the depot, each vehicle can be parked in its own track, which makes the depot similar to a parking lot for cars, trivializing the problem. As the following proposition shows, this bound is the best we can have in the general case.

Proposition 2.44. *There are instances of the* VPP *that can only be trivialized with the addition of $n - m$ tracks.*

Proof. Consider the family of instances of VPP that contain no pair of arrivals (departures) of the same type and such that $\tau(a_i) = \tau(d_{n-i+1})$, $1 \leq i \leq n$. Because there is a semi-crossing involving each pair of units (a, d) and (a', d'), a crossing-free solution can only be obtained if each vehicle is assigned to a track where it stays alone. In other words, these instances can only be trivialized if we add $n - m$ tracks to the depot. □

The family of instances presented in the proof of Proposition 2.44 represents the worst case scenario for the VPP, which can be considered unrealistic. Therefore, we generalize this result in the following proposition, obtaining a more reasonable instance-independent bound.

Lemma 2.45. *Let $\mathcal{T}'$ be the subset of $\mathcal{T}$ containing types $\mathcal{T}_i$ such that $t_i \mod \beta \neq 0$, $1 \leq i \leq t$. If*

$$\sum_{i=1}^{t} \left\lfloor \frac{t_i}{\beta} \right\rfloor + |\mathcal{T}'| - m$$

tracks are added to the depot, then it is possible to obtain a crossing-free solution for the problem.

Proof. Initially, we show the sufficiency. A solution for an instance of VPP which contains only uniform tracks does not have crossings, and such a solution is feasible if there are

$$m = \sum_{i=1}^{t} \left\lceil \frac{t_i}{\beta} \right\rceil = \sum_{i=1}^{t} \left\lfloor \frac{t_i}{\beta} \right\rfloor + |\mathcal{T}'|$$

tracks in the depot; $\sum_{i=1}^{t} \left\lfloor \frac{t_i}{\beta} \right\rfloor$ tracks will be completely filled, and for each type $\mathcal{T}_i \subseteq \mathcal{T}'$ there will be a track containing its $t_i \mod \beta$ "remaining" units. Therefore, if we add $\sum_{i=1}^{t} \left\lfloor \frac{t_i}{\beta} \right\rfloor + |\mathcal{T}'| - m$ tracks to the depot, there will be a crossing-free solution for the problem.

Now we show that this bound is tight. Consider the family of instances of VPP such that the arrivals of the same type are contiguous in $\mathcal{A}$, i.e., the first t_1 arrivals are of type $\mathcal{T}_1$, the next t_2 are of type $\mathcal{T}_2$ and so on, and $\mathcal{D}$ is the inversion of $\mathcal{A}$, i.e., the last t_1 departures are of type $\mathcal{T}_1$, preceeded by t_2 departures of type $\mathcal{T}_2$ and so on. In this scenario, the assignment of units of different types to the same track results in crossings. Consequently, crossing-free solutions can only be obtained with the addition of $\sum_{i=1}^{t} \left\lfloor \frac{t_i}{\beta} \right\rfloor + |\mathcal{T}'| - m$ tracks to the depot. □

Again, the bounds given by Propositions 2.44 and 2.45 are tight but conservative, as they are enforced by worst-case scenarios. It is possible to obtain better results if we explore the structure of the instances. The following proposition is based on Remark 2.16, that is, on the fact that the minimum number of first crossings is a non-trivial lower bound for the number of crossings for any instance of the VPP.

Proposition 2.46. *If the minimum number of first crossings is k, then it is sufficient to add k tracks in order to obtain a crossing-free solution.*

Proof. Consider an optimal solution of some instance I of VPP with k first crossings. Let (a, d) and (a', d') be crossing pairs assigned to consecutive positions p and q of track s with $p < q$. If we keep the assignments of the first p positions of s to this track and assign the remaining pairs (starting from (a', d')) to a new track, we obtain a new solution x' with k-1 first crossings. If we do this sequentially for every first crossing, after k steps there will be no first crossings (and, consequently, no crossing) in the solution. It follows that the instance derived from I with k additional tracks admits a crossing-free solution. $\square$

Chapter 3

A Binary Quadratic Programming Approach

The Vehicle Positioning Problem (VPP) has a natural formulation as a mixed integer quadratically constrained program. This MIQCP is closely related to the Quadratic Assignment Problem and, as far as we know, has not received any attention yet. We show in this chapter that such a formulation has interesting theoretical properties. Its QP relaxation produces, in particular, the first known nontrivial lower bound on the number of crossings. In our experiments, it also outperformed the integer linear models computationally. The strengthening technique that raises the lower bound might also be useful for other combinatorial optimization problems.

3.1 Introduction

The first models for the VPP were introduced by Winter (1998) [91] and Winter & Zimmermann (2000) [93], who modeled the VPP as a Quadratic Assignment Problem and used linearization techniques to solve it as an integer linear program.

Although the VPP was originally modeled as a binary quadratic program, this formulation was explored neither theoretically nor computationally. All research efforts that we are aware of concentrated on integer linear models that used more and more indices in order to produce tighter linearizations. Recent progress in Mixed Integer Nonlinear Programming (MINLP) and, in particular, in Mixed Integer Quadratically Constrained Programming (MIQCP) methods (Nowak (2005) [77]), however, has increased the attractivity of the original quadratic model. Besides the compactness of the formulations, quadratic programming models also yield potentially superior lower bounds from fractional quadratic programming relaxations. In fact, the LP relaxations of all known integer linear models for the VPP are not able to yield non-trivial lower bounds for every instance requiring crossings.

We investigate in this chapter binary quadratic programming formulations for the VPP. Our main result is that the QP relaxation of one of these models yields a nontrivial lower bound on the number of crossings, that is, the fractional QP lower bound is nonzero whenever shunting is required. Specially interesting is the penalization of inconsistent assignments employed in this formulation, a technique that appears to be easily applicable to other problems and which was very useful here. This model also had the best computational performance in our tests, even though it is not convex. We also tried to apply the convexification techniques of Hammer & Rubin (1970) [53], but the results were mixed. Convexification helped, but only when the smallest eigenvalue of the objective function was not too negative.

The chapter is organized as follows. Section 3.2 discusses integer linear and integer quadratic two-index models, i.e., we revisit the original formulation of Winter. In Section 3.3 we present integer linear and integer quadratic three-index models. One of them produces the already mentioned non-trivial QP bound.

All our computational experiments were done on an Intel(R) Core 2 Quad 2660 MHz with 4Gb RAM, running under openSUSE 11.1 (64 bits). We used `CPLEX` 11.2 (ILOG (2010) [57]) to solve linear programs, `SCIP` 1.0 for integer programs (Achterberg (2007) [2]), and `SNIP` 1.0 for integer non-linear programs (Vigerske (2009) [88]).

We thank Stefan Vigerske for his advice with respect to the formulation of quadratic integer programs and `SNIP` support.

3.2 Two-Index Models

The first formulation of the VPP was proposed by Winter & Zimmermann (2000) [93] and consists of a quadratic integer programming formulation for the problem. The model and its relevant details were presented in Subsection 2.2.1.1, so we just recall the formulation and its basic description here:

$$\begin{array}{lll}
(\mathbf{W}) & \min \displaystyle\sum_{(a,p)\dagger(a',q)} x_{a,p}x_{a',q} + & \displaystyle\sum_{(d,p)\dagger(d',q)} y_{d,p}y_{d',q} \\
\text{(i)} & \displaystyle\sum_a x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \displaystyle\sum_p x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \displaystyle\sum_d y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \displaystyle\sum_p y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \leq 1 & (a,p,d) \in \mathcal{A} \times \mathcal{P} \times \mathcal{D}, \tau(a) \neq \tau(d) \\
 & x_{a,p}, y_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}$$

The model uses binary variables $x_{a,p}$, with $a \in \mathcal{A}$ and $p \in \mathcal{P}$, and $y_{d,p}$, with $d \in \mathcal{D}$ and $p \in \mathcal{P}$. If $x_{a,p} = 1$ ($y_{d,p} = 1$), arriving vehicle a (departing trip d) is assigned to parking position p. Equations (**W**)(i)-(**W**)(iv) define the matching constraints and Inequalities (**W**)(v) enforce the coherence of these assignments by allowing only arrivals and departures of the same type to be assigned to a given parking position. Finally, the quadratic cost function calculates the number of crossings.

In order to solve the problem, Winter presented a linearization of (**W**) whose relaxation always yields a trivial lower bound (see Theorem 2.4). It is not difficult to modify Winter's proof in order to get a similar result for the QP relaxation of (**W**):

Theorem 3.1. *$V_{QP}(\boldsymbol{W}) = 0$ if $m > 1$.*

Proof. Let M be the set of assignments in $\mathcal{A} \to \mathcal{D}$ where each a_i is assigned to d_i (i.e., first arrival to first departure, second arrival to second departure, and so on) and where the assignment of the units (a_i, d_i) to the parking positions is made according to the following scheme, where each column represents a track:

(a_{n-m}, d_{n-m})	(a_{n-m+1}, d_{n-m+1})	$\dots$	(a_n, d_n)
$\vdots$	$\vdots$	$\vdots$	$\vdots$
(a_{m+1}, d_{m+1})	(a_{m+2}, d_{m+2})	$\dots$	(a_{2m}, d_{2m})
(a_1, d_1)	(a_2, d_2)	$\dots$	(a_m, d_m)

Namely, pair (a_i, d_i) is assigned to track $s_{(i-1) \bmod m}$, $1 \leq i \leq n$.

Such an assignment has no crossings. However, it is not always feasible for (**W**) because of type mismatches (cf. the coherence Equations (**W**)(v)). If the integrality of the variables is relaxed, though, and each pair (a_i, d_i) is assigned to the same relative position in each track, Equations (**W**)(v) can be satisfied. More precisely, if a pair (a_i, d_i) is assigned to the second position of some track, i.e., if $\lfloor (i-1)/m \rfloor = 1$, we fix $x_{a_i,p} = y_{d_i,p} = 1/m$ for each position $p \in \mathcal{P}$ which is the second position in some track, i.e., for each p such that $\lfloor (p-1)/m \rfloor = 1$. If $m > 1$, we have $x_{a,p} + y_{d,p} = \frac{2}{m} \leq 1$, i.e., Equations (**W**)(v) are satisfied. Since there are no crossings, the objective value is zero. □

One of the main challenges for (**W**) to be solved directly is that its objective function is not convex. This obstacle can be overcome using the following technique of Hammer & Rubin (1970) [53]. Initially, we observe that $\sum_{(a,p)\dagger(a',q)} x_{a,p}x_{a',q}$ can be written as x^TAx, where $A \in \{0,1\}^{n^2} \times \{0,1\}^{n^2}$ is the symmetric incidence matrix of all crossings involving pairs (a,p) and (a',p') in $\mathcal{A} \times \mathcal{P}$. If α is the minimum eigenvalue of A, we have

$$x^TAx = x^T(A - \alpha I)x + \alpha x^Tx. \tag{1}$$

As x is binary, this equation can be rewritten as

$$x^TAx = x^T(A - \alpha I)x + \alpha \sum_i x_i. \tag{2}$$

Finally, in our case, we have $\sum_i x_i = n$ for every feasible solution, that is,

$$x^TAx = x^T(A - \alpha I)x + \alpha n. \tag{3}$$

As $A - \alpha I$ is positive semidefinite, the function on the right is convex. The same idea works for the symmetric incidence matrix of all crossings involving pairs (d,p) and (d',p') in $\mathcal{D}\times\mathcal{P}$, i.e., $\sum_{(d,p)\dagger(d,q')} y_{d,p}y_{d,q'} = y^TA'y$. Moreover, $A' = A$. Then, the objective can be written as

$$x^TA'x - \alpha \sum_{(a,p)} (x_{a,p}^2 - x_{a,p}) + y^TA'y - \alpha \sum_{(d,p)} (y_{d,p}^2 - y_{d,p}). \tag{4}$$

Applying this substitution to (**W**), we obtain:

$$
\begin{array}{lll}
(\mathbf{CW}) & \min\ x^T A' x - \alpha \sum\limits_{(a,p)} (x_{a,p}^2 - x_{a,p}) + y^T A' y - \alpha \sum\limits_{(d,p)} (y_{d,p}^2 - y_{d,p}) & \\
\text{(i)} & \sum\limits_{a} x_{a,p} = 1 & p \in \mathcal{P} \\
\text{(ii)} & \sum\limits_{p} x_{a,p} = 1 & a \in \mathcal{A} \\
\text{(iii)} & \sum\limits_{d} y_{d,p} = 1 & p \in \mathcal{P} \\
\text{(iv)} & \sum\limits_{p} y_{d,p} = 1 & d \in \mathcal{D} \\
\text{(v)} & x_{a,p} + y_{d,p} \le 1 & {\scriptstyle (a,p,d) \in \mathcal{A}\times\mathcal{P}\times\mathcal{D} \atop \tau(a) \neq \tau(d)} \\
 & x_{a,p}, y_{d,p} \in \{0,1\} & a \in \mathcal{A}, d \in \mathcal{D}, p \in \mathcal{P}.
\end{array}
$$

Table 4 gives the results of a computational comparison among models (**W**), (**LW**), and (**CW**) on a test set of ten instances of small and medium sizes. The first column in these tables give the name t-m-β of the problem. Following our notation, t is the number of vehicle types, m is the number of tracks, and β is the number of parking positions per track. Based on these parameters, the arrival sequences $\mathcal{A}$ were randomly generated (i.e., the type of each vehicle was uniformly chosen among the t possibilities), while their respective departure sequences $\mathcal{D}$ were obtained after the application of Algorithm 1 in $\mathcal{A}$. The columns labeled Row, Col, and NZ give the number of constraints, variables, and non-zeros of the respective model. The numbers of rows and columns for the problems of model (**CW**) are the same as the ones for model (**W**). Columns Nod give the number of nodes in the search tree generated by the respective solver (`SCIP` with LP solver `CPLEX` for (**LW**) and `SNIP` for (**W**) and (**CW**)) and columns T give the computation time in seconds.

Comparisons of the computational performances of (**CW**) and (**W**) show that convexification led to an improvement, but not enough to outperform the linearized model (**LW**), in particular not on the larger instances.

3.3 Three-Index Models

In order to discuss some quadratic three-index models for the VPP, we present again the formulation (**LU**). For more details, we refer to Section 2.2.1.2.

			(**LW**)			(**W**)/(**CW**)			(**W**)			(**CW**)	
Name	Row	Col	NZ	Nod	T	Row	Col	NZ	Nod	T	NZ	Nod	T
3-6-4	10465	2305	43741	1343	58	9325	1165	21889	215	142	21913	1543	116
4-6-4	11617	2305	46045	12849	265	10477	1165	24193	816	214	29977	24217	690
5-6-4	12289	2305	47389	32870	654	11149	1165	25537	1010	237	25561	586	96
3-7-3	7141	1765	25257	234	18	6273	897	14995	590	58	15023	245	29
4-7-3	7897	1765	26769	17220	15	7029	897	16507	523	52	16535	324	32
5-7-3	8359	1765	27693	114	19	7491	897	17431	651	64	17459	858	42
3-7-4	16297	3137	68391	17220	124	14743	1583	33937	480	121	33965	2122	176
4-7-4	18145	3137	72087	7393	574	16591	1583	37633	1609	251	37661	1526	242
5-7-4	19209	3137	74215	60590	2171	17655	1583	39761	113997	11845	39789	1320	1544
3-7-5	31151	4901	152125	59992	3251	28715	2465	64471	6612	76685	64499	627	40145

Table 4: Comparing models (**LW**), (**W**), and (**CW**).

$$
\begin{array}{lll}
(\mathbf{LU}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum\limits_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(iv)} & \sum\limits_{a'<a} x_{a',s,d} + \sum\limits_{d' \leq d} x_{a,s,d'} - r_{a,s,d} \leq 1 & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F} \\
 & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

This model uses binary variables $x_{a,s,d}$, with $a \in \mathcal{A}$, $s \in \mathcal{S}$, $d \in \mathcal{D}$, and $\tau(a) = \tau(d)$, where $x_{a,s,d} = 1$ if and only if arriving vehicle a is assigned to departing trip d and to track s. Equalities (**LU**)(i) and (**LU**)(ii) are matching constraints for arrivals and departures, while Equalities (**LU**)(iii) are knapsack restrictions for tracks. Inequalities (**LU**)(iv) count crossings using binary variables $r_{a,s,d}$.

As we saw in Chapter 2, model (**LU**) has the best performance for the VPP. However, it contains several inequalities of type (**LU**)(iv), which are clearly the hardest ones in this formulation.

In this section, we propose a new approach, where the shunting constraints are eliminated and the crossings are counted directly by a quadratic cost function. This results in the following integer quadratic 3-index formulation for the problem:

$$
\begin{array}{lll}
(\mathbf{U}) & \min \displaystyle\sum_{\substack{s\\(a,d)\dagger(a',d')}} x_{a,s,d}x_{a',s,d'} & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F}.
\end{array}
$$

The matching and knapsack constraints are the same that appear in (**LU**), and because the number of crossings is directly computed in the cost function, variables $r_{a,s,d}$ and Inequalities (**LU**)(iv) can be eliminated.

Model (**U**) has the following property:

Remark 3.2. *Model (**U**) has $O(mn^2)$ variables and $O(n+m)$ constraints.*

Our key observation is that (**U**) can be strengthened by penalizing not only crossings, but also inconsistent assignments. The following model is an extension of (**U**) that takes this aspect into account:

$$
\begin{array}{lll}
(\mathbf{UI}) & \min \displaystyle\sum_{\substack{s\\(a,d)\dagger(a',d')}} x_{a,s,d}x_{a',s,d'} + \sum_{\substack{a\\(s,d)\neq(s',d')}} x_{a,s,d}x_{a,s',d'} & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F}.
\end{array}
$$

The objective function of (**UI**) contains an additional penalty term

$$\sum_{\substack{a\\(s,d)\neq(s',d')}} x_{a,s,d}x_{a,s',d'}$$

for inconsistent assignments of arrivals, i.e., if an arrival is assigned to more than one track and/or more than one trip, the value of the product of the variables representing such an inconsistent assignment is added. The penalty term is zero for every feasible integer solution, but it increases the objective value of the QP relaxation.

The same effect is obtained if we use the following penalty sum:

$$\sum_{\substack{d \\ (a,s)\neq(a',s')}} x_{a,s,d}x_{a',s',d}.$$

However, using both sums does not yield gains from a computational point of view. The following theorem shows how useful these penalty terms can be from a theoretical point of view:

Theorem 3.3. *$V_{QP}(\boldsymbol{UI}) > 0$ if $V(\boldsymbol{UI}) > 0$.*

Proof. If $V(\mathbf{UI}) > 0$, each feasible solution of the problem contains a crossing. Let x^* be an optimal solution of the QP relaxation of ($\mathbf{UI}$). Consider the vector $\lceil x^* \rceil$. If $\lceil x^* \rceil$ contains an integer solution, i.e., a feasible matching, there is a pair of assignments (a,s,d) and (a',s,d') such that $(a,s,d) \dagger (a',s,d')$, $\lceil x^*_{a,s,d} \rceil = 1$, and $\lceil x^*_{a',s,d'} \rceil = 1$. It follows that

$$\sum_{\substack{s \\ (a,d)\dagger(a',d')}} \lceil x^*_{a,s,d} \rceil \lceil x^*_{a',s,d'} \rceil \geq 1,$$

and because the cost function of ($\mathbf{UI}$) is also able to identify this crossing, we have that

$$\sum_{\substack{s \\ (a,d)\dagger(a',d')}} x^*_{a,s,d}x^*_{a',s,d'} > 0.$$

If $\lceil x^* \rceil$ does not contain an integer solution, there is an inconsistent assignment involving some arrival $a \in \mathcal{A}$, i.e., there are two pairs (s,d) and (s',d'), $(s,d) \neq (s',d')$, such that $x^*_{a,s,d} > 0$ and $x^*_{a,s',d'} > 0$. Therefore

$$\sum_{\substack{a \\ (s,d)\neq(s',d')}} x^*_{a,s,d}x^*_{a,s',d'} > 0.$$

It follows that any possible solution x^* is somehow penalized and has cost function greater than zero. As a consequence, we conclude that the QP relaxation of ($\mathbf{UI}$) always yields a non-trivial lower bound if the instance does not admit a crossing-free solution. □

As far as we know, $V_{QP}(\mathbf{UI})$ is the first nontrivial lower bound for the VPP. The linear relaxation of some linear models are also able to produce nonzero lower bounds, but not for all cases. As deciding if an instance of VPP

	(**LU**)					(**U**)				
Name	Row	Col	NZ	Nod	T/s	Row	Col	NZ	Nod	T/s
3-6-4	3511	4609	38017	1	1	61	1159	4621	28	4
4-6-4	3511	4321	30241	1	0	61	871	3463	69	3
5-6-4	3511	4153	25675	59	15	61	703	2797	16	2
3-7-3	3137	4117	30871	12	8	57	1037	4124	16	3
4-7-3	3137	3865	24816	1	1	57	785	3123	20	2
5-7-3	3137	3711	21274	54	6	57	631	2507	27	28
3-7-4	5552	7323	67803	1	1	71	1842	7351	22	10
4-7-4	5552	6861	53509	41	29	71	1380	5503	33	7
5-7-4	5552	6595	45389	1	1	71	1114	4432	21	4
3-7-5	8653	11439	126099	1	4	85	2871	11467	17	34
4-7-5	8653	10725	98582	59	44	85	2157	8604	22	21
5-7-5	8653	10291	82321	26	38	85	1723	6875	31	12
6-7-6	12440	14407	117307	227	200	99	2066	8240	27	32

Table 5: Comparing models (**LU**) and (**U**).

requires shuntings or not is $\mathcal{NP}$-complete, probably there is no linear model with this property.

Table 5 gives the results of a computational comparison between (**U**) and (**LU**) on the same set of test problems as in Section 3.2 plus some additional instances that could not be solved neither by (**CW**) nor by (**W**). Model (**UI**) could not be properly tested due to numerical problem.

The comparison between the results for (**CW**) and (**W**) from Section 3.2 and those for (**LU**) and (**U**) shows a clear superiority of the three-index models over the two-index models. Among the three-index models, (**U**) outperformed (**LU**). An instance 7-8-7, however, could not be solved using any of these formulations even after several hours of computation. We have also tried to apply the convexification technique of Hammer & Rubin (1970) [53] to (**U**), but this time it did not bring any performance gain. A possible explanation for this behavior is that the spectra of the objectives of the (**U**) instances have negative eigenvalues of much larger magnitude than those in the (**W**) instances. We remark that more sophisticated convexification techniques might improve the results (Billionnet & Elloumi (2007) [11]).

Chapter 4

A Set Partitioning Approach

We propose in this chapter a novel set partitioning model and an associated branch-and-price algorithm for the VPP and for the VPP^P. The formulations provide tight linear descriptions of the problems and can produce non-trivial lower bounds. The pricing problem, and hence the LP relaxation itself, can be solved in polynomial, respectively, pseudo-polynomial time if the number of first crossings should be minimized. In order to improve the performance of the algorithm, we introduce heuristics and discuss how to reduce symmetry. With these new formulations, it was possible to obtain good computational results for large-scale instances.

4.1 Introduction

The results presented in the previous chapters show that it is not possible to solve real-world scenarios of the VPP (and, consequently, of the VPP^P as well) just by modeling its instances as mixed integer programs and trying to solve them using standard solvers. One of the challenges lies in the weakness of the theoretical properties of the existing models. Namely, the optimal value of the linear relaxation of most of these formulations is equal to zero and rounding typically does not lead to good integer solutions. Also critical are the consumptions of time and memory, as they make large-scale scenarios intractable for this approach.

We propose in this chapter novel set partitioning models for the VPP and for the VPP^P and a branch-and-price solution approach to solve them. We show that the pricing problems, and hence the entire LP relaxations, can be solved in polynomial, respectively, pseudo-polynomial time if first crossings are to be minimized. Our computational results show that the linear relaxations yield non-trivial lower bounds for some scenarios. Finally, we also introduce heuristics, symmetry breaking techniques, and families of valid inequalities.

This chapter is organized as follows. In Section 4.2 we introduce our set partitioning models and discuss their theoretical properties. Branch-and-price approaches and their algorithmic aspects are investigated in Section 4.3, and computational results are reported in Section 4.4.

A similar technique was used by Diepen, van den Akker & Hoogeveen (2009) [30] in the context of airport gate and bus assignments, where aircrafts have to be assigned to gates and buses have to be assigned to itineraries connecting different flights. Their approach is slightly different, though, as the authors employed a formulation that demands pricing algorithms for two different types of columns (one for each assignment sub-problem), while our method solves the VPP using just one type of variables. A set partitioning approach to the assignment of units to tracks is part of a decomposition method employed by Lentink (2006) [72] to solve the TRAIN UNIT DISPATCHING PROBLEM. Ribeiro & Soumis (1994) [78] propose a column generation approach applied to the MULTIPLE-DEPOT VEHICLE SCHEDULING PROBLEM. Finally, regarding the column generation method, we refer to Lübbecke & Desrosiers (2005) [76] for an introduction, to Lübbecke & Desrosiers (2005) [75] for a general survey, and to Villeneuve et al. (2005) [89] for some applications of this technique to integer programs.

4.1.1 Configurations

In this chapter, the concept of *configuration* plays an important role. For this reason, we recall and expand the description presented in Chapter 2.

A *configuration* q is a sequence of assignments (a_i, s, d_k) to some track s such that the parking positions in s are filled consecutively according to the order of arrival of vehicles. We say that $a \in q$, $d \in q$, $s \in q$, or $(a, d) \in q$ if $(a, s, d) \in q$, and we denote the number of assignments involving track s by $|q|$. Typically, $|q| = \beta$, and if we do not assume that all vehicles have size 1, then we assume that $\sum_{j=1}^{|q|} l(a_j) \leq \beta$ for every configuration q. In resume, a configuration indicates the units assigned to a track. We denote by $q(\mathcal{A})$ $(q(\mathcal{D}))$ the set of arriving vehicles (departures) of configuration q. Given two configurations q and q', we denote by $q \cap q'$ their set of common elements (both arrivals and departures). A pair of configurations q and q' such that $|q \cap q'| > 0$ are said to be *intersecting*. Let $\mathcal{Q}_s$ denote the set of all configurations for track s, i.e., $s \in q$ if and only if $q \in \mathcal{Q}_s$. For the VPP^P, we refer to the set of configurations for period h as Q^h. Let $\mathcal{Q} = \bigcup_s \mathcal{Q}_s$ and $\mathcal{Q} = \bigcup_{h,s} \mathcal{Q}_s^h$ denote the set of all configurations of an instance of VPP and VPP^P, respectively. Finally, we say that a configuration is *uniform* if all its vehicles belong to the same vehicle type.

4.2 Set Partitioning Model

In a solution for the VPP, each arriving vehicle and each departing trip is assigned to exactly one track. We can describe the VPP as a SET PARTITIONING PROBLEM if we interpret tracks as sets and arrivals and departures as the elements that must be partitioned.

We propose the following *set partitioning model* for the VPP:

$$\begin{array}{llll}
(\mathbf{X}) & \min \sum_q c_q x_q & & \\
\text{(i)} & \sum_{a \in q} x_q & = 1 & a \in \mathcal{A} \\
\text{(ii)} & \sum_{d \in q} x_q & = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum_{s \in q} x_q & = 1 & s \in \mathcal{S} \\
 & x_q & \in \{0,1\} & \forall q \in \mathcal{Q}.
\end{array}$$

This model employs binary variables x_q, $q \in \mathcal{Q}$, to indicate the use of track configurations. Equations $(\mathbf{X})$(i) assert that each arriving vehicle a is assigned to exactly one configuration, $(\mathbf{X})$(ii) is analogous for departing trips d, and $(\mathbf{X})$(iii) allows exactly one configuration (which can be empty) for each track. Cost function c assigns penalties to the configurations. For us, they may contain the number of crossings or the number of first crossings.

We are considering here the version of the VPP with identical tracks. We can therefore improve model $(\mathbf{X})$ by working with generic configurations, which do not depend on a particular track. This allows us to replace $(\mathbf{X})$ (iii) by the single inequality

$$\text{(iii}'\text{)} \qquad \sum_q x_q \leq m,$$

which leads to the following equivalent formulation ($\mathbf{X}'$):

$$\begin{array}{llll}
(\mathbf{X}) & \min \sum\limits_{q} c_q x_q & & \\
\text{(i)} & \sum\limits_{a \in q} x_q & = 1 & a \in \mathcal{A} \\
\text{(ii)} & \sum\limits_{d \in q} x_q & = 1 & d \in \mathcal{D} \\
\text{(iii}'\text{)} & \sum\limits_{q} x_q & \leq m & \\
 & x_q & \in \{0,1\} & \forall q \in \mathcal{Q}.
\end{array}$$

With this substitution, we remove a significant amount of symmetry from the model. As we already discussed in Section 2.4.2, symmetry is one of the biggest challenges in the VPP and one of the reasons why the hitherto proposed models do not work well computationally.

Recall also that if we assume that all vehicles have size 1, then at most β of them can be assigned to a single track. In this case, there are (at most) $O(n^{2\beta})$ different sets of arrivals and departures that can be assigned to a track, and from each set it is possible to generate $O(\beta!)$ different arrival and departure sequences. Consequently, there are $O(mn^{2\beta}\beta!) = O(nn^{2\beta}\beta^{\beta}) = O(n^{3\beta+1})$ possible stack configurations.

Proposition 4.1. *If all vehicles have unit size, model ($\boldsymbol{X}$) has $O(n)$ constraints and $O(n^{3\beta+1})$ variables.*

From Lemma 2.24, we know that it suffices to consider the sequential matching of two subsequences $\mathcal{A}'$ and $\mathcal{D}'$, as the other possible assignments produce configurations with a larger number of crossings. If this fact is taken into account, it is possible to reduce model ($\mathbf{X}$) substantially.

Proposition 4.2. *If all vehicles have unit size and all configurations describe sequential matchings, model ($\boldsymbol{X}$) has $O(n)$ constraints and $O(n^{2\beta+1})$ variables.*

From now on, we assume that all the models and formulations contain only configurations describing sequential matchings.

Formulation ($\mathbf{X}$) has appealing theoretical properties. Interpreting the VPP as a partitioning problem for configurations, i.e., as a *combinatorial packing problem*, see Borndörfer (2004) [16], it follows:

Theorem 4.3. *(Borndörfer (2004) [16]) The intersection graph associated with formulation ($\boldsymbol{X}$) for a* VPP *on two tracks is perfect.*

This means that the 2-track case of formulation (**X**) can be solved by a cutting plane algorithm separating only clique constraints.

For the VPP^P, we propose the following similar *set partitioning model*:

$$\begin{array}{llll}
(\mathbf{X}^p) & \min & \sum_q c_q x_q^h & \\
\text{(i)} & & \sum_{a^h \in q} x_q^h = 1 & h, a^h \in \mathcal{A}_h \\
\text{(ii)} & & \sum_{d^h \in q} x_q^h = 1 & h, d^h \in \mathcal{D}_h \\
\text{(iii)} & & \sum_{s^h \in q} x_q^h = 1 & h, s^h \in \mathcal{S}_h \\
 & & x_q^h \in \{0,1\} & h, q \in \mathcal{Q}^h.
\end{array}$$

The main difference between (**X**) and (**X**^{p}) is the addition of index h to the binary variables, indicating the period to which the configuration belongs. Model (**X**^{p}) allows the same use of generic configurations as (**X**). Furthermore, the configuration variables for the weekend parking tracks can be identified. This means that we can replace the inequalities (**X**^{p}) (iii) by the following inequalities

$$\begin{array}{lll}
(\text{iii}^a) & \sum_q x_q^h \leq m & \text{weekday period } h; \\
(\text{iii}^b) & \sum_q x_q^h \leq b & \text{weekend period } h,
\end{array}$$

in order to obtain the following equivalent formulation ($\mathbf{X}^{p\prime}$):

$$\begin{array}{llll}
(\mathbf{X}^p) & \min & \sum_q c_q x_q^h & \\
\text{(i)} & & \sum_{a^h \in q} x_q^h = 1 & h, a^h \in \mathcal{A}_h \\
\text{(ii)} & & \sum_{d^h \in q} x_q^h = 1 & h, d^h \in \mathcal{D}_h \\
\text{(iii)}^{\text{a}} & & \sum_q x_q^h \leq m & \text{weekday period } h \\
\text{(iii)}^{\text{b}} & & \sum_q x_q^h \leq b & \text{weekend period } h \\
 & & x_q^h \in \{0,1\} & h, q \in \mathcal{Q}^h.
\end{array}$$

Again, this technique removes symmetry from the model, improving the computational performance.

The number of valid configurations can be calculated independently for each period, but vehicles that were parked after the last weekday period come back into use on the first weekday period. We therefore have $O(n^{2\beta})$ possible configurations on all periods except the first weekday period and the first weekend period, which should be considered simultaneously and have $O((2n)^{2\beta})$ possibilities.

Proposition 4.4. *If all vehicles have unit size, model ($\boldsymbol{X}^p$) has $O((p+w)n)$ constraints and $O((p+w-2)n^{2\beta}+(2n)^{2\beta})$ variables.*

We now compare the strengths of the linear relaxations of models (**LU**) and (**X**). Formulation (**LU**), presented in Chapter 2.2.1.2, is the most successful model for the VPP described in the literature so far. From Theorem 2.6, we know that $V_{LP}(\mathbf{LU}) = 0$ if $m > 1$. The following two theorems show that (**X**) is a tighter formulation for VPP than (**L**).

Theorem 4.5. *For every element of $P_{LP}(\boldsymbol{X})$, there is one element with the same objective value in $P_{LP}(\boldsymbol{LU})$.*

Proof. Let $(x_q)_{q\in\mathcal{Q}}$ be an element of $P_{LP}(\mathbf{X})$. We set the values of $(x_{a,s,d}, r_{a,s,d})$ for each $(a,s,d) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}$ as follows:

$$x_{a,s,d} = \sum_{(a,s,d)\in q} x^*_q;$$

$$r_{a,s,d} = \sum_{\substack{(a,d'),(a',d)\in q\\ (a,d')\dagger(a',d)\\ s\in q}} x^*_q.$$

We show that $(x_{a,s,d}, r_{a,s,d})$ belongs to $P_{LP}(\mathbf{LU})$:

$$\text{(i)} \quad \sum_{(s,d)} x_{a,s,d} = \sum_{(s,d)} \sum_{(a,s,d)\in q} x_q^* = \sum_{a\in q} x_q^* = 1 \qquad a \in \mathcal{A}$$

$$\text{(ii)} \quad \sum_{(a,s)} x_{a,s,d} = \sum_{(a,s)} \sum_{(a,s,d)\in q} x_q^* = \sum_{d\in q} x_q^* = 1 \qquad d \in \mathcal{D}$$

$$\text{(iii)} \quad \sum_{(a,d)} x_{a,s,d} = \sum_{s\in q} x_q^* |q| \leq \sum_{s\in q} x_q^* \beta \leq \beta \qquad s \in \mathcal{S}$$

$$\begin{aligned}
\text{(iv)} \quad & \sum_{a'<a} x_{a',s,d} + \sum_{d'\leq d} x_{a,s,d'} - 1 \leq \\
& \sum_{\substack{(a',d),(a,d')\in q \\ a'<a, d'\leq d \\ s\in q}} 2x_q^* + \sum_{\substack{(a',d)\vee(a,d')\in q \\ a'<a, d'\leq d \\ s\in q}} x_q^* - 1 = \\
& 2 \sum_{\substack{(a',d),(a,d')\in q \\ a'<a, d'\leq d \\ s\in q}} x_q^* - \sum_{\substack{(a',d),(a,d')\in q \\ a'<a, d'\leq d \\ s\in q}} x_q^* = \\
& \sum_{\substack{(a',d),(a,d')\in q \\ a'<a, d'\leq d \\ s\in q}} x_q^* = r_{a,s,d} \qquad (a,s,d) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}
\end{aligned}$$

The inequalities above show that the pair $(x_{a,s,d}, r_{a,s,d})$ is an element of $P_{LP}(\mathbf{L})$, and as

$$\sum_{a,d} r_{a,s,d} = \sum_{\substack{(a,d'),(a',d)\in q \\ (a,d')\dagger(a',d) \\ q\in\mathcal{Q}_s}} x_q = \sum_{q\in\mathcal{Q}_s} c_q x_q,$$

it follows that $(x_q)_{q\in Q}$ and $(x_{a,s,d}, r_{a,s,d})$ have the same cost value. □

Now we show that $(\mathbf{X})$ provides nontrivial lower bounds for some instances of VPP.

Theorem 4.6. $V_{LP}(\boldsymbol{X}\,) > 0$ *for some instances of* VPP *that require shuntings.*

Proof. We construct a family of instances of VPP containing a unit which belongs to every matching and which can only belong to configurations q such that $c_q > 0$. Let $\mathcal{A}$ and $\mathcal{D}$ be such that $\tau(a_1) = \tau(d_n) = \mathcal{T}_k$, with $t_k = 1$, and such that $l(a) = 1$ for every $a \in \mathcal{A}$. Assume that the depot has n parking positions, i.e., feasible solutions contain only configurations with β units.

In every instance of this family, a_1 must be assigned to d_n and $c_q > 0$ for every configuration q such that $(a_1, d_n) \in q$. As $\sum_{a_1 \in q} x_q = \sum_{d_n \in q} x_q = 1$, it is clear that for every feasible solution of the linear relaxation of $(\mathbf{X})$, at least one of the configurations q containing a_1 (and, consequently, d_n as well) will be present, i.e., $x_q > 0$. Consequently, the objective value of this solution is greater or equal than $c_q x_q > 0$, yielding a non-trivial lower bound for the original mixed integer program. $\square$

From Theorems 4.5 and 4.6, we conclude that $(\mathbf{X})$ is stronger than $(\mathbf{LU})$ from a theoretical point of view. A similar proof yields the following corollary regarding VPP^P:

Corollary 4.7. *$V_{LP}(\boldsymbol{X}^p\ ) > 0$ for some instances of* VPP^P *that require shuntings.*

Theorem 4.6 and Corollary 4.7 show that the linear relaxations of $\mathbf{X}$ and $\mathbf{X}^p$ can produce non-trivial lower bounds for some instances of both problems. The design of formulations whose linear relaxation are always able to yield non-trivial lower bounds does not seem to be a reasonable goal. With such models, one would be able to decide the necessity of shuntings by solving a linear program, and this would prove $\mathcal{P} = \mathcal{NP}$. Both zero and non-zero lower bounds will come up in our computations.

4.3 Branch-and-Price Framework

We propose to solve $(\mathbf{X})$ and $(\mathbf{X}^p)$ using a branch-and-price algorithm. For the pricing subproblem, we developed a heuristic and an efficient exact algorithm for the minimization of first crossings. In order to speed up the computational performance of the framework, we investigated: heuristics to generate initial solutions and to compute matchings containing some fixed columns (typically the ones generated by the pricing algorithms); pruning strategies to remove redundant nodes from the solution tree; and families of inequalities to increase the quality of the linear relaxations.

4.3.1 Pricing Algorithms

A pricing algorithm is employed to check if the optimal solution obtained with the current pool of columns is optimal for the original relaxed program. If this is not the case, the algorithm identifies columns that can be incorporated into the model in order to improve the solution. In our case, a configuration whose associated reduced cost is negative is the index of such a variable.

The idea is to use a dynamic program that records in each state the last arriving vehicle and the last departing trip assigned to the current configuration. One problem with this approach is that book-keeping becomes necessary if crossings should be minimized. Namely, when we are deciding if we want to add (a, d) to the current configuration, we want to know exactly how many units in $\mathcal{A} \times \mathcal{D}$ that were assigned previously to the configuration are involved in a crossing with (a, d) (just knowing about the relation between (a, d) and the last added element is not enough). Moreover, this technique is not able to verify if a given arrival or departure appeared already, and therefore it may produce invalid configurations.

These problems can be addressed as follows. First, Winter (1998) [91] has shown that considering only configurations where the arriving vehicles do not produce crossings, i.e., with ascending arrival times, does not change the minimal number of crossings. The following proposition shows that this result can be adapted to scenarios where first crossings have to be minimized, and for this reason we will assume that sets $\mathcal{Q}$ and $\mathcal{Q}^p$ contain only such configurations:

Proposition 4.8. *For every instance of the* VPP*, there is a solution that minimizes the number of first crossings where each configuration has its units ordered according to the arrival times.*

Proof. We prove this fact using the same strategy employed by Winter (1998) [91]. This claim clearly holds for crossing-free configurations, so we will concentrate on configurations containing at least one first crossing.

Let q be a configuration such that the units are not ordered in the parking positions according to the arrival times. We claim that the configuration q' composed by the same units as q whose distribution in the track follows the order of $q'(\mathcal{A})$ is such that $c_q \geq c_{q'}$, i.e., q' will not contain more first crossings than q.

Suppose that this assertion does not hold. Let (a_i, d_l) and (a_j, d_k) be a pair

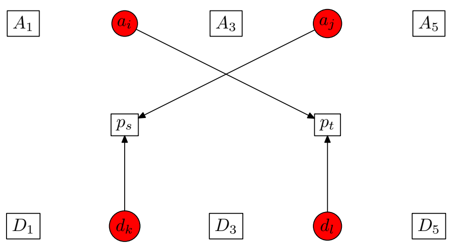


Figure 1: Configuration with units not ordered by arrival time

of units assigned to positions s and t in q, respectively, with $t = s+1$. This scenario is presented in Figure 1.

Assume that unit (a, d) is parked in position $t + 1$. If we reorganize the parking positions s, t and $t+1$ in such a way that the units (a_i, d_l), (a_j, d_k), and (a, d) are ordered according to their time of arrival, direct verification shows that if $(a, d) \in \{A_5 \times D_3\} \cup \{A_1 \times D_3\} \cup \{A_1 \times D_1\}$, then the number of first crossings involving these vehicles will decrease, and if not, they will remain the same.

Similarly, if (a, d) is assigned to parking position $s-1$ and we make the same reorganization, if $(a, d) \in \{A_3 \times D_3\} \cup \{A_3 \times D_5\} \cup \{A_5 \times D_3\} \cup \{A_5 \times D_5\}$, the number of first crossings will decrease, and if not, they will remain the same.

Therefore, we conclude that if we reorder the units of q according to the times of arrival of its vehicles, we will obtain a configuration q' such that $c_{q'} \leq c_q$. $\square$

Another relevant fact is that allowing several assignments to cover a departure is not problematic. It slows down the convergence of the algorithm, but, as we will see, it makes the pricing problem easier. We therefore consider relaxations $(\mathbf{X}'')$ and $(\mathbf{X}^{p\prime\prime})$ of the formulations $(\mathbf{X}')$ and $(\mathbf{X}^{p\prime})$ which extend $\mathcal{Q}$ and $\mathcal{Q}^p$ to sets $\mathcal{Q}''$ and $\mathcal{Q}^{p\prime\prime}$ that include such configurations, and which relax constraints $(\mathbf{X}')$ (ii) and $(\mathbf{X}^{p\prime})$ (ii) to

$$\text{(ii}'\text{)} \qquad \sum_{d \in q} x_q \geq 1 \quad d \in \mathcal{D}; \text{ and}$$

$$\text{(ii}'\text{)} \qquad \sum_{d \in q''} x_h^p \geq 1 \quad h, d \in \mathcal{D},$$

respectively. This construction has a similar gist as the so-called q-path relaxation, which is popular in vehicle routing (see Baldacci, Toth & Vigo (2007) [6]).

Proposition 4.9. $0 \leq V_{LP}(\boldsymbol{X}'') \leq V_{LP}(\boldsymbol{X})$ *and* $0 \leq V_{LP}(\boldsymbol{X}^{p\prime\prime}) \leq V_{LP}(\boldsymbol{X}^p)$.

Theorem 4.10. *The pricing problem for the LP relaxation of model ($\boldsymbol{X}$) can be solved in pseudo-polynomial time $O(mn^4\beta)$ if first crossings are minimized.*

Proof. The pricing problem can be solved by dynamic programming. For a fixed track s, consider a state space $\mathcal{H}$ indexed by tuples in $\mathcal{A} \times \mathcal{D} \times [0, \beta]$. Let C be a matrix indexed by elements of $\mathcal{H}$ such that entry $c[a][d][k]$ holds the minimum reduced cost of a configuration of size k whose last assigned unit is (a, d). Let us denote the dual variables associated with constraints ($\mathbf{X}''$) (i), (ii′), and (iii′) by σ_a, ω_d, and π_s, respectively. Then the recurrence for the entries of C is as follows:

$$c[a'][d'][k'] = \mathtt{min}_{a<a',d\neq d',k=k'-l(a)} \left\{c[a][d][k] - \sigma_{a'} - \omega_{d'} + \alpha_{d,d'}\right\},$$

where $\alpha_{d,d'} = 1$ if $d > d'$, and 0 else. Finally, if $k = 0$, then $c[a][d][k] := -(\sigma_a + \omega_d + \pi_s)$. Matrix C can be determined in time $O(n^4\beta)$, and if we compute it for each track s, the pricing algorithm consumes time $O(mn^4\beta)$. □

Corollary 4.11. *The pricing problem for the LP relaxation of $\boldsymbol{X}^{p\prime\prime}$ can be solved in pseudo-polynomial time $O(Pmn^4\beta)$ if first crossings are minimized.*

Proof. The pricing problem for the VPP^P is similar to the pricing problem for the VPP. Basically, the dynamic program in the proof of Theorem 4.10 must be applied to each period, but in addition to that, we also have to consider configurations containing arrivals of the first weekend period and departures of the first weekday period. Consequently, we execute the pricing routine $P+1$ times in each iteration, which leads to an $O(Pmn^4\beta)$ pseudo-polynomial time algorithm. □

Theorem 4.12. *The pricing problem for the LP relaxation of ($\boldsymbol{X}$) can be solved in pseudo-polynomial time $O(mn^2\beta)$ if first crossings are minimized and the assignments $\mathcal{A} \times \mathcal{D}$ are fixed.*

Proof. The pricing routine for this special case can also be solved by dynamic programming. Let $\mathcal{U} \subset \mathcal{A} \times \mathcal{D}$ be the set of fixed units, and denote by $u(a)$ and $u(d)$ the arrival and the departure composing unit u, respectively. For a fixed track s, consider a state space $\mathcal{H}$ indexed by tuples in $\mathcal{U} \times [0, \beta]$. Let C be a matrix indexed by H such that entry $c[u][k]$ holds the minimum reduced cost of a configuration of size k whose last assigned unit is u. Let us denote the dual variables associated with constraints ($\mathbf{X}''$) (i), (ii′), and (iii′) by σ_a, ω_d, and π_s, respectively. Then the recurrence for the entries of C is as follows:

$$c[u'][k'] = \mathtt{min}_{\substack{u(a)<u'(a)\\ k=k'-l(u'(a))}} \left\{c[u][k] - \sigma_{u'(a)} - \omega_{u'(d)} + \alpha_{u,u'}\right\},$$

where $\alpha_{u,u'} = 1$ if $u(d) > u'(d)$, and 0 else. The initialization is $c[u][0] := -(\sigma_{u'(a)} + \omega_{u'(d)} + \pi_s)$. Matrix C can be determined in time $O(n^2\beta)$, and if we compute it for each track s, the pricing algorithm consumes time $O(mn^2\beta)$. □

Corollary 4.13. *The pricing problem for the LP relaxation of $\boldsymbol{X}^{p\prime\prime}$ can be solved in pseudo-polynomial time $O(Pmn^2\beta)$ if first crossings are minimized and the assignments $\mathcal{A} \times \mathcal{D}$ are fixed.*

Proof. Using the same proof strategy employed in Corollary 4.11, this results follows directly from Theorem 4.12. □

Recall that if vehicles have unit size, then $n = m\beta$. Therefore:

Corollary 4.14. *The pricing problem for the LP relaxation of ($\boldsymbol{X}''$) can be solved in polynomial time $O(n^5)$ if first crossings are minimized and vehicles have unit size.*

Corollary 4.15. *The pricing problem for the LP relaxation of ($\boldsymbol{X}^{p\prime\prime}$) can be solved in polynomial time $O(Pn^5)$ if first crossings are minimized and vehicles have unit size.*

Corollary 4.16. *The pricing problem for the LP relaxation of ($\boldsymbol{X}''$) can be solved in polynomial time $O(n^3)$ if first crossings are minimized, vehicles have unit size, and the assignments $\mathcal{A} \times \mathcal{D}$ are fixed.*

Corollary 4.17. *The pricing problem for the LP relaxation of ($\boldsymbol{X}^{p\prime\prime}$) can be solved in polynomial time $O(Pn^3)$ if first crossings are minimized, vehicles have unit size, and the assignments $\mathcal{A} \times \mathcal{D}$ are fixed.*

4.3.2 Heuristics for the Generation of Columns

In a column generation approach, it is important to initialize the column pool with a promising set of configurations. Moreover, in order to speed up the convergence of the algorithm, usually it is interesting to incorporate into the model, in each iteration, a set of columns that compose a complete solution together with the variable(s) with negative reduced cost selected by the pricing routine.

We investigate now algorithms that can be used to generate valid matchings for the VPP. The procedures described here can be directly employed in the root node of the branch-and-price tree, but we remark that they can be easily adapted to scenarios where there are fixed assignments and/or fixed configurations.

4.3.2.1 Uniform-Tracks Heuristic

We introduce a greedy procedure based on the concept of *uniform tracks* in order to construct a set of configurations with a potentially small number of crossings.

The following proposition presents a lower bound for the number of *non-uniform* tracks in a feasible solution for an instance of the VPP (or for one period in an instance of the VPP^P).

Proposition 4.18. *If all tracks have the same size, there must be at least*

$$B_{NU} := \frac{\sum_{i=1}^{t} t_i \mod \beta}{\beta}$$

non-uniform tracks in any solution of the VPP*.*

Proof. In any feasible solution of the VPP containing as many uniform tracks as possible, at least $t_i \mod \beta$ units of type $\mathcal{T}_i$ will be assigned to non-uniform tracks. Therefore, any feasible solution contains at least B_{NU} non-uniform tracks. □

Based on this lower bound, we developed an algorithm that assigns $t_i \mod \beta$ arrivals and departures of type $\mathcal{T}_i$, $1 \leq i \leq t$, to B_{NU} tracks. If the remaining arrivals and departures are arranged in order to compose $m - B_{NU}$ uniform tracks, we obtain a feasible matching for one period. As the following theorem shows, this procedure does not always produce optimal solutions.

Theorem 4.19. *There are instances of the* VPP *for which there is no optimal solution that uses only B_{NU} non-uniform tracks.*

Proof. We construct a family of instances of the VPP for which any optimal solution contains $B_{NU} + 1$ non-uniform tracks. Let $t = 2$, with $t_1 = \beta - 1$ and $t_2 = \beta + 1$, $n = 6 + 2k$, $k \in \mathbb{N}$. It is clear that $B_{NU} = 1$. Assume that a_n, d_{n-2}, d_{n-3}, a_{n-5}, arrivals a_{2i} such that $1 \leq 2i \leq n - 6$, $i \in \mathbb{N}$, and

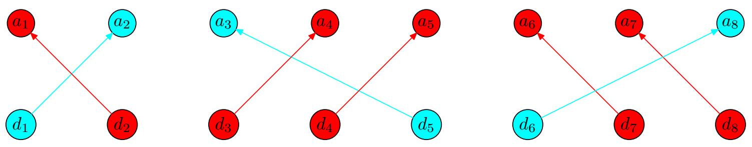


Figure 2: Example of instance which requires non-uniform tracks for crossing-free solutions.

departures d_{2i+1} such that $1 \leq 2i+1 \leq n-6$ are of type $\mathcal{T}_1$, while the remaining elements are of type $\mathcal{T}_2$. Figure 2 shows an example for $\beta = 4$, where the blue (light-colored) elements are of type $\mathcal{T}_1$ and the red (dark-colored) elements are of type $\mathcal{T}_2$.

Every unit of type $\mathcal{T}_2$ is involved in at least one crossing with a unit of type $\mathcal{T}_1$. Therefore, any solution containing just one non-uniform track will contain at least one crossing. Conversely, if we assign arrivals a_{2i}, $1 \leq 2i \leq n-6$, a_{n-5}, a_{n-4}, and a_{n-3} and departures d_{2i+1}, $1 \leq 2i+1 \leq n-6$, d_{n-5}, d_{n-4}, and d_{n-3} to one track and the remaining to the others and if we use the sequential assignment for both configurations, we will obtain two non-uniform tracks composed of units of type $\mathcal{T}_1$ and type $\mathcal{T}_2$ which are crossing-free for each instance of this family. $\square$

We build the configurations of the non-uniform tracks in two phases. In the first phase, we try to insert the $t_i \mod \beta$ remaining units of type $\mathcal{T}_i$, $1 \leq i \leq t$, in B_{NU} crossing-free tracks. We do it greedily and in a sequential way. More specifically, we check for each arrival a (following the order defined in $\mathcal{A}$) if we still have to insert elements of type $\tau(a)$ to the non-uniform tracks and if there is any unassigned departing trip d in $\mathcal{D}$ such that unit (a,d) can be assigned to one of the current configurations without producing a first crossing. If this is the case, we add the unit to a configuration. If there is more than one trip satisfying these conditions, we pick the one which departs first.Finally, if a can (or should) not be assigned, we proceed to the next arrival, leaving it unassigned.

If all B_{NU} configurations have β units in the end of the first phase, we are done. If this is not the case, we complete the configurations in a second phase. In this step, the assignment of arrivals and departures follow the sequential matching and first crossings are allowed.

In both phases, when there are several track candidates to which we can assign a given unit, we choose the one which has the smallest number of assigned elements. If there are still several options, we choose one randomly among the suitable candidates.

Finally, for each unit u, it is possible to determine the maximum number of vehicles that will arrive after u and that can compose with it a crossing-free configuration. This idea has further applications in the framework and will be discussed in details in Section 4.3.3.2, but we remark that the results of the heuristic can be improved if we also perform this check in the first phase.

4.3.2.2 Simulated Annealing

Based on the algorithm proposed by Czech (1999) [25] for the Set Partitioning Problem, we developed the subroutine described in Algorithm 4. The method keeps a current solution x, and in each iteration, two units assigned to different tracks exchange positions, yielding a new matching y. Let Δ denote the difference between the number of first crossings of x and the number of first crossings of y. If $\Delta < 0$, y becomes the current solution. If not, we generate a random number in the interval $(0, 1)$ and verify if it is smaller than $T/(T + \frac{n\Delta}{2})$, where T is a global parameter which is reduced in each outer iteration. If this is the case, y becomes the current solution. The parameter T is decreased in each iteration, which makes transitions to matchings with more crossings less likely to occur in later iterations.

Algorithm 4 Simulated Annealing for the VPP

```
Fix all the assignments $\mathcal{A} \to \mathcal{D}$
Create an initial solution $x$
for $i := 0$ to $outSteps$ do
  $T := T * \alpha$
  for $j := 0$ to $inSteps$ do
    Create solution $y$ after switching the positions of 2 random units
    $\Delta = c(x) - c(y)$
    if $\Delta < 0$ or $random(0,1) < T/(T + \frac{n\Delta}{2})$ then
      $x = y$
    end if
  end for
end for
```

Local search heuristic methods are useful when it is possible to improve a solution with a few local modifications. Unfortunately, this is not always the case for the VPP.

Proposition 4.20. *There are instances of the* VPP *which admit pairs of solutions x and y such that x contains one crossing, y is crossing-free, and x and y do not have any configuration in common.*

Proof. Consider the family $\mathcal{I}$ of instances of the VPP such that $\beta = m$; the arrival sequence is $\mathcal{A} = \{1, 2, 3, ..., n\}$, where each number represents a vehicle type (therefore, $\tau(a_i) \neq \tau(a_k)$ for $1 \leq i \neq k \leq n$); and the departing sequence $\mathcal{D}$ is as follows:

$$\begin{aligned}\mathcal{D} = \{ \quad & n-m+1, n-m+2, \dots, n-3, n-2, n, n-1, \\ & n-2m+1, n-2m+2, \dots, n-m, \\ & \vdots \\ & 1, 2, \dots, m\};\end{aligned}$$

that is, if we partition $\mathcal{A}$ into m subsequences of size $\beta = m$, invert the order of these subsequences, and exchange the departing order of trips of type n and $n-1$, we obtain $\mathcal{D}$.

If each arrival a_i is assigned to the track $s_{\lceil (i)/m \rceil}$, the resulting solution x contains one crossing (which involves the units of type $n-1$ and n).

If each arrival a_i is assigned to the track $s_{((i-1) \mod m)+1}$, we will obtain a crossing-free solution y. Finally, if follows from the construction that x and y do not have configurations in common. □

The previous proposition shows that an almost optimal solution may not be transformed into an optimal one with just a few modifications of the matching. This fact suggests that local search methods may not be a good strategy for the VPP. In addition to that, it is also clear that approaches based on random sampling are not likely to produce good matchings if not executed for long periods of time.

4.3.3 Exact Solution

The column generation approach is a complete solution method for linear programs. Namely, columns are added to the restricted master problem in each iteration, and the method stops if no variable with negative (or positive if we have a maximization problem) reduced cost is identified by an exact pricing subroutine, which proves the optimality of the current best solution. However, if we have a mixed integer linear program, it is not possible to know if the set of variables generated during the solution of the relaxed master problem also contains an optimal integer solution for the original master problem. Therefore, we can only have an exact solution method for the VPP based on the formulation $\mathbf{X}$ if a branch-and-price strategy is implemented.

4.3.3.1 Branching Criteria

In order to construct a solution tree for the branch-and-price algorithm, we use a criterion based on the method proposed by Ryan & Foster (1981) [79] for the Set Partitioning Problem and on the strategies suggested by Savelsbergh (1997) [81] for the Generalized Assignment Problem. Basically, for every node belonging to the i-th level of the solution tree, each vehicle a_j, $1 \leq j \leq i$, has its assignments to a trip $fixD(a_j)$ and to a track $fixS(a_j)$ fixed. The level of the root node is zero, and the level of the remaining nodes is equal to the level of its parent plus one.

If all the tracks are similar, their labels can be ignored. However, in order to assert that an integer solution will eventually be produced, it is also necessary to indicate in each node which units can be parked in the same track and which can not. Therefore, assignments $\mathcal{A} \rightarrow \mathcal{S}$ also have to be fixed in this case. Consequently, for each node u belonging to the the i-th level, a configuration containing vehicles a_j and a_k, $1 \leq j, k \leq i$, will belong to u's LP only if $fixS(a_j) = fixS(a_k)$. We call this strategy *sequential fixing criterion*.

An alternative criterion consists of choosing any non-fixed element in $\mathcal{A}$ to have its assignments determined in the i-th step, and not necessarily the i-th vehicle. This scheme gives more freedom of choice, but it makes the pricing problem much harder to solve. Namely, if we know that vehicles a_i and a_j, $a_i < a_j$, are assigned to the same track s, it may be impossible to know in advance how many vehicles assigned to s will be positioned before a_i, after a_j, or after a_i and before a_j. Therefore, an exact pricing subroutine would have to test all the possibilities. Conversely, if the sequential fixing criterion is used, then we know that the k units assigned to track s will certainly occupy its first k parking positions, and this situation can be easily handled after simple modifications of the pricing routines presented in Section 4.3.1. The same holds for the heuristics presented in Section 4.3.2.

There are other criterion to fix the variables which can also be easily incorporated by the pricing algorithms. For example, we could initially fix all the assignments $\mathcal{A} \rightarrow \mathcal{D}$, and only after this we would start to fix the assignments $\mathcal{A} \rightarrow \mathcal{S}$. It is not clear from a theoretical point of view which strategy is better, but preliminary computational tests showed that the sequential fixing criterion has a more satisfactory performance.

Solution Tree The solution tree of our framework was implemented as a red-black tree. This data structure was chosen because it is a balanced binary tree and its insertion and deletion operations consume time $O(lgN)$, where N is the number of nodes which currently belong to the tree. For further details, we refer to Cormen et al. (2001) [23].

The key of the nodes (i.e., the number assigned to them used in the organization of the tree) contains the lower bound obtained from the parents. If there is a draw, we choose the node with the highest level. Nodes whose fixed assignments contain longer uniform sequences, i.e., sequences of units of the same type assigned to consecutive positions in a track, and nodes with a larger number of occupied tracks also have preference. After solving a node from the i-th level, we generate its "offspring", i.e., one node for each possible assignment involving the $(i+1)$-th arrival, and add them to the tree.

Integer solutions are obtained after the execution of the heuristics and after the solution of some LPs. The investigation of the nodes finishes when the tree becomes empty or when an integer solution whose cost equals the current global lower bound is found.

4.3.3.2 Pruning Strategies

The complete solution tree of our branch-and-price framework has exponential size. Consequently, the method can only be executed until its completion under reasonable time and memory constraints if we identify some pruning strategies that will help to eliminate nodes from the tree.

Proposition 4.21. *For every node v of the tree, if U is the number of crossings of the best integer solution found so far, then we do not need to investigate the descendents of v if the optimal value lb of its linear relaxation is such that $\lceil lb \rceil \geq U$.*

Proof. The optimal value lb computed for the relaxation of the current node v yields a lower bound for each of its descendents. Moreover, it is clear that every feasible solution of $\mathbf{X}$ is integer. Therefore, if lb is not integer, then any feasible integer solution for the problem that takes the fixed assignments described in v into account will contain at least $\lceil lb \rceil$ first crossings. Consequently, if $U \geq \lceil lb \rceil$, the investigation of the descendents of v is irrelevant and should not be conducted. □

For each unit (a, d), let $M(a, d)$ be any set $\{(a', d') : a' > a\} \cup \{(a, d)\}$ such that $(a_i, d_j), (a_k, d_l) \in M(a, d)$ only if (a_i, d_j) and (a_k, d_l) are noncrossing. The cardinality $m(a, d)$ of the largest set $M(a, d)$ for all the units (a, d) can be computed in $O(n^4)$ by Algorithm 5. Function $ok : \mathcal{A} \times \mathcal{D} \to \{true, false\}$ indicates if arrival a can be assigned to departure d and function $ok_2 : \mathcal{A} \times \mathcal{A} \to \{true, false\}$ indicates if arrivals a_i and a_j can be assigned to the same track. We present the algorithm for the case where all tracks are equal, but it is easy to modify it in order to compute m for each track individually.

Algorithm 5 Computation of $m(a, d)$

```
for $a_1 := 1$ to $n$ do
  for $d_1 := 1$ to $n$ do
    if $ok(a_1, d_1)$ = false then
      continue
    end if
    for $a_2 := a_1 - 1$ down to 1 do
      for $d_2 := d_1 - 1$ down to 1 do
        if $ok(a_2, d_2)$ = false or $ok_2(a_1, a_2)$ = false then
          continue
        end if
        if $m(a_2, d_2) < m(a_1, d_1) + 1$ then
          $m(a_2, d_2) = m(a_1, d_1) + 1$
        end if
      end for
    end for
  end for
end for
```

Let c be the number of crossings involving the fixed elements of the current node. If a_i will be assigned to d_j and $m(a_i, d_j) = k$, where m is computed taking into account the fixed assignments for the node, then any solution containing (a_i, d_j) assigned to some track s with less than $\beta - k$ elements will clearly contain at least $c + 1$ crossings. As a consequence, if $c + 1 \geq U$, we can avoid the insertion of nodes containing these fixed assignments to the solution tree.

Some of the symmetry breaking techniques presented in Section 2.4.2 can be used in the present context as pruning strategies. For example, it is possible to avoid the insertion of a node in the tree if there is another one which was already added and which is equal if its tracks are properly relabeled. Based on this, if the assignments of the i-th vehicle are going to be fixed and there are several empty tracks, we will only generate the node containing the assignment of a_i to the first empty track and the nodes describing the assignment of a_i to the non-empty tracks.

With this strategy, it is clear that, for each node, if a_i and a_j are the first vehicles assigned to the tracks s_k and s_l, respectively, and if $a_i < a_j$, then $k < l$. Any node containing a_i as the first arrival assigned to s_l and a_j as the first arrival assigned to s_k will not be added to the solution tree.

Therefore, in a situation where all the tracks are equal, if $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ have been assigned to the first position of their respective tracks, only one out of the $k!$ $(\leq m!)$ possible distributions of the (incomplete) configurations will appear in the tree.

It is possible to use the same idea in a more general scenario. Let q_i and q_j be two partial configurations (i.e., configurations to which may add other units without violating the size restrictions of the tracks) containing k elements each $(1 \leq k \leq \beta)$. Let (a_w, d_x) and (a_y, d_z) be the k-th elements assigned to q_i and q_j, respectively. Let $q_{i'}$ $(q_{j'})$ be the configuration originated from q_i (q_j) after the removal of (a_w, d_x) $((a_y, d_z))$ and insertion of (a_y, d_z) $((a_w, d_x))$. If the k-th arrivals of $q_{i'}$ and $q'_{j'}$ are a_y and a_w, respectively, and if the sum of the number of first crossings of $q_{i'}$ and $q_{j'}$ is equal to the sum of the number of first crossings of q_i and q_j, then we say that q_i and q_j are *interchangeable configurations*.

Interchangeable configurations are a source of symmetry, but this situation can be avoided in the same way we avoid track symmetry. Namely, in the y-th level of the tree (when we define the assignments of arrival a_y), before creating a node where the assignment (a_y, s_i, d_z) is fixed and (a_y, d_z) is the k-th unit of the track, we can verify if the resulting partial configuration is interchangeable with the partial configuration of some track s_j which also contains k elements and such that $j > i$. If this is the case, we do not have to add this node, as the last unit assigned to s_j (say, (a_w, d_x)) was also assigned to track s_i in a previous iteration and a descendent z of this node will have exactly the same situation described in the node that we avoided, i.e., z contains the same set of partial configurations, but with unit (a_y, d_z) assigned to the k-th position of track s_j and with unit (a_w, d_x) assigned to the k-th position of track s_i.

4.3.4 Cuts

We investigate now families of clique inequalities for $(\mathbf{X})$ and $(\mathbf{X}^p)$ based on the work of Balas & Saltzman (1989) [5] on the Three-index Assignment Problem.

4.3.4.1 Intersection Graph.

In order to describe families of clique inequalities in this section, we will use the *intersection graph* $G^{\mathcal{Q}}$ of sets $\mathcal{Q}$, defined as follows. Each vertex of $G^{\mathcal{Q}}$ represents a configuration in $\mathcal{Q}$, so here we will overcharge the notation and use q in order to refer both to a configuration in $\mathcal{Q}$ and to its representing vertex in $G^{\mathcal{Q}}$. There is an edge connecting a pair of vertices q, q' of $G^{\mathcal{Q}}$ if and only if $|q \cap q'| > 0$, i.e., if the configurations have at least one element (an arrival or a departure) in common.

From $(\mathbf{X})$(i) and $(\mathbf{X})$(ii), it follows that it is not possible to have in the same matching two or more configurations belonging to a clique in $G^{\mathcal{Q}}$. As a consequence, it is possible to define the following valid clique inequality for $(\mathbf{X})$ for each clique C in $G^{\mathcal{Q}}$:

$$\sum_{q \in C} x_q \leq 1.$$

Instances With Two Tracks. If $m = 2$ and if $(\mathbf{X})$ contains only configurations describing sequential matchings, each configuration q can only compose a matching with the (unique) configuration q' such that $|q' \cap q| = 0$. Therefore,

$$\sum_{|q' \cap q| = 0} x_{q'} = 2(1 - x_q) \qquad q \in \mathcal{Q}.$$

and

$$x_q = x_q' \qquad q, q' \in \mathcal{Q}, |q \cap q'| = 0.$$

are valid inequalities which clearly imply any other family of equalities and/or inequalities for this special case of the problem.

Types with Two Elements For every type $\mathcal{T}_i$ such that $t_i = 2$, there are only two possible and self-excluding assignments involving its elements.

For example, if $\mathcal{A}^i = \{a_j, a_k\}$ and $\mathcal{D}^i = \{d_l, d_m\}$, a_j is assigned to d_l (d_m) if and only if a_k is assigned to d_m (d_l). Therefore, equalities

$$\begin{aligned}\sum_{(a_j,d_l)} x_q &= \sum_{(a_k,d_m)} x_q; \text{ and}\\ \sum_{(a_j,d_m)} x_q &= \sum_{(a_k,d_l)} x_q\end{aligned}$$

hold for these types of vehicle.

4.3.4.2 Fixed Sets of Arrivals and Departures

Initially, we consider the families of cliques that can be obtained if we fix sets of arrivals and departures.

Remark 4.22. *For every arrival a in $\mathcal{A}$, the set of configurations*

$$C_1(a) = \{q \in \mathcal{Q} : a \in q\}$$

induces a clique in $G^{\mathcal{Q}}$.

Remark 4.23. *For every departure d in $\mathcal{D}$, the set of configurations*

$$C_1(d) = \{q \in \mathcal{Q} : d \in q\}$$

induces a clique in $G^{\mathcal{Q}}$.

Clique inequalities defined by classes $C_1(a)$ and $C_1(d)$ are implied by Constraints $(\mathbf{X})(i)$ and $(\mathbf{X})(ii)$, respectively.

Remark 4.24. *There are $2n$ cliques of classes $C_1(a)$ and $C_1(d)$, each containing $O(n^{2\beta-1})$ elements.*

Remark 4.25. *For every pair of subsequences $\mathcal{A}' \subseteq \mathcal{A}$ and $\mathcal{D}' \subseteq \mathcal{D}$, the set*

$$C_1(\mathcal{A}', \mathcal{D}') = \{q \in \mathcal{Q} : \mathcal{A}' \subseteq q(\mathcal{A}) \text{ and } \mathcal{D}' \subseteq q(\mathcal{D})\}$$

induces a clique in $G^{\mathcal{Q}}$ such that $C_1(\mathcal{A}', \mathcal{D}') \subseteq C_1(a)$ for every $a \in \mathcal{A}'$ and such that $C_1(\mathcal{A}', \mathcal{D}') \subseteq C_1(d)$ for every $d \in \mathcal{D}'$.

Lemma 4.26. *Let $\mathcal{A}'$ be a subsequence of $\mathcal{A}$ containing $3 \leq 2k-1 \leq 2\beta-1$ arrivals. The set of configurations*

$$C_2(\mathcal{A}') = \{q \in \mathcal{Q} : |q(\mathcal{A}) \cap \mathcal{A}'| \geq k\}$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. Each configuration in $C_2(\mathcal{A}')$ contains at least k elements from a set of size $2k-1$. Consequently, it follows from the Pigeonhole Principle that each pair of elements of $C_2(\mathcal{A}')$ has at least one element in common, i.e., $C_2(\mathcal{A}')$ is a clique.

Set $C_2(\mathcal{A}')$ is maximal because for every configuration q such that $|\mathcal{A}' \cap q(\mathcal{A})| < k$, there is at least one configuration q' in $C_2(\mathcal{A}')$ such that $q(\mathcal{A}) \cap q'(\mathcal{A}) = \emptyset$ and $q(\mathcal{D}) \cap q'(\mathcal{D}) = \emptyset$. Consequently, $\{q\} \cup C_2(\mathcal{A}')$ does not induce a clique in $G^{\mathcal{Q}}$. □

Clearly, a similar claim holds for subsets $\mathcal{D}'$ of $\mathcal{D}$.

Lemma 4.27. *Let $\mathcal{D}'$ be a subsequence of $\mathcal{D}$ containing $3 \leq 2k-1 \leq 2\beta - 1$ departures. The set of configurations*

$$C_2(\mathcal{D}') = \{q \in \mathcal{Q} : |q(\mathcal{D}) \cap \mathcal{D}'| \geq k\}$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Remark 4.28. *There are $\sum_{k=2}^{\beta} \binom{n}{2k-1}$ different sets $C_2(\mathcal{A}')$ ($C_2(\mathcal{D}')$) in $G^{\mathcal{Q}}$, and each of these sets has $O\left(\sum_{j=k}^{j\leq 2k-1} \binom{2k-1}{j}\binom{n-2k+1}{\beta-j}\binom{n}{\beta}\right)$ elements.*

Proof. It is possible to construct $\binom{n}{2k-1}$ subsets $\mathcal{A}'$ of $\mathcal{A}$ ($\mathcal{D}'$ of $\mathcal{D}$) of cardinality $2k-1$, $2 \leq k \leq \beta$.

The number of configurations containing j arrivals in $\mathcal{A}'$ (departures in $\mathcal{D}'$), $k \leq j \leq 2k-1$, is bounded by $\binom{2k-1}{j}\binom{n-2k+1}{\beta-j}\binom{n}{\beta}$, as there are $\binom{2k-1}{j}$ different choices for the j arrivals (departures) that belong to $\mathcal{A}'$ ($\mathcal{D}'$), $\binom{n-2k+1}{\beta-j}$ different choices for the other $\beta-j$ arrivals (departures), and $O\left(\binom{n}{\beta}\right)$ choices for the departures (arrivals) complementing the configuration. □

Theorem 4.29. *Let $\mathcal{A}'$ and $\mathcal{D}'$ be subsequences of $\mathcal{A}$ and $\mathcal{D}$, respectively, such that $|\mathcal{A}'| + |\mathcal{D}'| = 2k-1$, $3 \leq 2k-1 \leq 2\beta - 1$. The set of configurations*

$$C_2(\mathcal{A}', \mathcal{D}') = \{q \in \mathcal{Q} : |q(\mathcal{A}) \cap \mathcal{A}'| + |q(\mathcal{D}) \cap \mathcal{D}'| \geq k\}$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. Similar to the proofs of Lemmas 4.26 and 4.27. □

Remark 4.30. *There are $\sum_{k=2}^{\beta} \sum_{a=0}^{2k-1} \binom{n}{a}\binom{n}{2k-1-a}$ different sets $C_2(\mathcal{A}', \mathcal{D}')$ in $G^{\mathcal{Q}}$, and each of these sets has $O\left(\sum_{j=k}^{2k-1} \sum_{a=0}^{|\mathcal{A}'|} \binom{|\mathcal{A}'|}{a}\binom{n-|\mathcal{A}'|}{\beta-a}\binom{|\mathcal{D}'|}{j-a}\binom{n-|\mathcal{D}'|}{\beta-j+a}\right)$ elements.*

Proof. From the $2k-1$ elements of $\mathcal{A}' \cup \mathcal{D}'$, there are $\binom{n}{a}$ possible selections of $\mathcal{A}'$ and $\binom{n}{2k-1-a}$ possible selections of $\mathcal{D}'$, $0 \leq a \leq 2k-1$ and $2 \leq k \leq \beta$.

Configurations with j elements of $\mathcal{A}' \cup \mathcal{D}'$ have a elements from $\mathcal{A}'$ and $j-a$ elements from $\mathcal{D}'$. These configurations must cover $\beta - a$ arrivals out of $n - |\mathcal{A}'|$ elements of $\mathcal{A} \setminus \mathcal{A}'$ and $\beta - j + a$ departures out of $n - |\mathcal{D}'|$ elements of $\mathcal{D} \setminus \mathcal{D}'$. $\square$

Theorem 4.31. *Let $\mathcal{A}'$ and $\mathcal{D}'$ be subsequences of $\mathcal{A}$ and $\mathcal{D}$, respectively, such that $|\mathcal{A}'| + |\mathcal{D}'| = 2k-1$, $2\beta + 1 \leq 2k-1 \leq 4\beta - 3$, and such that one of the following conditions holds:*

- $1 \leq |\mathcal{A}'|, |\mathcal{D}'| < 2\beta$; *or*
- *if* $|\mathcal{A}'| = 2\beta + \alpha$, $\alpha \geq 0$, *then* $|\mathcal{D}'| \geq \alpha + 1$; *or*
- *if* $|\mathcal{D}'| = 2\beta + \alpha$, $\alpha \geq 0$, *then* $|\mathcal{A}'| \geq \alpha + 1$.

For such subsequences $\mathcal{A}'$ and $\mathcal{D}'$, the set of configurations

$$C_2^*(\mathcal{A}', \mathcal{D}') = \{q \in \mathcal{Q} : |q(\mathcal{A}) \cap \mathcal{A}'| + |q(\mathcal{D}) \cap \mathcal{D}'| \geq k\}$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. Initially, we show that $C_2^*(\mathcal{A}', \mathcal{D}')$ is not defined if none of the cardinality constraints indicated in the hypothesis holds. If $|\mathcal{D}'| = 0$ then $|\mathcal{A}'| = 2k-1 \geq 2\beta + 1$, and as there is no configuration q in $\mathcal{Q}$ such that $|q(\mathcal{A}) \cap \mathcal{A}'| + |q(\mathcal{D}) \cap \mathcal{D}'| = |q(\mathcal{A}) \cap \mathcal{A}'| \geq k = \beta + 1$, $C_2^*(\mathcal{A}', \mathcal{D}')$ is empty in this case. The same proof shows that we have a similar situation if $|\mathcal{A}'| = 0$.

If $|\mathcal{A}'| = 2\beta + \alpha$, $\alpha \geq 0$, any configuration in $C_2^*(\mathcal{A}', \mathcal{D}')$ should contain at least $\frac{\alpha}{2} + \frac{|\mathcal{D}'|}{2} + \frac{1}{2}$ elements of $|\mathcal{D}'|$. Therefore,

$$\begin{aligned}
|\mathcal{D}'| &\geq \frac{\alpha}{2} + \frac{|\mathcal{D}'|}{2} + \frac{1}{2} \\
2|\mathcal{D}'| &\geq \alpha + |\mathcal{D}'| + 1 \\
|\mathcal{D}'| &\geq \alpha + 1.
\end{aligned}$$

A similar argument shows that if $|\mathcal{D}'| = 2\beta + \alpha$, $\alpha \geq 0$, then $C_2^*(\mathcal{A}', \mathcal{D}')$ is non-empty if and only if $|\mathcal{A}'| \geq \alpha + 1$.

If the set $C_2^*(\mathcal{A}', \mathcal{D}')$ is defined (and non-empty), each of its configurations contains at least k elements in common with $\mathcal{A}' \cup \mathcal{D}'$. As $|\mathcal{A}'| \cup |\mathcal{D}'| = 2k-1$, it follows from the Pigeonhole Principle that each pair of configurations in $C_2^*(\mathcal{A}', \mathcal{D}')$ have at least one element in common.

Finally, given any configuration q' such that $|q'(\mathcal{A}) \cap \mathcal{A}'| + |q'(\mathcal{D}) \cap \mathcal{D}'| < k$ (which holds for every configuration that does not belong to $C_2^*(\mathcal{A}', \mathcal{D}')$), there is at least one configuration q in $C_2^*(\mathcal{A}', \mathcal{D}')$ such that $|q \cap q'| = 0$. Therefore, $C_2^*(\mathcal{A}', \mathcal{D}')$ can not be extended to larger clique in $G^{\mathcal{Q}}$, which shows that it is indeed maximal. □

4.3.4.3 Fixed Configuration

Let q be a configuration. We denote the family of sets containing β elements of $q(\mathcal{A}) \cup q(\mathcal{D})$ by $I(q)$. For each set S in $I(q)$, there is exactly one set S' in $I(q)$ such that $S \cap S' = \emptyset$. We say that S and S' are *complementing sets*.

If we take a subfamily $I' \subseteq I(q)$ containing exactly one member of each pair of complementing sets, the set of configurations

$$C^\beta(q, I') = \{q' \in \mathcal{Q} : |q' \cap q| = \beta \text{ and } \exists S \in I' \text{ s.t. } q' \supseteq S\}$$

induces a clique in $G^{\mathcal{Q}}$. We say that I' is a *maximal crossing subfamily* of $I(q)$.

Remark 4.32. *For each q, there are $2^{\binom{2\beta}{\beta}/2}$ different maximal crossing subfamilies I'. Consequently, there are $2^{\binom{2\beta}{\beta}/2}$ different sets $C^\beta(q, I')$ for every q, each containing $O\left(\binom{2n-2\beta}{\beta}\binom{2\beta}{\beta}\right)$ elements.*

Proof. There are $\binom{2\beta}{\beta}$ elements in $I(q)$ (and, therefore, $\binom{2\beta}{\beta}/2$ pairs of complementing sets) and we have to choose one element out of each pair of complementing sets in order to obtain a set I', which gives us $2^{\binom{2\beta}{\beta}/2}$ possibilities. Finally, each of the $\binom{2\beta}{\beta}/2$ sets S of I' is contained in $q(\mathcal{A}) \cup q(\mathcal{D})$ for $O(\binom{2n-2\beta}{\beta})$ configurations q of $\mathcal{Q}$. □

The cliques induced by sets $C^\beta(q, I')$ are not maximal, but they will be useful in some of the followings results.

Proposition 4.33. *Let q be a configuration in $\mathcal{Q}$ and I' be a maximal crossing subfamily of $I(q)$. The set*

$$C_3(q, I') = \{q' \in \mathcal{Q} : |q \cap q'| \geq \beta + 1\} \cup C^\beta(q, I')$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. Let q_1 and q_2 be two configurations in $C_3(q, I')$. If $|q_1 \cap q| \geq \beta + 1$ and $|q_2 \cap q| \geq \beta$, then it follows from the Pigeonhole Principle that $|q_1 \cap q_2| >$

0. If $|q_1 \cap q| = |q_2 \cap q| = \beta$, $q_1 \in C^{\beta}(q, I')$ and $q_2 \in C^{\beta}(q, I')$. In this case, it follows from the definition of $C^{\beta}(q, I')$ that $|q_1 \cap q_2| > 0$. Therefore, if $q_1, q_2 \in C_3(q, I')$, then $|q_1 \cap q_2| > 0$.

For every configuration q' such that $|q' \cap q| = \beta' < \beta$, it is possible to identify at least one configuration q'' such that $|q'' \cap q'| = 0$ and $|q'' \cap q| = 2\beta - \beta' > \beta$, i.e., q'' belongs to $C_3(q, I')$ and is not adjacent to q' in $G^{\mathcal{Q}}$. This observation allows us to infer that $C_3(q, I')$ is maximal clique in $G^{\mathcal{Q}}$. □

Remark 4.34. *For every configuration q, there are $2^{\binom{2\beta}{\beta}/2}$ sets $C_3(q, I')$, and each of these sets has $O\left(\binom{2\beta}{\beta+1}\binom{2n-\beta-1}{\beta-1} + \binom{2\beta}{\beta}\binom{2n-2\beta}{\beta}\right)$ elements.*

Proof. The number of families $C_3(q, I')$ for a given q is equal to the number of choices of I', which is $2^{\binom{2\beta}{\beta}/2}$ (see Remark 4.32).

There are $O\left(\binom{2\beta}{\beta+1}\binom{2n-\beta-1}{\beta-1}\right)$ configurations that share at least $\beta + 1$ elements with q, and from Remark 4.32 we know that each set $C^{\beta}(q, I')$ contains $O\left(\binom{2\beta}{\beta}\binom{2n-2\beta}{\beta}\right)$ configurations. Consequently, family $C_3(q, I')$ contains $O\left(\binom{2\beta}{\beta+1}\binom{2n-\beta-1}{\beta-1} + \binom{2\beta}{\beta}\binom{2n-2\beta}{\beta}\right)$ elements. □

It is possible to employ this idea in order to construct other families of inequalities if we consider intersections involving either only arrivals or only departures. Assume that β is even. For a given configuration q, we define $I^{\mathcal{A}}(q)$ ($I^{\mathcal{D}}(q)$) as the family of sets containing $\beta/2$ arrivals (departures) of q. For each set S in $I^{\mathcal{A}}(q)$ ($I^{\mathcal{D}}(q)$), there is exactly one set S' in $I^{\mathcal{A}}(q)$ ($I^{\mathcal{D}}(q)$) such that $S \cap S' = \emptyset$. We say that S and S' are complementing sets. If we take a subfamily $I' \subseteq I^{\mathcal{A}}(q)$ ($I^{\mathcal{D}}(q)$) containing exactly one member of each pair of complementing sets, then the sets of configurations

$$C^{\mathcal{A}}(q, I') = \{q' \in \mathcal{Q} : \exists S \in I' \text{ s.t. } |q'(\mathcal{A}) \cap S| = |q'(\mathcal{A}) \cap q(\mathcal{A})| = \beta/2\}$$

and

$$C^{\mathcal{D}}(q, I') = \{q' \in \mathcal{Q} : \exists S \in I' \text{ s.t. } |q'(\mathcal{D}) \cap S| = |q'(\mathcal{D}) \cap q(\mathcal{D})| = \beta/2\}$$

induce cliques in $G^{\mathcal{Q}}$. We will overcharge the notation and also refer to family I' as a maximal crossing subfamily of $I^{\mathcal{A}}(q)$ or $I^{\mathcal{D}}(q)$.

If β is odd, we assume that families $I^{\mathcal{A}}(q)$ and $I^{\mathcal{D}}(q)$ (and, consequently, sets $C^{\mathcal{A}}(q, I')$ and $C^{\mathcal{D}}(q, I')$) are empty.

Remark 4.35. *For each configuration q, there are $2^{\binom{\beta}{\beta/2}/2}$ different families I' in $I^{\mathcal{A}}(q)$ and in $I^{\mathcal{D}}(q)$, and consequently, $2^{\binom{\beta}{\beta/2}/2}$ different sets $C^{\mathcal{A}}(q, I')$ and $C^{\mathcal{D}}(q, I')$, each with cardinality $O\left(\binom{\beta}{\beta/2}\binom{n-\beta}{\beta/2}\binom{n}{\beta}\right)$.*

Similarly to $C^\beta(q,I')$, sets $C^{\mathcal{A}}(q,I')$ and $C^{\mathcal{D}}(q,I')$ do not induce maximal cliques in $G^{\mathcal{Q}}$, but they are useful in the definition of the families of inequalities discussed below.

Proposition 4.36. *Let q be a configuration in $\mathcal{Q}$ and I' be a maximal crossing subfamily of $I^{\mathcal{A}}(q)$. The set*

$$C_4^{\mathcal{A}}(q,I') = \{q' \in \mathcal{Q} : q(\mathcal{A}) \cap q'(\mathcal{A}) \geq \lfloor \beta/2 \rfloor + 1\} \cup C^{\mathcal{A}}(q,I')$$

induces a maximal clique in the graph $G^{\mathcal{Q}}$.

Proposition 4.37. *Let q be a configuration in $\mathcal{Q}$ and I' be a maximal crossing subfamily of $I^{\mathcal{D}}(q)$. The set*

$$C_4^{\mathcal{D}}(q,I') = \{q' \in \mathcal{Q} : q(\mathcal{D}) \cap q'(\mathcal{D}) \geq \lfloor \beta/2 \rfloor + 1\} \cup C^{\mathcal{D}}(q,I')$$

induces a maximal clique in the graph $G^{\mathcal{Q}}$.

Remark 4.38. *If β is odd, there is only one set $C_4^{\mathcal{A}}(q,I')$ for each q, and*

$$|C_4^{\mathcal{A}}(q,I')| = O\left(\sum_{j=\lfloor \beta/2 \rfloor+1}^{\beta} \binom{\beta}{j}\binom{n-\beta}{\beta-j}\binom{n}{\beta}\right).$$

If β is even, for each configuration q there are $2^{\binom{\beta}{\beta/2}/2}$ sets $C_4^{\mathcal{A}}(q,I')$ and

$$|C_4^{\mathcal{A}}(q,I')| = O\left(\sum_{j=\beta/2+1}^{\beta} \binom{\beta}{j}\binom{n-\beta}{\beta-j}\binom{n}{\beta} + \binom{\beta}{\beta/2}\binom{n-\beta}{\beta/2}\binom{n}{\beta}\right).$$

The same observations hold for $C_4^{\mathcal{D}}(q,I')$.

The proofs of Propositions 4.36 and 4.37 are similar to the proof of Proposition 4.33, and the proof of Remark 4.38 is similar to the proof of Remark 4.34.

4.3.4.4 Fixed Pairs of Configurations

The following families of cliques in $G^{\mathcal{Q}}$ are obtained from pairs of configurations which do not have elements in common.

Proposition 4.39. *Let $q, q' \in \mathcal{Q}$ be a pair of configurations such that $|q \cap q'| = 0$ and let I' be a maximal crossing subfamily of $I^{\mathcal{A}}(q)$. Assume that*

$$C'^{\mathcal{A}}(q,q',I') = \{q'' \in C^{\mathcal{A}}(q',I') : |q'' \cap q| \geq 1\}$$

and that

$$C''^{\mathcal{A}}(q, q', I') = \{q'' \in \mathcal{Q} : |q''(\mathcal{A}) \cap q'(\mathcal{A})| \geq \lfloor \beta/2 \rfloor + 1 \text{ and } |q'' \cap q| \geq 1\}.$$

The set

$$C_5^{\mathcal{A}}(q, q', I') = \{q\} \cup C''^{\mathcal{A}}(q, q', I') \cup C'^{\mathcal{A}}(q, q', I')$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. It is clear that q intersects with each element of $C_5^{\mathcal{A}}(q, q', I')$. Using the Pigeonhole Principle, we can verify that the configurations of set $C''^{\mathcal{A}}(q, q', I')$ are pairwise intersecting. Because each element of $C'^{\mathcal{A}}(q, q', I')$ belong to some set $C^{\mathcal{A}}(q', I')$, its members are also intersecting. Finally, if $C'^{\mathcal{A}}(q, q', I') \neq \emptyset$ (i.e., if β is even), because each $q_1 \in C'^{\mathcal{A}}(q, q', I')$ contains $\beta/2$ arrivals of $q'(\mathcal{A})$ and because each $q_2(\mathcal{A})$ in $C''^{\mathcal{A}}(q, q', I')$ contains at least $\beta/2 + 1$ arrivals of $q'(\mathcal{A})$, $|q_1 \cap q_2| > 0$. Therefore, we conclude that $C_5^{\mathcal{A}}(q, q', I')$ induces a clique in $G^{\mathcal{Q}}$.

Suppose q'' extends the clique $C_5^{\mathcal{A}}(q, q', I')$. Because q is a clique, it is clear that q'' must be such that $|q'' \cap q| \geq 1$, and as q'' does not belong to $C_5^{\mathcal{A}}(q, q', I')$, we can also assume that $|q''(\mathcal{A}) \cap q'(\mathcal{A})| < \lfloor \beta/2 \rfloor$. For any such configuration, it is possible to identify at least one configuration q^* such that $|q^*(\mathcal{A}) \cap q'(\mathcal{A})| > \lfloor \beta/2 \rfloor$, $|q^* \cap q| = 1$, and $|q^* \cap q''| = 0$, i.e., q^* belongs to $C_5^{\mathcal{A}}(q, q', I')$ and is not adjacent to q'' in $G^{\mathcal{Q}}$. This observation allows us to infer that $C_5^{\mathcal{A}}(q, q', I')$ induces a maximal clique in $G^{\mathcal{Q}}$. □

The same holds if we consider $\mathcal{D}$ instead of $\mathcal{A}$.

Proposition 4.40. *Let $q, q' \in \mathcal{Q}$ be a pair of configurations such that $|q \cap q'| = 0$ and let I' be a maximal crossing subfamily of $I^{\mathcal{D}}(q)$. Assume that*

$$C'^{\mathcal{D}}(q, q', I') = \{q'' \in C^{\mathcal{D}}(q', I') : |q'' \cap q| \geq 1\}$$

and that

$$C''^{\mathcal{D}}(q, q', I') = \{q'' \in \mathcal{Q} : |q''(\mathcal{D}) \cap q'(\mathcal{D})| \geq \lfloor \beta/2 \rfloor + 1 \text{ and } |q'' \cap q| \geq 1\}.$$

The set

$$C_5^{\mathcal{D}}(q, q', I') = \{q\} \cup C''^{\mathcal{D}}(q, q', I') \cup C'^{\mathcal{D}}(q, q', I')$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Remark 4.41. *If β is odd, there is only one set $C_5^{\mathcal{A}}(q,q',I')$ and*

$$|C_5^{\mathcal{A}}(q,q',I')| = O\left(\binom{\beta}{\lfloor\beta/2\rfloor+1}\binom{2\beta}{1}\binom{n-\lfloor\beta/2\rfloor-1}{\beta-\lfloor\beta/2\rfloor-1}\binom{n}{\beta}\right).$$

If β is even, there are $O\left(2^{\binom{\beta}{\beta/2}/2}\right)$ sets $C_5^{\mathcal{A}}(q,q',I')$ and

$$|C_5^{\mathcal{A}}(q,q',I')| = O\left(2\beta\binom{n}{\beta}\left(\binom{\beta}{\beta/2}\binom{n-\beta}{\beta/2}+\binom{\beta}{\beta/2+1}\binom{n-\beta/2-1}{\beta-\beta/2-1}\right)\right).$$

Proof. Similar to the proof of Remark 4.34. The estimate of $|C_5^{\mathcal{A}}(q,q',I')|$ is not tight, as it does not consider the intersection involving q and configurations in $C_5^{\mathcal{A}}(q,q',I')$. □

There are as many cliques of family $C_5^{\mathcal{D}}(q,q',I')$ as there are cliques of family $C_5^{\mathcal{A}}(q,q',I')$, and the analysis of their cardinalities is similar to the one presented in Remark 4.41.

Finally, we can consider intersections with both sets $\mathcal{A}$ and $\mathcal{D}$.

Proposition 4.42. *Let $q,q' \in \mathcal{Q}$ be a pair of configurations such that $|q \cap q'| = 0$ and let I' be a maximal crossing subfamily of $I(q)$. Assume that*

$$C^{'}(q,q',I') = \{q'' \in C^{\beta}(q',I') : |q'' \cap q| \geq 1\}$$

and that

$$C^{''}(q,q',I') = \{q'' \in \mathcal{Q} : |q'' \cap q'| \geq \beta+1 \text{ and } |q'' \cap q| \geq 1\}.$$

The set

$$C_5(q,q',I') = \{q\} \cup C^{''}(q,q',I') \cup C^{'}(q,q',I')$$

induces a maximal clique in $G^{\mathcal{Q}}$.

Proof. Similar to the proof of Proposition 4.39. □

Remark 4.43. *If β is odd, there is only one set $C_5(q,q',I')$ and*

$$|C_5(q,q',I')| = O\left(\binom{2\beta}{\beta+1}\binom{2\beta}{1}\binom{2n-\beta-2}{\beta-2}\right).$$

If β is even, there are $O\left(2^{\binom{2\beta}{\beta}/2}\right)$ sets $C_5(q,q',I')$ and

$$|C_5(q,q',I')| = O\left(2\beta\left(\binom{2n-2\beta-1}{\beta-1}\binom{2\beta}{\beta}+\binom{2\beta}{\beta+1}\binom{2n-\beta-2}{\beta-2}\right)\right).$$

Proof. If q'' is an element of $C"(q,q',I')$, we know that from the 2β elements covered by q'', at least $\beta+1$ belong to q'' and that at least one is also covered by q. Its remaining $\beta-2$ elements belong to the other $2n-\beta-2$ elements of $\mathcal{A}$ and $\mathcal{D}$. From this analysis, if follows that there are $O\left(\binom{2\beta}{\beta+1}\binom{2\beta}{1}\binom{2n-\beta-2}{\beta-2}\right)$ configurations q''.

Finally, if β is even, some configurations that have $\beta/2$ elements in common with q' also belong to $C_5(q,q',I')$. For such configurations, there are $\binom{2\beta}{\beta}/2$ possible choices for the elements belonging to the intersection, $\binom{2\beta}{1}$ choices for the common element with q, and $O\left(\binom{2n-2\beta}{\beta}\right)$ possibilities to complete the configurations (again, this is not a tight estimate). □

4.4 Computational Results

This section presents the results of a computational evaluation of our set partitioning approach. All computations were done on a 64-bit Intel(R) Core(TM)2 Quad with 2.83 GHz, 8 GB of RAM memory, running openSuse Linux 11.2. Our code is implemented in `C++` and was compiled using `g++` 4.4.1. We used the callable library of CPLEX 12.1.0 ILOG (2010) [57] to solve the LPs.

Our code solves formulations $(\mathbf{X}'')$ and $(\mathbf{X}^{p\,''})$ to minimize first crossings using the branch-and-price algorithm described in Section 4.3.3. For each node of the solution tree, we use the heuristics presented in Section 4.3.2 in order to generate a set of columns composing a feasible solution that respects all the fixed assignments. After this, we start with the column generation algorithm. In each step, we try to obtain a configuration with negative reduced cost using the routine described in 4.12, assuming that the non-fixed assignments follow the sequential matchings (see Section 2.4.1.1). If it fails, then we solve the exact recurrence described in 4.10. After computing a configuration q with minimal reduced cost, we employ the heuristics again (adapted to the fixed assignments) in order to generate matchings containing q. The configurations of these matchings are added to the problem and the resulting LP is resolved. This procedure is repeated until no more improving configuration exists, i.e., until the LP is solved to optimality. For Algorithm 4, we used $T = n^2$, $\alpha = 0.97$, $\mathit{outSteps} = 20$, and $\mathit{inSteps} = 1000$ as parameters.

We tested some of the cuts presented in Section 4.3.4, but preliminary results showed that their inclusion make the algorithm slower. Therefore, we did not used them in our final programs.

For both problems, two groups of instances were generated. The first contains relatively “few” vehicle types, while the second contains “many”. From a practical point of view, the first group is more relevant, as public transportation companies usually do not have a large number of different vehicle types. The second group is interesting from a theoretical point of view, as it shows that $(\mathbf{X}'')$ and $(\mathbf{X}^{p}{}'')$ are able to yield non-trivial lower bounds for some scenarios of the problems.

Table 6 reports computational results for instances of the VPP which were created as follows. Each sequence $\mathcal{A}$ was uniformly generated at random and the respective sequence $\mathcal{D}$ was obtained from $\mathcal{A}$ after the application of Algorithm 1 (i.e., $\mathcal{D}$ is a random permutation of $\mathcal{A}$). The names $n-m-t$ of the instances indicate the number n of arrivals and departures, the number m of tracks, and the number t of vehicle types. All the vehicle types have the same size, i.e., $l(a) = 1, a \in \mathcal{A}$. The columns list the number *Row* of rows, the number *Col* of configurations generated by the branch-and-price algorithm, the number NZ of non-zeros of the final reduced master problem, the value V_{LP} of the LP relaxation, the value V_{IP} of the best integer solution obtained during the computation, the lower bound LB for the minimum number of first crossings of any optimal integer solution, and the time to solve the instance in CPU seconds. Instances of the first group were solved exactly, and instances of the second group were given a time limit of three hours (or at least the time required for the solution of the root node).

Table 7 reports computational results for the VPP^P. The instances were generated in the same way as the ones we used for the VPP. The name $p-w-n-m-e-t$ of the instances indicates the number p of weekday periods, the number w of weekend periods, the number n of arrivals on weekday periods, the number m of tracks, the number e of weekend periods, and the number t of vehicle types. The columns show the same informations as in Table 7. We remark that the numbers of rows, columns, and non-zeros are the sum of the values of all the periods. Note that the number of periods for a real-world instance would be 14, with a morning and evening period for every day of the week. In other words, the instances named `10-4-*-*-*-*` correspond to a typical “standard week”. For every instance, we gave a time limit of thirty minutes (or at least the time required for the solution of the root node) for every period to be solved.

The performance of the algorithms for the “realistic” groups of instances

Instance	Row	Col	NZ	V_{LP}	V_{IP}	LB	Time
20-4-12	40	149	1679	2	2	2	1
40-5-15	80	127297	2163556	0.2	2	2	9705
63-7-8	126	135849	2581245	0	0	0	10063
64-8-12	128	79598	1353233	0	0	0	3701
96-12-7	192	24	600	0	0	0	1
150-15-6	300	30	930	0	0	0	1
160-20-10	320	40	1000	0	0	0	1
200-20-10	400	40	1240	0	0	0	1
40-2-4	80	6787	269854	0.781522	3	1	10800
57-3-10	114	21794	843788	0.899608	8	1	10800
80-4-32	160	34340	1405094	4.086957	13	5	10800
81-9-81	162	127143	2415678	1	1	1	9263
90-6-37	180	50925	1577724	1.391626	12	2	10800
120-6-25	240	43037	1762543	0.277460	14	1	10800
160-10-80	320	57624	1901436	0.662060	19	1	10800

Table 6: Solving the VPP using $\mathbf{X}''$.

can be considered satisfactory. It is interesting to notice that the instances of the multi-periodic scenarios are more challenging than their single-period counterparts and that the heuristics had a good performance.

The results also show that the instances of the second groups are very challenging. We remark that, for these cases, we solved directly the recurrence described in Theorem 4.10, as the use of the pricing routine described in Theorem 4.12 made the programs much slower. These instances require crossings, and we were able to prove this fact with the results of the linear relaxations.

Instance	Row	Col	NZ	V_{LP}	V_{IP}	LB	Time
4-2-20-4-15-5	210	996	11151	1	1	1	5
5-2-36-6-24-5	432	17246	224611	0	0	0	116
6-3-30-5-18-5	444	23296	303069	1	3	3	324
10-3-64-8-48-6	1536	113026	1922842	0	2	0	3147
10-4-80-10-56-8	2000	394759	6712449	0	8	0	12376
10-4-150-15-100-10	3700	218620	4594080	0	9	0	7214
10-4-120-12-90-8	3060	500433	10510547	0	13	0	18058
10-4-150-15-110-9	3800	153239	3221435	0	5	0	7155
5-2-36-6-24-15	432	73172	951249	6.0909	9	7	2205
6-3-30-5-18-20	444	37778	491341	7.4755	12	12	186
10-3-64-8-48-17	1536	621668	10415856	1.2	33	4	20254
10-4-80-10-56-20	2000	737861	12544639	1	35	2	9780
10-4-150-15-100-46	3700	777113	16321498	5.8132	99	8	23487
10-4-120-12-90-40	3060	756737	15892603	2.3333	105	10	23403
10-4-150-15-110-45	3801	775479	16287302	1	92	1	23480

Table 7: Solving the VPP^P using $\mathbf{X}^{p''}$

Chapter 5

The Progressive Approach

In this chapter, we introduce the Progressive Approach, a method that can be used to produce feasible solutions for certain ILPs quickly. We present exact algorithms for the VPP and for the VPP^P based on this technique and show through computational experiments that they have very satisfactory performances for large-scale scenarios of both problems.

5.1 Introduction

One of the main challenges faced by the formulations proposed for the VPP and for the VPP^P is the large number of valid assignments and, consequently, the large number of feasible solutions. This makes large-scale scenarios of these problems virtually intractable. It is somehow clear, though, that some assignments are more likely to appear in optimal solutions than others, and this fact was not explored previously. We propose in this chapter a methodology to solve ILP formulations of problems for which it is possible to estimate, for each pair of variables, which one is more likely to appear in an optimal solution, and show how it can be applied to the VPP and to the VPP^P. Our computational results show that it is possible to obtain exact solutions for large-scale scenarios of both problem in less than one hour of computation with algorithms based on our Progressive Approach.

The chapter is organized as follows. Initially, we present a generic and problem-independent description of the method in Section 5.2. In Sections 5.3 and 5.4 we apply the Progressive Approach to model (**LU**) and to a new model that explores the concept of uniform tracks, respectively. Finally, our computational experiments are reported in Section 5.5.

5.2 The Progressive Approach

The Progressive Approach is described in Algorithm 6:

Algorithm 6 The Progressive Approach

```
$r(M') := \emptyset$
$v(M') := \emptyset$
while $M' \neq M$ do
    add rows from $r(M) \setminus r(M')$ to $r(M')$
    add columns from $v(M) \setminus v(M')$ to $v(M')$
    $x := solve(M')$
    if $c(x) = LB(M)$ then
        break
    end if
end while
```

The term $LB(M)$ represents a known lower bound for M, $solve(M')$ returns an optimal solution of M', and $c(x)$ is the objective value of $x \in P(M')$. For every ILP Model M, let $r(M)$ and $v(M)$ denote its sets of rows and columns, respectively. Model M' is always a "sub-model" of M, that is, $r(M') \subseteq r(M)$ and $v(M') \subseteq v(M)$ are invariants of the algorithm. In addition to that, we want $r(M')$ and $v(M')$ to be such that if x is a feasible solution of M', then x is the projection of some element x' of M such that $c(x) = c(x')$. As a consequence, the addition of variables and constrains to M' can not be done in an arbitrary way.

In each step, Algorithm 6 computes an optimal solution x for M' and verifies if $c(x) = LB(M)$. If this is the case, the algorithm stops, as we found already a projection of an optimal solution for M. If not, some variables from $v(M) - v(M')$ and some constraints from $r(M) - r(M')$ are added to M', and the algorithm proceeds to the next iteration using the best solution found so far as a warm start. Algorithm 6 stops when M' becomes M, so this method is exact if executed until its completion.

We remark that the Progressive Approach has similarities with column generation approaches (and also with sifting), where variables are added and the optimality of the current solution is verified with the help of a pricing algorithm. This comparison helps us to identify the main weakness and the main strength of Algorithm 6. A pricing algorithm gives a formal criterion to add interesting variables and typically helps to prove the optimality and to stop the algorithm before all the variables have been added to the problem.

Apparently, it is not an easy task to develop a generic rigorous procedure with the same capabilities for Algorithm 6. Conversely, a column generation approach can only be applied to a restricted family of formulations, while the method described here has a wider scope of application.

Finally, we remark that an algorithm based on the Progressive Approach can be trivially transformed into a heuristic if its execution is interrupted when $c(x)$ is still less than $LB(M)$.

5.3 Progressive Approach Applied to (LU)

In Section 2.4.1, we investigated some aspects of the VPP in order to obtain valid inequalities, and our analysis showed that the concepts of *sequential matching* and *complete sequential matching* play an important role in the problem. For instance, Lemma 2.24 proves that sequential matchings always provide the best assignments involving a set of arrivals and a set of departures. Moreover, pairs involved in a large number of semi-crossings hardly belong to optimal solutions. These observations give us directions to establish a hierarchy of elements of $\mathcal{A} \times \mathcal{D}$.

Let $dist : \mathcal{A} \times \mathcal{D} \rightarrow \mathbb{N}$ denote the absolute difference between the position of a in $\mathcal{A}^{\tau(a)}$ and the position of d in $\mathcal{D}^{\tau(a)}$ for each pair $(a, d) \in \mathcal{A} \times \mathcal{D}$. For example, if a is the first arrival in $\mathcal{A}^{\tau(a)}$ and d is the third departure in $\mathcal{D}^{\tau(a)}$, then $dist(a, d) = 2$. If $\tau(a) \neq \tau(d)$, we assume that $dist(a, d) = \infty$.

Algorithm 7 Progressive Approach Applied to (**LU**)

```
for $(a, s, d) \in \mathcal{F}$ do
    add $r_{a,s,d}$ to $v(M')$
end for
$r(M') := r(M)$
$count := 0$
while $c(solve(M')) > LB(M)$ and $count < n$ do
    for $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ do
        if $dist(a, d) = count$ then
            add $x_{a,s,d}$ to $v(M')$
        end if
    end for
    $count := count + 1$
end while
```

Algorithm 7 is a procedure based on the Progressive Approach and on (**LU**) to solve the VPP. The algorithm keeps a current model M', which is a projection of M. In each iteration, new assignments (i.e., new variables) from $r(M) \setminus r(M')$ are added to M' according to the function d. In the k-th iteration, M' will contain the assignment (a, d) if and only if $dist(a, d) \leq k$. In particular, in the first iteration only assignments of the complete sequential matching belong to M'. Algorithm 7 stops after at most $max_{1\leq i \leq t} t_i$ steps, when M' becomes equal to M. Nontrivial values for $LB(M)$ can be obtained, e.g., from Inequalities 1 and 2.

Finally, if x is a solution of one of the models M' generated by Algorithm 7, then the solution x', given by

$$x'_i = \begin{cases} x_i & \text{if } i \in v(M'), \\ 0 & \text{if } i \in v(M) \setminus v(M'), \end{cases}$$

is a solution for M such that $c(x) = c(x')$. Therefore, the procedure can stop after the computation of any vector x such that $c(x) = LB(M)$.

The following proposition shows that, in some cases, Algorithm 7 will only stop when M' becomes M.

Proposition 5.1. *For some instances of* VPP*, Algorithm 7 can only find an optimal solution when M' becomes M.*

Proof. Consider the family of instances of VPP such that $m = t-1$, $t_i = m$ for each type $\mathcal{T}_i$ (and therefore $\beta = m + 1$), and the first β arrivals belong to different types (say, $\tau(a_i) = \mathcal{T}_i$ for $1 \leq i \leq t$). Furthermore, the types of the remaining vehicles are such that:

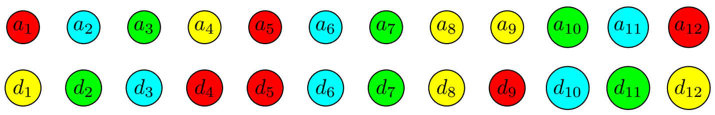
$$\tau(a_i) = \begin{cases} \tau(a_{((i-1) \mod \beta)+1}) & \text{if } \beta < i \leq \beta(m-1), \\ \tau(a_{(\beta-1-((i-1) \mod \beta))+1}) & \text{if } i > \beta(m-1). \end{cases}$$

The types of the departures are such that:

$$\tau(d_i) = \begin{cases} \tau(d_{(\beta-1-((i-1) \mod \beta))+1} & \text{if } i \leq \beta, \\ \tau(d_{((i-1) \mod \beta)+1} & \text{else.} \end{cases}$$

Figure 1 shows the instance of this family with $t = 4$.

Instances of this family admit a crossing-free solution, but such a matching can only be obtained if the last arrival is assigned to the first departure of each type. As a consequence, Algorithm 7 will be able to produce an



a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11} d_{12}

Figure 1: Example with four vehicle types

optimal solution only after the inclusion of the variables describing these assignments, which happens only in its last iteration. □

5.4 A New Model for the VPP

Kroon, Lentink & Schrijver (2006) [70] explored the concept of *uniform tracks*, i.e., tracks containing only vehicles of one type, in order to eliminate variables and inequalities from their formulations by fixing the number of uniform (and non-uniform) tracks. Based on this idea, we present the following "uniform-track-oriented" model:

$$
\begin{array}{llll}
(\mathbf{LF}) & \min \sum_{(a,s,d)} r_{a,s,d} & & \\
(\mathrm{i}) & \sum_{(s,d)} x_{a,s,d} & \leq 1 & a \in \mathcal{A} \\
(\mathrm{ii}) & \sum_{(a,s)} x_{a,s,d} & \leq 1 & d \in \mathcal{D} \\
(\mathrm{iii}) & \sum_{(a,d)} x_{a,s,d} & = \beta & s \in \mathcal{Z} \\
(\mathrm{iv}) & \sum_{(a,s,d)} x_{a,s,d} - u_i \beta & = m^i & \mathcal{T}_i \in \mathcal{T} \\
(\mathrm{v}) & \sum_{a'<a} x_{a',s,d} + \sum_{d' \leq d} x_{a,s,d'} - r_{a,s,d} & \leq 1 & (a,s,d) \in \mathcal{A} \times \mathcal{Z} \times \mathcal{D} \\
 & x_{a,s,d} & \in \{0,1\} & {\scriptstyle (a,s,d) \in \mathcal{A} \times \mathcal{Z} \times \mathcal{D}, \ \tau(a) = \tau(d)} \\
 & r_{a,s,d} & \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{Z} \times \mathcal{D} \\
 & u_i & \in \mathbb{N} & 0 \leq i < t.
\end{array}
$$

The set $\mathcal{Z} \subseteq \mathcal{S}$ contains tracks that may contain non-uniform configurations. For each type $\mathcal{T}_i$, $m^i = t_i \mod \beta$. Model (**LF**) employs binary variables $x_{a,s,d}$, with $a \in \mathcal{A}$, $s \in \mathcal{Z}$, $d \in \mathcal{D}$, and $\tau(a) = \tau(d)$, where $x_{a,s,d} = 1$ whenever arriving vehicle a is assigned to track s and to departing trip d, $\tau(a) = \tau(d)$. Binary variables $r_{a,s,d}$ indicate if there is a crossing involving arrival a and departure d in track s. Integer variables u_i count the number of uniform tracks containing units of type $\mathcal{T}_i$ that can be transformed in non-uniform tracks. Equations (**LF**)(i) and (**LF**)(ii) assert that each arriving vehicle and departing trip is assigned to at most one non-uniform track, respectively. We remark that (**LF**)(i) and (**LF**)(ii) can not be equalities because assignments involving the $|\mathcal{S}| - |\mathcal{Z}|$ uniform tracks do not have to be explicitly represented in a feasible solution of (**LF**). Equalities (**LF**)(iii) indicate that exactly β vehicles are parked in each track of $\mathcal{Z}$; (**LF**)(iv) defines the amount of vehicles of each type that can be assigned to non-uniform tracks; and (**LF**)(v) counts the number of crossings (which can only occur in non-uniform tracks).

The main idea of the formulation is that, for each vehicle type $\mathcal{T}_i$, if we assign ϕ_i units to $|\mathcal{Z}|$ (possibly non-uniform) tracks, $m^i \leq \phi_i \leq u_i\beta + m^i \leq t_i$, the remaining $t_i - \phi_i$ vehicles can be trivially parked in uniform tracks. Kroon, Lentink & Schrijver (2006) [70] assumed in their models that $|\mathcal{S} \setminus \mathcal{Z}|$ is fixed and greater than zero, but we can only be certain that (**LF**) will deliver an optimal solution if $\mathcal{Z} = \mathcal{S}$. In this case, (**LF**) can be seen as an expansion of (**LU**) where the number of non-uniform tracks is explicitly represented by variables u_i, $1 \leq i \leq t$.

It is clear that it is not interesting to solve the VPP directly using (**LF**), as (**LU**) is clearly easier. Instead, we developed Algorithm 8, a procedure based on the Progressive Approach in which $|\mathcal{Z}|$ is increased in each iteration. In the first step, $|\mathcal{Z}| = \sum_{\mathcal{T}_i \in \mathcal{T}} m^i/\beta$, as this is the lower bound on the number of non-uniform tracks (Proposition 4.18). In each iteration, a new track is added to $\mathcal{Z}$, i.e., a track which was previously uniform becomes available for the composition of non-uniform configurations. Finally, similarly to what we had in Algorithm 7, assignment variables are also progressively added to the current sub-problem according to the function $dist : \mathcal{A} \rightarrow \mathcal{D}$. The algorithm stops when $\mathcal{Z}$ becomes $\mathcal{S}$ or when it formally concludes that the best solution found so far is optimal for the original problem.

It is reasonable to expect good solutions after few iterations of Algorithm 8. However, worst-case scenarios can also appear, as the following proposition shows:

Proposition 5.2. *For some instances of* VPP*, Algorithm 8 can only find an optimal solution in its last possible step.*

Proof. Consider the family of instances whose first $(m-1)\beta$ arrivals and departures are of type $\mathcal{T}_1$ and the remaining β units are of pairwise different types, with $\tau(a_{(m-1)\beta+i}) = \tau(d_{m\beta-i+1})$, $1 \leq i \leq \beta$. Figure 2 shows an example with 9 elements and 3 tracks.

Algorithm 8 Progressive Approach applied to (**LF**)

```
select $\mathcal{Z} \subseteq \mathcal{S}$ such that $|SS| = (\sum_{t \in \mathcal{T}} mt/\beta) - 1$
for $(a, s, d) \in \mathcal{A} \times \mathcal{Z} \times \mathcal{D}$ do
  add $r_{a,s,d}$ to $v(M')$
end for
for $i = 0$ to $t - 1$ do
  add $u_i$ to $v(M')$
end for
$r(M') := r(M)$
while $\mathcal{Z} \neq \mathcal{S}$ do
  select track $s$ in $\mathcal{S} \setminus \mathcal{Z}$
  $\mathcal{Z} := \{s\} \cup \mathcal{Z}$
  for $(a, d) \in \mathcal{A} \times \mathcal{D}$ such that $\tau(a) = \tau(d)$ do
    add $r_{a,s,d}$ to $v(M')$
  end for
  for $count = 0$ to $n$ do
    for $(a, d) \in \mathcal{A} \times \mathcal{D}$ do
      if $dist(a, d) = count$ then
        add $x_{a,s,d}$ to $v(M')$
      end if
    end for
    if $c(solve(M')) = LB(M)$ then
      stop algorithm
    end if
  end for
end while
```

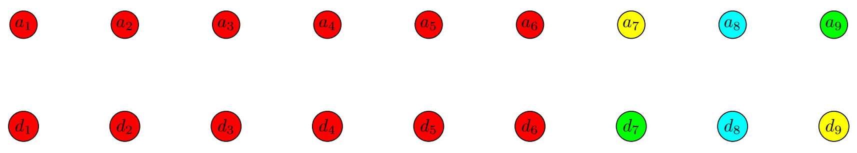


Figure 2: Example with nine units and three tracks

The configurations composing the optimal solutions instances of this family are as follows: the first $\beta - 1$ units belong to type $\mathcal{T}_1$ and the last contains a vehicle a_i such that $i > (m-1)\beta$. Consequently, an optimal solution can only be found if we allow all the tracks to be non-uniform, so Algorithm 8 will yield an optimal solution for instances of this family only when $\mathcal{Z}$ becomes $\mathcal{S}$, i.e., in its last iteration. $\square$

5.5 Computational Results

This section contains the results of a computational evaluation of the algorithms presented in this chapter. Our tests were conducted on a 64-bit Intel(R) Core(TM)2 Quad with 2.83 GHz, 8 GB of RAM memory, running openSuse Linux 11.2. Our code is implemented in `C++` and was compiled using `g++` 4.4.1 and with the callable library of CPLEX 12.1.0 (ILOG (2010) [57]).

Tables 8 and 9 show the results of computations of Algorithms 7 and 8 with instances of the VPP, respectively. In the case of the Algorithm 8, we also use the global lower bound equations presented in Section 2.4.1 for each pair (a, d) in $\mathcal{A} \times \mathcal{D}$ in order to improve the running time. The first column of these tables contains the name n-m-t of the instances. Following our notation, n is the number of arrivals, m is the number of tracks, and t is the number of types. Based on these parameters, the arrival sequences $\mathcal{A}$ were randomly generated (i.e., the type of each vehicle was uniformly chosen among the t possibilities), while their respective departure sequences $\mathcal{D}$ were obtained after the application of Algorithm 1 in $\mathcal{A}$. Columns Iterations, First Solution, V_{IP}, and Time indicate the number of iterations, the cost value of the solution obtained in the first iteration, the optimal value, and the time consumed for the computation of an optimal solution for the respective instance. With both algorithms, we were able to obtain very quickly optimal solutions for all the cases in less than 30 minutes.

Comparing these two tables, we can see that Algorithm 8 does not need many iterations, which can also be seen as a strong reason to explore uniformity of tracks to solve the VPP. However, even with more iterations, Algorithm 7 presented a better performance, specially on the harder and larger instances, and a smaller consumption of memory. For these reasons, we chose model (**LU**) in order to formulate and solve VPP^P.

Our results involving Algorithm 7 adapted to the VPP^P are presented in

Instance	Iterations	First Solution	V_{IP}	Time
20-4-12	6	4	4	1
40-5-15	8	5	1	830
63-7-8	1	0	0	1
64-8-12	1	0	0	1
96-12-7	1	0	0	5
150-15-6	1	0	0	27
150-15-15	1	0	0	34
160-20-10	1	0	0	59
160-20-40	1	0	0	698
200-20-10	1	0	0	185

Table 8: Solving the VPP using (**LU**)

Table 10. We generated the instances using Algorithm 1 and Algorithm 2 following the scheme described in Chapter 2. The names p-w-n-m-e-t of the instances indicate the number p of weekday periods, the number w of weekend periods, the number n of arrivals on weekday periods, the number m of tracks, the number e of trips on weekend periods, and the number t of vehicle types. In order to solve these instances, we compute the optimal solution for the first and the last weekday period simultaneously, while the other periods were solved independently. So, the number of iterations indicated in the second column of Table 8 is the sum of the number of iterations used on the computation of all the periods. Finally, Columns V_{IP} and Time indicate the solution obtained by the algorithm and its running time, respectively. All these solution are optimal and could be computed with modest time consumption.

Instance	Iterations	First Solution	V_{IP}	Time
20-4-12	2	5	4	0
40-5-15	1	1	1	45
63-7-8	1	0	0	3
64-8-12	1	0	0	3
96-12-7	1	0	0	4
150-15-6	1	0	0	3
150-15-15	1	0	0	391
160-20-10	1	0	0	28
160-20-40	1	0	0	1358
200-20-10	1	0	0	202

Table 9: Solving the VPP using (**LF**)

Instance	Iterations	V_{IP}	Time
4-2-20-4-15-5	11	2	1
5-2-36-6-24-5	10	0	1
6-3-30-5-18-5	25	3	9
10-3-64-8-48-6	13	0	11
10-4-80-10-56-8	14	0	28
10-4-120-12-90-8	14	0	146
10-4-150-15-110-9	14	0	487
10-4-150-15-100-10	14	0	452

Table 10: Solving the VPP^P using (**LU**)

Chapter 6

Extensions of the Vehicle Positioning Problem

In this chapter, we investigate two important extensions of the Vehicle Positioning Problem: the VPP$_+$ and the VPP$_+^P$. We introduce formulations of the VPP$_+$, which are classified according to the way assignment decisions are decomposed, and compare them from a theoretical and from a computational point of view. We discuss exact and heuristic solution methods for the VPP$_+^P$ and present the results of our computational experiments with the most satisfactory one, which is based on the Progressive Approach.

6.1 Introduction

Previously, we presented the basic versions of the VPP and of the VPP^P and efficient methods to solve them. For certain real-world scenarios, though, additional aspects must be taken into account, and some modifications may lead to significant changes in these problems. Therefore, new solution methods should be developed.

In this chapter, we investigate the VPP$_+$ and the VPP$_+^P$, which are important extensions of the VPP and of the VPP^P, respectively. Several formulations and solution approaches are suggested and their performances and properties are theoretically and computationally compared. The most important result is an algorithm for the VPP$_+^P$ based on the Progressive Approach, which is able to produce satisfactory solutions for large-scale scenarios of the problem.

This chapter is divided as follows. In Section 6.2 we introduce the new aspects addressed by the VPP$_+$ and by the VPP$_+^P$. In Section 6.3 we introduce exact ILP models for the VPP$_+$ and compare them from a theoretical and from a computational point of view. Our results show that the strength of these formulations is proportional to the number of elements that compose the indices of the variables representing assignment decisions. Finally, in Section 6.4, we present exact and heuristic methods for the VPP$_{+}^{P}$ and discuss their applicability in practical scenarios.

6.2 Additional Aspects

In the previous chapters, we assumed that each trip could only be assigned to one type of vehicle. However, some transportation companies have more flexible policies and let each departure d be serviced by any vehicle whose type belongs to the set $\mathcal{T}(d) \subseteq \mathcal{T}$. Conversely, due to infrastructure constraints, a track s may only be used by a vehicle if its type belongs to the set $\mathcal{T}(s) \subseteq \mathcal{T}$. We say that $\mathcal{T}(d)$ and $\mathcal{T}(s)$ are *nontrivial* if $|\mathcal{T}(d)| > 1$ and $\mathcal{T}(s) \neq \mathcal{T}$, respectively.

For the VPP, these new aspects do not substantially change the problem. Namely, it is possible to adapt the models presented in the previous chapters in order to include and exclude assignment variables according to the contents of all sets $\mathcal{T}(d)$ and $\mathcal{T}(s)$.

In contrast, the VPP^{P} becomes more challenging with nontrivial sets $\mathcal{T}(d)$ or $\mathcal{T}(s)$. Recall that the relation between departures in period h and arrivals in period $h+1$ is given by the function $sync : \mathcal{D} \rightarrow \mathcal{A}$. If $|\mathcal{T}(d)| = 1$, we know the type of arriving vehicle $sync(d)$, but if $\mathcal{T}(d)$ is nontrivial, arrival $sync(d)$ can be of any type in $\mathcal{T}(sync(d)) = \mathcal{T}(d)$. Moreover, if $sync(d)$ is assigned to d', we must ensure that d and d' are assigned to the same vehicle type. Therefore, solutions for scenarios with nontrivial sets $\mathcal{T}(d)$ require the *synchronization* of consecutive periods. The most important consequence of this fact is the necessity to consider the *assignment of vehicle types to arrivals and to departures* in the ILP formulations.

A careful selection of the e vehicles that will service trips during the weekend periods is essential when there are nontrivial sets $\mathcal{T}(d)$ and $\mathcal{T}(s)$. As we will see, bad choices can lead to infeasible scenarios.

Another additional aspect that will be considered is the Δ *restriction*, which refers to the difference of time Δ that must pass between any two departures assigned to the same track. This constraint is important in real-world scenarios and, to the best of our knowledge, it was not considered in any previous work.

We denote by VPP$_+$ and VPP$_+^P$ the extensions of VPP and VPP^P, respectively, that incorporate these new aspects.

6.3 Models for the VPP$_+$

In this section, we present ILP formulations for the VPP$_+$. We classify the models according to the number of elements that compose the indices of the variables and compare them from a theoretical and from a computational point of view.

Our goal is to identify a suitable base model for the VPP$_+^P$. For this reason, our formulations will compute assignments $\mathcal{D} \rightarrow \mathcal{T}$ and $\mathcal{A} \rightarrow \mathcal{T}$, i.e., we will not assume that the types of the arrivals are known in advance.

6.3.1 Quad-Index Model

Formulation (**QI**) can be seen as a natural extension of (**LU**), as each feasible assignment in $\mathcal{A} \times \mathcal{S} \times \mathcal{D} \times \mathcal{T}$ is represented by one variable. This model employs binary variables $x_{a,s,d,t}$ and $r_{a,s,d}$. Variable $x_{a,s,d,t}$ represents the assignment of arriving vehicle a to track s, to departing trip d, and to vehicle type $\mathcal{T}_t$; it belongs to the program if and only if $\mathcal{T}_t \in \mathcal{T}(a)$, $\mathcal{T}_t \in \mathcal{T}(s)$, and $\mathcal{T}_t \in \mathcal{T}(d)$. Variable $r_{a,s,d}$ indicates if there is a crossing involving arrival a and departure d on track s. Equalities (**QI**)(i) and (**QI**)(ii) are the assignment constraints for arriving vehicles and departing trips, respectively. Equalities (**QI**)(iii) indicate how many vehicles of each type are available, while the capacities of the tracks are controlled in (**QI**)(iv). The number of crossings is calculated in (**QI**)(v), which is similar to (**LU**)(*iv*), and (**QI**)(vi) is the formulation of the Δ restrictions.

$$
\begin{array}{llll}
(\mathbf{QI}) & \min & \sum\limits_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & & \sum\limits_{(s,d,t)} x_{a,s,d,t} = 1 & a \in \mathcal{A} \\
\text{(ii)} & & \sum\limits_{(a,s,t)} x_{a,s,d,t} = 1 & d \in \mathcal{D} \\
\text{(iii)} & & \sum\limits_{(a,s,d)} x_{a,s,d,t} = t_t & \mathcal{T}_t \in \mathcal{T} \\
\text{(iv)} & & \sum\limits_{(a,d,t)} x_{a,s,d,t} = \beta & s \in \mathcal{S} \\
\text{(v)} & \sum\limits_{a'<a} x_{a',s,d,t} + & \sum\limits_{d'<d} x_{a,s,d',t} \leq 1 + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
\text{(vi)} & & \sum\limits_{\substack{(a,d',t) \\ |d'-d|<\Delta}} x_{a,s,d',t} \leq 1 & (s,d) \in \mathcal{S} \times \mathcal{D} \\
& & x_{a,s,d,t} \in \{0,1\} & {\scriptstyle (a,s,d,t) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \times \mathcal{T}} \atop {\scriptstyle t \in \mathcal{T}(a) \cap \mathcal{T}(d) \cap \mathcal{T}(s)} \\
& & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

Proposition 2.26 shows that one can omit variable $r_{a,s,d}$ (and its respective inequality) from (**LU**) if $\tau(a) = \tau(d)$. We remark that this simplification is not applicable to (**QI**) if $\mathcal{T}(a)$ or $\mathcal{T}(d)$ is nontrivial, as we do not know in advance the type of a or d.

6.3.2 Triple-Index Models

In Formulation (**TI**), assignment decisions are decomposed into two classes of variables. The model employs binary variables $x_{a,s,d}$, $y_{a,s,t}$, and $r_{a,s,d}$. Variable $x_{a,s,d}$ represents the assignment of arriving vehicle a to track s and to departing trip d; it belongs to the formulation if and only if $\mathcal{T}(a) \cap \mathcal{T}(d) \cap \mathcal{T}(s) \neq \emptyset$. Variable $y_{a,s,t}$ models the assignment of arrival a to track s and to vehicle type $\mathcal{T}_t$; it belongs to the formulation if and only if $\mathcal{T}_t \in \mathcal{T}(a)$ and $\mathcal{T}_t \in \mathcal{T}(s)$. Finally, variable $r_{a,s,d}$ indicates if there is a crossing involving arriving vehicle a and departing trip d on track s. Equalities (**TI**)(i), (**TI**)(ii), and (**TI**)(iii) are the assignment constraints for arrivals and departures. The number of vehicles of each type and the capacity of the tracks are controlled in (**TI**)(iv) and (**TI**)(v), respectively. The coherence among variables $x_{a,s,d}$ and $y_{a,s,t}$ is modeled in (**TI**)(vi) and (**TI**)(vii). Finally, crossings are counted in (**TI**)(viii) and the Δ restrictions are formulated in (**TI**)(ix). Similarly to (**QI**), Proposition 2.26 also does not help to reduce the number of variables $r_{a,s,d}$ and Inequalities (**TI**)(viii) if $\mathcal{T}(a)$ or $\mathcal{T}(d)$ is nontrivial.

$$
\begin{array}{lrll}
(\mathbf{TI}) & \min & \displaystyle\sum_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & & \displaystyle\sum_{(s,t)} y_{a,s,t} = 1 & a \in \mathcal{A} \\
\text{(iv)} & & \displaystyle\sum_{(a,s)} y_{a,s,t} = t_t & \mathcal{T}_t \in \mathcal{T} \\
\text{(v)} & & \displaystyle\sum_{(a,t)} y_{a,s,t} = \beta & s \in \mathcal{S} \\
\text{(vi)} & & y_{a,s,t} + \displaystyle\sum_{\substack{d \\ t \notin \mathcal{T}(d)}} x_{a,s,d} \leq 1 & (a,s,t) \in \mathcal{A} \times \mathcal{S} \times \mathcal{T} \\
\text{(vii)} & & x_{a,s,d} + \displaystyle\sum_{\substack{t \\ t \notin \mathcal{T}(d)}} y_{a,s,t} \leq 1 & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
\text{(viii)} & & \displaystyle\sum_{a' < a} x_{a',s,d} + \sum_{d' < d} x_{a,s,d'} \leq 1 + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
\text{(ix)} & & \displaystyle\sum_{\substack{(a,d') \\ |d'-d| < \Delta}} x_{a,s,d'} \leq 1 & (s,d) \in \mathcal{S} \times \mathcal{D} \\
 & & x_{a,s,d} \in \{0,1\} & \substack{(a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\ \mathcal{T}(a) \cap \mathcal{T}(s) \cap \mathcal{T}(d) \neq \emptyset} \\
 & & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & & y_{a,s,t} \in \{0,1\} & \substack{(a,s,t) \in \mathcal{A} \times \mathcal{S} \times \mathcal{T} \\ t \in \mathcal{T}(a) \cap \mathcal{T}(s)} .
\end{array}
$$

Formulation (**TI2**) also makes a decomposition of the assignments in two groups of variables. The model employs binary variables $x_{a,s,d}$, $z_{a,t,d}$, and $r_{a,s,d}$. Variable $x_{a,s,d}$ represents the assignment of arriving vehicle a to track s and to departing trip d; it belongs to the formulation if and only if $\mathcal{T}(a) \cap \mathcal{T}(d) \cap \mathcal{T}(s) \neq \emptyset$. Variable $z_{a,t,d}$ denotes the assignment of arrival a to vehicle type $\mathcal{T}_t$ and to departure d; it belongs to the formulation if and only if $\mathcal{T}_t \in \mathcal{T}(a)$ and $\mathcal{T}_t \in \mathcal{T}(d)$. Finally, variable $r_{a,s,d}$ indicates if there is a crossing involving arriving vehicle a and departing trip d on track s. Equalities (**TI2**)(i), (**TI2**)(ii), (**TI2**)(iii), and (**TI2**)(iv) are the assignment constraints for arrivals and departures. The number of vehicles of each type and the capacity of the tracks are controlled in (**TI2**)(v) and (**TI2**)(vi), respectively. The coherence among variables $x_{a,s,d}$ and $y_{a,s,t}$ is modeled in (**TI2**)(vii). Finally, the crossings are counted in (**TI2**)(viii) and the Δ restrictions are formulated in (**TI2**)(ix). Again, the reductions proposed in Proposition 2.26 cannot be always applied in (**TI2**).

$$
\begin{array}{llll}
(\textbf{TI2}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d} & & \\
\text{(i)} & \displaystyle\sum_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(t,d)} z_{a,t,d} = 1 & a \in \mathcal{A} \\
\text{(iv)} & \displaystyle\sum_{(a,t)} z_{a,t,d} = 1 & d \in \mathcal{D} \\
\text{(v)} & \displaystyle\sum_{(a,d)} z_{a,t,d} = t_t & \mathcal{T}_t \in \mathcal{T} \\
\text{(vi)} & \displaystyle\sum_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(vii)} & \displaystyle\sum_{s} x_{a,s,d} = \sum_{t} z_{a,t,d} & (a,d) \in \mathcal{A} \times \mathcal{D} \\
\text{(viii)} & \displaystyle\sum_{a'<a} x_{a',s,d} + \sum_{d'<d} x_{a,s,d'} \leq 1 + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
\text{(ix)} & \displaystyle\sum_{\substack{(a,d') \\ |d'-d|<\Delta}} x_{a,s,d'} \leq 1 & (s,d) \in \mathcal{S} \times \mathcal{D} \\
 & x_{a,s,d} \in \{0,1\} & {\scriptstyle (a,s,d)\in\mathcal{A}\times\mathcal{S}\times\mathcal{D} \atop \mathcal{T}(a)\cap\mathcal{T}(s)\cap\mathcal{T}(d)\neq\emptyset} \\
 & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & z_{a,t,d} \in \{0,1\} & {\scriptstyle (a,t,d)\in\mathcal{A}\times\mathcal{T}\times\mathcal{D} \atop t\in\mathcal{T}(a)\cap\mathcal{T}(d)} .
\end{array}
$$

6.3.3 Theoretical Aspects

The following result shows that the VPP$_+$ is difficult from the theoretical point of view.

Corollary 6.1. *The* VPP$_+$ *is* $\mathcal{NP}$*-complete.*

Proof. This claim follows from the fact that each instance of the VPP is an instance of the VPP$_+$ where $|\mathcal{T}(d)| = 1$ for every departure d, $\mathcal{T}(s) = \mathcal{T}$ for every track s, and $\Delta = 0$, i.e., the Δ restrictions are trivially satisfied by any feasible assignment. As the VPP is $\mathcal{NP}$-complete (Winter (1998) [91]), we conclude that the VPP$_+$ is also $\mathcal{NP}$-complete. □

The following remarks describe the size of the programs generated by the models that we presented for the VPP$_+$:

Remark 6.2. *Model (**TI**) uses $O(mn^2)$ variables and $O(mn^2)$ constraints.*

Remark 6.3. *Model (**TI2**) uses $O(mn^2 + tn^2)$ variables and $O(mn^2)$ constraints.*

Remark 6.4. *Model (**QI**) uses $O(mtn^2)$ variables and $O(mn^2)$ constraints.*

Memory consumption plays an important role in the solution of the VPP$_+^P$. Regarding this aspect, Remarks 6.2, 6.3, and 6.4 suggest that (**TI**) and (**TI2**) are more interesting than (**QI**). However, as the following results show, the use of coherence constraints by the three-index models has important theoretical consequences.

Proposition 6.5. *For every solution (x, r) in P_{LP}(**QI**), there is a solution (x^t, y, r) in P_{LP}(**TI**).*

Proof. Let (x, r) be an element of $P_{LP}(\mathbf{QI})$. We define a pair (x^t, y) as follows:

$$\begin{aligned} x^t_{a,s,d} &= \sum_t x_{a,s,d,t}; \\ y_{a,s,t} &= \sum_d x_{a,s,d,t}. \end{aligned}$$

We show that (x^t, y) is a feasible solution of (**TI**):

$$\begin{array}{llll}
(i) & \sum\limits_{(s,d)} x^t_{a,s,d} = \sum\limits_{(s,d,t)} x_{a,s,d,t} = 1 & a \in \mathcal{A} \\
(ii) & \sum\limits_{(a,s)} x^t_{a,s,d} = \sum\limits_{(a,s,t)} x_{a,s,d,t} = 1 & d \in \mathcal{D} \\
(iii) & \sum\limits_{(s,t)} y_{a,s,t} = \sum\limits_{(d,s,t)} x_{a,s,d,t} = 1 & a \in \mathcal{A} \\
(iv) & \sum\limits_{(a,s)} y_{a,s,t} = \sum\limits_{(a,s,d)} x_{a,s,d,t} = t_t & \mathcal{T}_t \in \mathcal{T} \\
(v) & \sum\limits_{(a,t)} y_{a,s,t} = \sum\limits_{(a,d,t)} x_{a,s,d,t} = \beta & s \in \mathcal{S} \\
(ix) & \sum\limits_{\substack{(a,d') \\ |d'-d|<\Delta}} x^t_{a,s,d'} = \sum\limits_{\substack{(t,a,d') \\ |d'-d|\leq\Delta}} x_{a,s,d',t} \leq 1 & (s,d) \in \mathcal{S} \times \mathcal{D}
\end{array}$$

Because there is no incoherence in x, if $\sum\limits_{\substack{d \\ t\in\mathcal{T}(d)}} x_{a,s,d,t} = k$, then $\sum\limits_{\substack{d \\ t\notin\mathcal{T}(d)}} x_{a,s,d,t} \leq 1 - k$ for every (a, s, t) in $\mathcal{A} \times \mathcal{S} \times \mathcal{T}$, which implies (**TI**)(vi). A similar argument shows that (x^t, y) also satisfies (**TI**)(vii).

Finally, for all $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$, we have

$$\sum_{a'<a} x^t_{a',s,d} + \sum_{d'<d} x^t_{a,s,d'} \;=\; \sum_{\substack{t\\a'<a}} x_{a',s,d,t} + \sum_{\substack{t\\d'<d}} x_{a,s,d',t} \;\leq 1 + r_{a,s,d},$$

which shows that (x^t, y, r) belongs to $P_{LP}(\mathbf{TI})$. $\square$

Theorem 6.6. *There are elements (x, y) of $P_{LP}(\textbf{\textit{TI}})$ representing solutions of the* VPP$_+$ *which are not associated with elements of $P_{LP}(\textbf{\textit{QI}})$.*

Proof. We present an instance for which there is an element of $P_{LP}(\mathbf{TI})$ describing a solution that can not be represented by any element of $P_{LP}(\mathbf{QI})$. It has four vehicle types, with $t_1 = 2$ and $t_i = 1$, $2 \leq i \leq 4$. For each arrival a in $\mathcal{A}$, we have $\mathcal{T}(a) = \mathcal{T}$. We assume that $m = 1$, so the track can be neglected. Departures in $\{d_1, d_2, d_3\}$ can only be serviced by vehicles of type $\mathcal{T}_1$ and $\mathcal{T}_2$, while departures in $\{d_4, d_5\}$ are restricted to types $\mathcal{T}_3$ and $\mathcal{T}_4$. Finally, we assume that the departing times are such that Inequalities $(\mathbf{TI})(ix)$ are trivially satisfied, i.e., for any pair of departures d_i and d_j, $|d_i - d_j| \geq \Delta$.

For each (a_i, d_j, t_k) in $\mathcal{A} \times \mathcal{D} \times \mathcal{T}$, we assign values to x_{a_i,s,d_j} and to y_{a_i,s,t_k} according to the scheme presented in Figures 1 and 2, i.e., x_{a_i,s,d_j} is equal to the value that appears in the arc with tail a_i and head d_j, and y_{a_i,s,t_k} is equal to the value that appear in arc with tail $\mathcal{T}_k$ and head a_i. If an arc is not represented, its respective value is zero.

Constraints $(\mathbf{TI})(i) - (\mathbf{TI})(v)$ are clearly satisfied. For triples in $\mathcal{A} \times \mathcal{S} \times \mathcal{T}$,

$$y_{a,s,t} + \sum_{\substack{d\\t \notin \mathcal{T}(d)}} x_{a,s,d} \leq 0.5 + 0.4 < 1$$

if t represents $\mathcal{T}_1$ or $\mathcal{T}_2$, and

$$y_{a,s,t} + \sum_{\substack{d\\t \notin \mathcal{T}(d)}} x_{a,s,d} \leq 0.2 + 0.6 < 1$$

if t represents $\mathcal{T}_3$ or $\mathcal{T}_4$. Consequently, $(\mathbf{TI})(vi)$ is satisfied.

For $(\mathbf{TI})(vii)$, we have

$$x_{a,s,d} + \sum_{\substack{t\\t \notin \mathcal{T}(d)}} y_{a,s,t} \leq 0.2 + 0.7 < 1,$$

so $(\mathbf{TI})(vii)$ is also satisfied. Therefore, (x, y) belongs to $P_{LP}(\mathbf{TI})$.

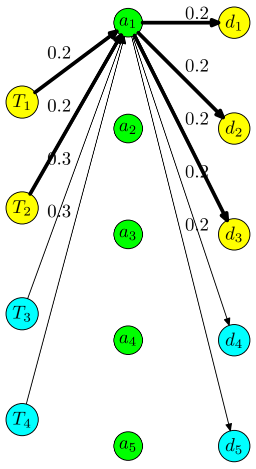


Figure 1: Incoherent assignments involving arrival a_1

Figure 1 shows a synchronization problem involving arrival a_1. If we look at the assignments as a flow, we have only 0.4 units coming from $\mathcal{T}_1$ and $\mathcal{T}_2$, while 0.6 units are assigned to d_1, d_2, and d_3. Assignments with these characterizations can not be represented by any element of $P_{LP}(\mathbf{QI})$, so we conclude that there are elements in $P_{LP}(\mathbf{TI})$ describing solutions which can not be represented by any element of $P_{LP}(\mathbf{QI})$. $\square$

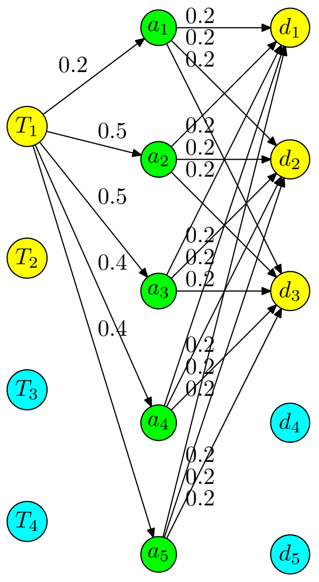
The previous results show that the space of feasible solutions of $P_{LP}(\mathbf{QI})$ is more compact than the space of $P_{LP}(\mathbf{TI})$. A similar analysis shows that $P_{LP}(\mathbf{QI})$ is also stronger than $P_{LP}(\mathbf{TI2})$.

Proposition 6.7. *For every solution (x, r) in P_{LP}(**QI**), there is a solution (x^t, z, r) in P_{LP}(**TI2**).*

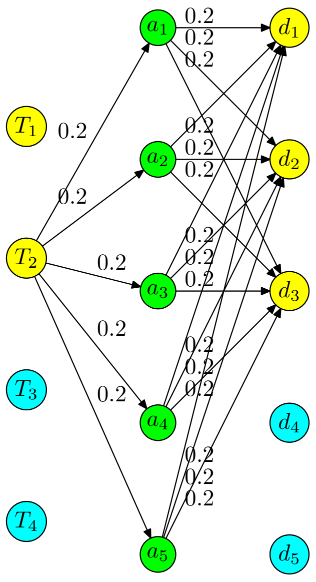
Proof. Let (x, r) be an element of $P_{LP}(\mathbf{QI})$. We define a pair (x^t, z) as follows:

$$
\begin{aligned}
x^t_{a,s,d} &= \sum_t x_{a,s,d,t}, \\
z_{a,t,d} &= \sum_s x_{a,s,d,t}.
\end{aligned}
$$

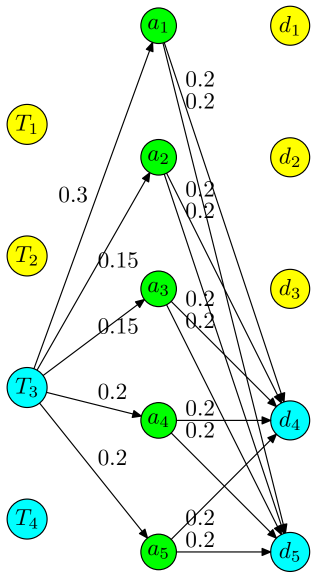
For the constraints of (**TI2**), we have the following:



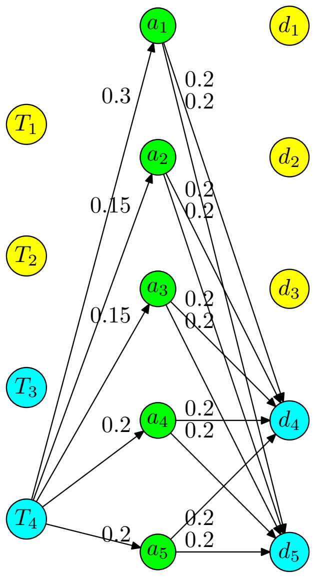
(a) Assignments of type $\mathcal{T}_1$



(b) Assignments of type $\mathcal{T}_2$



(c) Assignments of type $\mathcal{T}_3$



(d) Assignments of type $\mathcal{T}_4$

Figure 2: Complete Assignment

$$\begin{array}{lllll}
(i) & \sum_{(s,d)} x^t_{a,s,d} = \sum_{(s,d,t)} x_{a,s,d,t} = 1 & & a \in \mathcal{A} \\
(ii) & \sum_{(a,s)} x^t_{a,s,d} = \sum_{(a,s,t)} x_{a,s,d,t} = 1 & & d \in \mathcal{D} \\
(iii) & \sum_{(t,d)} z_{a,t,d} = \sum_{(d,s,t)} x_{a,s,d,t} = 1 & & a \in \mathcal{A} \\
(iv) & \sum_{(a,t)} z_{a,t,d} = \sum_{(a,s,t)} x_{a,s,d,t} = 1 & & d \in \mathcal{D} \\
(v) & \sum_{(a,d)} z_{a,t,d} = \sum_{(a,s,d)} x_{a,s,d,t} = t_t & & \mathcal{T}_t \in \mathcal{T} \\
(vi) & \sum_{(a,d)} x^t_{a,s,d} = \sum_{(a,d,t)} x_{a,s,d,t} = \beta & & s \in \mathcal{S} \\
(vii) & \sum_{s} x^t_{a,s,d} = \sum_{(s,t)} x_{a,s,d,t} = \sum_{t} z_{a,t,d} & & (a,d) \in \mathcal{A} \times \mathcal{D} \\
(ix) & \sum_{\substack{(a',d') \\ |d'-d|<\Delta}} x^t_{a',s,d'} = \sum_{\substack{(t,a',d') \\ |d'-d|<\Delta}} x_{a',s,d',t} \le 1 & & (s,d) \in \mathcal{S} \times \mathcal{D}
\end{array}$$

Finally, for all $(a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$, we have

$$\sum_{a'<a} x^t_{a',s,d} + \sum_{d'<d} x^t_{a,s,d'} \quad = \sum_{\substack{t \\ a'<a}} x_{a',s,d,t} + \sum_{\substack{t \\ d'<d}} x_{a,s,d',t} \quad \le 1 + r_{a,s,d},$$

which shows that (x^t, z, r) belongs to $P_{LP}(\mathbf{TI2})$. □

Theorem 6.8. *There are elements (x,z,r) of P_{LP}(**TI2**) associated with solutions of* VPP$_+$ *which are not associated with elements of P_{LP}(**QI**).*

Proof. We can adapt the example used to prove Theorem 6.6 in order to show this claim. For each a in $\mathcal{A}$ and d in $\mathcal{D}$, $\mathcal{T}(a) = \mathcal{T}(d) = \mathcal{T}$, with $t_1 = 2$ and $t_i = 1$, $2 \le i \le 4$. We assume that $m = 5$ (and, consequently, $\beta = 1$), that tracks in $\{s_1, s_2, s_3\}$ can only be assigned to vehicles of type $\mathcal{T}_1$ and $\mathcal{T}_2$, and that tracks in $\{s_4, s_5\}$ can only be assigned to vehicles of type $\mathcal{T}_3$ and $\mathcal{T}_4$. Finally, departing times are such that the Δ restrictions are trivially satisfied, i.e., for any pair of departures d_i and d_j, $|d_i - d_j| \ge \Delta$.

We can define x and z according to the solution illustrated in Figures 1 and 2 by assuming that the departures are interpreted as the tracks and that the arrivals are interpreted as units (a,d), i.e., (x,z) describe an integer assignment of arrivals to departures. As a consequence, we have that

$$\sum_{s} x_{a,s,d} = \sum_{t} z_{a,t,d} = 1, \tag{1}$$

which shows that Equalities (**TI2**)(vii) are satisfied by (x, y). Clearly, as in Theorem 6.6, this is also the case for Constraints (**TI2**)(i) – (**TI2**)(vi). Finally, as $\beta = 1$, we can take a vector r containing only zeros in order to compose an element (x, z, r) of P_{LP}(**TI2**), which is incoherent and can not be represented by any element of P_{LP}(**QI**). □

Models (**QI**), (**TI**), and (**TI2**) compute the number of crossings similarly to (**LU**). We saw in Proposition 2.6 that the linear relaxation of (**LU**) is always equal to zero. Using a family of instances similar to the one employed in this proposition, we can show that weaker results hold for (**QI**), (**TI**), and (**TI2**).

Proposition 6.9. *If the Δ restrictions are trivially satisfied (or ignored) and $m \geq 2$, then V_{LP}(**QI**) $= 0$.*

Proof. Let M be any feasible matching of $\mathcal{A}$ to $\mathcal{D}$ and to $\mathcal{T}$. Let $x_{a,s,d,t} = \frac{1}{m}$ if $(a, d, t) \in M$, and $x_{a,s,d,t} = 0$ otherwise. For clarity reasons, we denote the i-th element of M by (a^i, d^i, t^i), where the order is defined according to the arrival sequence (i.e., $a^i = a_i$, $1 \leq i \leq n$).

The following expressions show that the matching constraints of (**QI**) are satisfied by x:

$$\begin{array}{llll}
(i) & \sum\limits_{(s,d,t)} x_{a,s,d,t} = \sum\limits_{s,i} x_{a^i,s,d^i,t^i} = m\frac{1}{m} = 1 & a \in \mathcal{A}; \\
(ii) & \sum\limits_{(a,s,t)} x_{a,s,d,t} = \sum\limits_{s,i} x_{a^i,s,d^i,t^i} = m\frac{1}{m} = 1 & d \in \mathcal{D}; \\
(iii) & \sum\limits_{(a,s,d)} x_{a,s,d,t} = \sum\limits_{\substack{(s,i)\\ t^i = \mathcal{T}_t}} x_{a^i,s,d^i,t^i} = t_t m\frac{1}{m} = t_t & \mathcal{T}_t \in \mathcal{T}.
\end{array}$$

As $\sum\limits_{(a,s,d,t)} x_{a,s,d,t} = m\beta$, it is clear that

$$\sum_{(a,d,t)} x_{a,s,d,t} = \frac{1}{m} \sum_i x_{a^i,s,d^i,t^i} = \beta \quad s \in \mathcal{S}.$$

Finally, regarding the crossing constraints, we have

$$\sum_{a' < a} x_{a',s,d,t} + \sum_{d' < d} x_{a,s,d',t} = \frac{2}{m} \leq 1 \quad (a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D},$$

which shows that (**QI**)(v) is satisfied by the solution (x, r) with $r_{a,s,d} = 0$, $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$, if $m \geq 2$. Therefore, we conclude that $(x, 0)$ is in $P_{LP}(\mathbf{QI})$ and that $V_{LP}(\mathbf{QI}) = 0$ if $m \geq 2$. $\square$

Corollary 6.10. *If the Δ restrictions are trivially satisfied (or ignored) and $m \geq 2$, then $V_{LP}(\boldsymbol{TI}) = V_{LP}(\boldsymbol{TI2}) = 0$.*

Proof. It follows from Proposition 6.9 and from the fact that every element in $P_{LP}(\mathbf{QI})$ can be represented as an element of $P_{LP}(\mathbf{TI})$ and of $P_{LP}(\mathbf{TI2})$ with the same objective value. $\square$

The following proposition shows that the relaxed versions of the three models can yield nontrivial lower bounds for a certain family of instances of the VPP$_+$.

Proposition 6.11. *There are instances of* VPP$_+$ *for which the linear relaxations of (**QI**), (**TI**), and (**TI2**) yield nontrivial lower bounds.*

Proof. Let $\mathcal{I}$ be the family of instances of VPP$_+$ such that $t = n$ (and, therefore, $t_i = 1$ for every type $\mathcal{T}_i$), $|\mathcal{T}(d_i)| = 1$ for every departure $d_i \in \mathcal{D}$, $|\mathcal{T}(a_i)| = 1$ for every arrival $a_i \in \mathcal{A}$, $m = 2$, $\mathcal{A}$ is the inverse of $\mathcal{D}$ (i.e., there is a crossing involving each pair of elements of $\mathcal{A} \times \mathcal{D}$), and each trip d_i can not be assigned to the same track as trips d_{i-1} (when it exists) and d_{i+1} (when it exists) due to the Δ restrictions.

Suppose that the linear relaxation of any of the models produces a solution x in $P_{LP}(\mathbf{QI})$ whose objective value is equal to zero. Let $x'_{d,s}$ be the sum of the assignments involving d to s. In this case, $x'_{d_i,s_1} + x'_{d_{i+2},s_1} \leq 1$ and $x'_{d_i,s_2} + x'_{d_{i+2},s_2} \leq 1$. As $x'_{d_i,s_1} + x'_{d_{i+2},s_1} + x'_{d_i,s_2} + x'_{d_{i+2},s_2} = 2$, we have that $x'_{d_i,s_1} + x'_{d_{i+2},s_1} = 1$ and $x'_{d_i,s_2} + x'_{d_{i+2},s_2} = 1$. However, due to the Δ restrictions, we have that $x'_{d_i,s_1} + x'_{d_{i+1},s_1} + x'_{d_{i+2},s_1} \leq 1 \rightarrow x'_{d_{i+1},s_1} = 0$ and that $x'_{d_i,s_2} + x'_{d_{i+1},s_2} + x'_{d_{i+2},s_2} \leq 1 \rightarrow x'_{d_{i+1},s_2} = 0$, which shows that x is not a valid solution. A similar argument shows that the same fact holds for $P_{LP}(\mathbf{TI})$ and $P_{LP}(\mathbf{TI2})$. $\square$

We conclude this section showing that (**TI2**) becomes stronger if $\mathcal{T}(s) = \mathcal{T}$ for each track s in $\mathcal{S}$.

Theorem 6.12. *If $\mathcal{T}(s) = \mathcal{T}$ for each track in the depot, then there is an element (x^q, r) in $P_{LP}(\boldsymbol{QI})$ for every (x^t, z, r) in $P_{LP}(\boldsymbol{TI2})$.*

Proof. Let (x^t, z, r) be an element of $P_{LP}(\mathbf{TI2})$. Let $x^t_{a,s,d}$ denote the demand of consumer (a, s, d) that has to be supplied with resources coming from suppliers $z_{a,t,d}$, where $\mathcal{T}_t \in \mathcal{T}(a) \cap \mathcal{T}(d)$. Using this interpretation, we obtain a description of a vector x^q such that the amount of commodities sent from $z_{a,t,d}$ to $x^t_{a,s,d}$ is described by $x^q_{a,s,d,t}$.

Vector x^q can be greedily computed as follows. Given a consumer $x^t_{a,s,d}$, we assign resources from some supplier $z_{a,t,d}$ to it until the demand of the consumer is satisfied or until the resources of the supplier are exhausted. After this, we proceed to the next $x^t_{a,s',d}$ and/or to the next $z_{a,t',d}$, and this operation is repeated until all the assignments have been made. The existence of such a distribution follows from the fact that (x^t, z) satisfies Equation (**TI2**)(vii).

We claim that (x^q, r) belongs to P_{LP}(**QI**). The sum of the supplies involving each a and each d is clearly equal to one, so (**QI**)(i) and (**QI**)(ii) are satisfied. There is no "oversupply", so the accumulation of resources received by consumers involving each track s sums up to β, while the amount of supplies coming from all the suppliers of type $\mathcal{T}_i$ sums up to t_i. With this, we have Equations (**QI**)(iii) and (**QI**)(iv) satisfied. Finally, the validity of Equations (**QI**)(v) and (**QI**)(vi) for (x^q, r) follows from the fact that each pair (a, d) is assigned to a track s with the same value in (x^t, z) and in x^q. We conclude that (x^q, r) belongs to P_{LP}(**QI**). □

Finally, we remark that it is also possible to readapt formulation (**TI**) if we substitute variables $y_{a,s,t}$ for variables $y_{d,s,t}$, which represents the assignment of departure d to track s and to vehicle type $\mathcal{T}_t$. The resulting formulation (**TI**') has the same theoretical and computational properties as (**TI**).

6.3.4 Computational Results

Here we compare Models (**QI**), (**TI**), and (**TI2**) from a computational point of view. Table 11 gives the results obtained from the computation of 7 instances in a PC with an Intel Core2 Quad 2.83GHz processor and 16GB RAM using solver `CPLEX` 12.2. The first column in this table gives the name n-m-t of the problem. Following the notation we have been using in this text, n is the number of vehicle, m is the number of tracks, and t is the number of vehicle types. Given these parameters, an arrival sequence $\mathcal{A}$ containing vehicles with only one allowed type is randomly built, and using Algorithm 1 we generate a sequence $\mathcal{D}$ with the same property. Finally, the first $\lfloor t_1/\beta \rfloor$ tracks will allow only vehicles of type $\mathcal{T}_1$. If $t_1 \mod \beta \neq 0$,

	(QI)					**(TI)**					**(TI2)**				
Name	Row	Col	NZ	Sol	T	Row	Col	NZ	Sol	T	Row	Col	NZ	Sol	T
30-5-4	4569	10420	200132	0	3	8548	8449	128116	0	2	5457	9780	131587	0	5
49-7-15	16927	65390	2493457	0	331	34929	34760	923713	0	159	19407	43476	893493	30	*
56-7-4	22075	43276	1314161	0	123	37749	37570	921985	0	114	25037	41922	939333	0	7951
63-9-8	35864	101384	4374641	0	647	69026	68820	2146676	0	726	39861	78295	2146675	32	*
77-11-20	65404	292609	18344203	6	*	135156	134894	5545809	0	7073	71483	162968	5333351	110	10442
77-11-7	65391	162654	7845515	0	1677	122709	122460	4539523	0	1723	71268	135164	4552909	121	*
81-9-5	59225	124277	5481373	0	1788	103221	102964	3624219	3	*	65269	112481	3633545	194	*

Table 11: Comparing models (**QI**), (**TI**), and (**TI2**)

track $s_{\lfloor t_1/\beta \rfloor+1}$ will allow (at least) types $\mathcal{T}_1$ and $\mathcal{T}_2$. Then we repeat this step for the t_2 units of $\mathcal{T}_2$, starting from track $s_{\lfloor t_1/\beta \rfloor+1}$, and we follow this procedure until all the units have been distributed in the depot. At this point, it is clear that the instance already admits a feasible solution. We complete the description of sets $\mathcal{T}(s)$, $\mathcal{T}(d)$, and $\mathcal{T}(a)$ by adding the other types (each type is added with probability 0.5). The columns labeled Row, Col, NZ, Sol, and T give the number of constraints, the number of variables, the number of non-zeros, the solution, and the running time in seconds of the respective model. The solutions presented were computed within 10800 seconds. Programs that exceeded this limit were interrupted and have $*$ substituting their running times.

Formulation (**TI**) performs better with smaller instances, but Model (**QI**) has a better performance when we consider instances which are more similar to real-world scenarios (i.e., with more vehicles and not so many different types of vehicles). Finally, Model (**TI2**) has a very poor performance.

6.4 Solution Methods for the VPP$_+^P$

In this section, we present an exact integrated ILP formulation and propose an exact and a heuristic solution approach for the VPP$_+^P$.

6.4.1 An Exact Model for VPP$_+^P$

We present an integer programming model for the VPP$_+^P$ based on (**QI**):

$$
\begin{array}{llll}
(\mathbf{QP}) & & \min \sum_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & & \sum_{(s,d,t)} x_{a,s,d,t} = 1 & a \in \mathcal{A} \\
\text{(ii)} & & \sum_{(a,s,t)} x_{a,s,d,t} = 1 & d \in \mathcal{D} \\
\text{(iii)} & & \sum_{(a,s,d)} x_{a,s,d,t} = t_t & \mathcal{T}_t \in \mathcal{T} \\
\text{(iv)} & & \sum_{(a,d,t)} x_{a,s,d,t} = \beta & s \in \mathcal{S} \\
\text{(v)} & \sum_{a'<a} x_{a',s,d,t} + & \sum_{d'<d} x_{a,s,d',t} \leq 1 + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
\text{(vi)} & & \sum_{\substack{(a,d',t) \\ |d'-d|<\Delta}} x_{a,s,d',t} \leq 1 & (s,d) \in \mathcal{S} \times \mathcal{D} \\
\text{(vii)} & & \sum_{a,s} x_{a,s,d,t} = \sum_{d',s} x_{sync(d),s,d',t} & (d,t) \in \mathcal{D} \times \mathcal{T} \\
& & x_{a,s,d,t} \in \{0,1\} & {\scriptstyle (a,s,d,t) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}\times\mathcal{T} \atop t \in \mathcal{T}(a)\cap\mathcal{T}(s)\cap\mathcal{T}(d)} \\
& & r_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}
$$

The description of model (**QP**) is similar to the one provided for (**QI**). The only difference is the additional set of equations (**QP**)(vii), which asserts that each arrival $sync(d)$ is assigned to the same vehicle type as departure d.

Corollary 6.13. *The* VPP$^P_+$ *is* $\mathcal{NP}$*-complete.*

Proof. The same technique applied in the proof of Corollary 6.1 can be applied in order to reduce the VPP^P to the the VPP$^P_+$. Therefore, as the VPP^P is $\mathcal{NP}$-complete (see Proposition 2.32), we conclude that the VPP$^P_+$ is also $\mathcal{NP}$-complete. □

Similarly to the VPP^P, special attention must be dedicated to the weekend periods if $e < n$. The active vehicles are parked in $b = e/\beta$ tracks of the depot, and the inactive ones stay in the remaining $m-b$ tracks (these tracks will always accept every type of vehicle). The $n-e$ vehicles that stay inactive during the weekend periods will service trips of the first weekday period, so the positions of these elements must be bookkept. In the previous chapters, we dealt with this issue by computing the assignments of elements of the first and of the last weekday periods simultaneously, but here we adopt a different strategy. Namely, we introduce dummy arrivals a_i^h, $p+1 \leq h \leq p+w+1$, $e+1 \leq i \leq n$, and dummy departures d_i^h, $p \leq h \leq p+w$, $e+1 \leq i \leq n$, such that $d_i^h = a_i^{h+1}$, $p+1 \leq h \leq p+w$, $e+1 \leq i \leq n$, that indicate parking during the weekend. Periods compose a cyclic order, so period $p+w+1$ represents the first weekday period of the next cycle (week). The insertion

n	t	e	m	b	p	w	Time
4	4	2	2	1	10	2	1
9	9	6	3	2	10	2	323
16	16	12	4	3	10	2	193755
16	16	12	4	3	11	4	2986
25	25	20	5	4	11	4	311791
25	25	20	5	4	10	2	-

Table 12: Solving the VPP$_+^P$ using (**QP**)

of auxiliary artificial elements was also employed by Winter (1998) [91].

Table 12 presents the results of computational evaluations of (**QP**) with randomly generated instances that admit crossing-free solutions. Columns n, t, e, m, b, p, and w describe the parameters of the examples according to the notation we are using in this text, and Time denotes the running time in seconds of the computations, which were performed in a Sun Galaxy 4600 M2 with AMD Opteron 2,6 GHz with 16 kernels and 256 GB of RAM running CPLEX 11.0 under Linux SuSe 9.

As we can see, the programs generated from model (**QP**) are computationally very challenging. For the last instance, which contains only 25 vehicles and 12 periods, it was not possible to obtain a feasible solution after weeks of computation. The results show that this approach is not suitable for practical applications, so we investigate alternative methodologies in the following sections.

6.4.2 Sequential Approach

In the sequential approach, each period is solved separately without taking into account information and details about the whole problem, i.e., instances are decomposed and each part is solved almost independently from the others. Algorithm 9 describes this solution method.

Periods are sequentially enumerated in this procedure and the counting starts from the second weekend period. $T(a)$ and $T(d)$ denote the vehicle type assigned to a and d, respectively, and $sync^{-1}(a)$ is the inverse function of $sync$, i.e., $sync^{-1}(a) = d$ if and only if $sync(a) = d$.

Basically, Algorithm 9 solves each period h taking into account only the assignments that were previously made, thus avoiding coherence problems between matchings. In the first step, there is no fixed assignment, so the

Algorithm 9 Sequential Approach for VPP$_+^P$

```
Solve period 0
for $k := 1$ to $p + w - 2$ do
  for $d \in \mathcal{D}_{k-1}$ do
    $T(sync(d)) := T(d)$
  end for
  Solve period $k$
end for
for $d \in \mathcal{D}_{p+w-2}$ do
  $T(sync(d)) := T(d)$
end for
for $a \in \mathcal{A}_0$ do
  $d := sync^{-1}(a)$
  $T(d) := T(a)$
end for
Solve period $p + w - 1$
```

second weekend period is solved as an independent instance of the VPP$_+$. Conversely, because the periods compose a cyclic sequence, period $p+w-1$ is computed when the assignments $\mathcal{A}_{p+w-1} \to \mathcal{T}$ and $\mathcal{D}_{p+w-1} \to \mathcal{T}$ are already fixed. Any other period h, $1 \leq h \leq p + w - 2$, has its assignment $\mathcal{A}_h \to \mathcal{T}$ determined according to $\mathcal{D}_{h-1} \to \mathcal{T}$. Finally, the parking of inactive vehicles during weekend periods is modeled with dummy arrivals and departures.

6.4.2.1 Weekend Periods and Infeasibility

The sequential approach described in Algorithm 9 takes “local decisions”, as sets $\mathcal{T}(d)$ for d in $\mathcal{D}_{h'}$, $h' \neq h - 1$, are not taken into account when period h is solved. It is clear that such a myopic strategy may not be able to deliver optimal solutions, and the following proposition shows that the consequences can be even more drastic for instances with more than two weekend periods.

Proposition 6.14. *If $w > 2$, the sequential approach can produce infeasible solutions for the* VPP$_+^P$.

Proof. Let I be an instance of the VPP$_+^P$ such that $w > 2$; $e = n/2$; $t_1 = t_2 = e$; $\mathcal{T}(d) = \mathcal{T}$ if d belongs to the first two weekend periods; and $\mathcal{T}(d) = \{\mathcal{T}_1\}$ if d belongs to the remaining weekend periods.

Let p_{p+w-1}, p_0, and p_1 be periods of I, with p_0 representing the first weekend period. The assignments $\mathcal{D}_{p+w-1} \to \mathcal{T}$, $\mathcal{A}_0 \to \mathcal{T}$, $\mathcal{D}_0 \to \mathcal{T}$, and $\mathcal{A}_1 \to \mathcal{T}$ are determined in the first step of Algorithm 9. Therefore, the number of vehicles of each type that will stay active during the weekend periods is also fixed.

If Algorithm 9 chooses any element of type $\mathcal{T}_2$ to service trips from period p_0, it will be impossible to assign vehicles to every trip of the last $w-2$ weekend periods. Therefore, the sequential approach may not be able to produce a feasible solution for I. □

This infeasibility issue does not happen when the algorithm chooses a set of active vehicles containing t_i^w elements of type $\mathcal{T}_i$, $1 \le i \le t$, such that $\sum_{i=1}^{t} t_i^w = e$ and which is able to service all the trips of all the weekend periods. We refer to the identification of such a set as the Weekend Trips Cover Problem (**WTCP**). The instance of the **WTCP** associated with a certain scenario of the VPP$_+^P$ is defined by the description of all the departures of its weekend periods, i.e., their arrival times and their sets $\mathcal{T}(d)$; set $\mathcal{T}(s)$ for each of the b tracks used by active vehicles; values t_i, $1 \le i \le t$. The following proposition shows that the **WTCP** is hard from the theoretical point of view.

Proposition 6.15. *The **WTCP** is $\mathcal{NP}$-complete.*

Proof. We reduce the **3DM** to the **WTCP** in order to prove our claim. Let I be an instance of the **3DM** with a set of triples $T \in X \times Y \times Z$ such that $|X| = |Y| = |Z|$. We transform I in an instance I' of **WTCP** as follows. Sets X, Y, and Z describe the trips of three weekend periods, and therefore $e = |X| = |Y| = |Z|$. The order of the elements of X, Y, and Z can be ignored, and we assume that the Δ restrictions are trivially satisfied. For each triple $\{x, y, z\}$ in T, there is a type of vehicle $\mathcal{T}_i$, with $t_i = 1$, which can only be assigned to x in X, to y in Y, and to z in Z. Because $T = O(n)$, this reduction can be made in polynomial time.

It is clear that we have a set of e triples in T covering each element of X, Y, and Z from instance I if and only if there is a set of vehicles that can service all the weekend trips of instance I'. As a consequence, we conclude that the **WTCP** is $\mathcal{NP}$-complete. □

The following model is an ILP formulation that decides the **WTCP**.

$$
\begin{array}{llll}
(\mathbf{FT}) & & & \\
\text{(i)} & \displaystyle\sum_{(t,s)} x_{d,t,s} = 1 & & d \in \mathcal{D} \\
\text{(ii)} & \displaystyle\sum_{(d,s)\in\mathcal{D}_i\times\mathcal{S}} x_{d,t,s} \le t_t & & \mathcal{T}_t \in \mathcal{T}, p \le i \le p+w-1 \\
\text{(iii)} & \displaystyle\sum_{(d,s)\in\mathcal{D}_i\times\mathcal{S}} x_{d,t,s} = & \displaystyle\sum_{(d,s)\in\mathcal{D}_{i+1}\times\mathcal{S}} x_{d,t,s} & \mathcal{T}_t \in \mathcal{T}, p \le i < p+w-1 \\
\text{(iv)} & \displaystyle\sum_{(d,t)\in\mathcal{D}_p\times\mathcal{T}} x_{d,t,s} = \beta & & s \in \mathcal{S} \\
\text{(v)} & \displaystyle\sum_{\substack{(d',t)\\ |d-d'|<\Delta}} x_{d,t,s} \le 1 & & (s,d) \in \mathcal{S}\times\mathcal{D} \\
 & x_{d,t,s} \in \{0,1\} & & \substack{(d,t,s)\in\mathcal{D}\times\mathcal{T}\times\mathcal{S}\\ t\in\mathcal{T}(d)\cap\mathcal{T}(s)} .
\end{array}
$$

Model (**FT**) uses binary variables $x_{d,t,s}$ to represent the assignment of vehicle type $\mathcal{T}_t$ to departure d and to tracks s; it belongs to the model if and only if $\mathcal{T}_t \in \mathcal{T}(d)$ and $\mathcal{T}_t \in \mathcal{T}(s)$. Equalities (**FT**)$(i)$ guarantee that each departing trip is assigned to exactly one type and one track. Inequalities (**FT**)(ii) assert that $t_i^w \le t_i$, $1 \le i \le t$. Inequalities (**FT**)(iii) certify that the same number t_i^w of elements of type $\mathcal{T}_i$, $1 \le i \le t$, is used in each weekend period. Finally, (**FT**)(iv) guarantees that the chosen elements can be parked in the b tracks available on the weekend periods, and (**FT**)(v) formulates the Δ restrictions.

If x is a solution produced by (**FT**), values of t_i^w, $1 \le i \le t$, can be obtained in the following way:

$$
t_i^w = \sum_{(d,s)} x_{d,i,s} \qquad 1 \le i \le t.
$$

It is possible to adjust Algorithm 9 in order to fix the number of vehicles of each type according to a solution $\{t_1^w, t_2,^w, \ldots, t_t^w\}$ obtained for the respective **WTCP**. With this adjustment, the sequential approach is always able to deliver a feasible solution.

6.4.2.2 Integrated Periods

In the decomposition method applied to the VPP^P, the first and the last weekday periods are solved in an integrated way. We investigate now the applicability of this technique in the context of the VPP$^P_+$. Assuming that periods 1 and p are the first and the last weekday periods, respectively, we propose the following model to solve this integrated scenario:

$$
\begin{array}{lrll}
(\mathbf{FL}) & \min \displaystyle\sum_{(a,s,d)} r_{a,s,d} & & \\
\text{(i)} & \displaystyle\sum_{(s,d,t)} x_{a,s,d,t} = 1 & & a \in \mathcal{A} \\
\text{(ii)} & \displaystyle\sum_{(a,s,t)} x_{a,s,d,t} = 1 & & d \in \mathcal{D} \\
\text{(iii)} & \displaystyle\sum_{(a,s,d)\in\mathcal{A}\times\mathcal{S}\times\mathcal{D}_1} x_{a,s,d,t} = t_t & & \mathcal{T}_t \in \mathcal{T} \\
\text{(iv)} & \displaystyle\sum_{(a,s,d)\in\mathcal{A}_p\times\mathcal{S}\times\mathcal{D}} x_{a,s,d,t} = t_t & & \mathcal{T}_t \in \mathcal{T} \\
\text{(v)} & \displaystyle\sum_{(a,s,d)\in\mathcal{A}\times\mathcal{S}\times\mathcal{D}_p} x_{a,s,d,t} = t^w_t & & \mathcal{T}_t \in \mathcal{T} \\
\text{(vi)} & \displaystyle\sum_{(a,d,t)} x_{a,s,d,t} = \beta & & s \in \mathcal{S} \\
\text{(vii)} & \displaystyle\sum_{a'<a} x_{a',s,d,t} + \sum_{d'<d} x_{a,s,d',t} \le 1 + r_{a,s,d} & & (a,s,d) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D} \\
\text{(viii)} & \displaystyle\sum_{\substack{(a,d',t)\\ |d'-d|<\Delta}} x_{a,s,d',t} \le 1 & & (s,d) \in \mathcal{S}\times\mathcal{D} \\
& x_{a,s,d,t} \in \{0,1\} & & \substack{(a,s,d,t)\in\mathcal{A}\times\mathcal{S}\times\mathcal{D}\times\mathcal{T}\\ t\in\mathcal{T}(a)\cap\mathcal{T}(s)\cap\mathcal{T}(d)} \\
& r_{a,s,d} \in \{0,1\} & & (a,s,d) \in \mathcal{A}\times\mathcal{S}\times\mathcal{D}.
\end{array}
$$

Model (**FL**) is almost similar to Formulation (**QP**). The main difference lies in the criterion of inclusion of variables. Namely, variables $x_{a,s,d,t}$ and $r_{a,s,d}$ belong to (**FL**) if and only if:

- $a \in \mathcal{A}_p$, $s \in \mathcal{S}_p$, and $d \in \mathcal{D}_p$; or
- $a \in \mathcal{A}_p$, $s \in \mathcal{S}_1$, and $d \in \mathcal{D}_1$; or
- $a \in \mathcal{A}_1$, $s \in \mathcal{S}_1$, and $d \in \mathcal{D}_1$.

Besides, (**FL**) contains only variables $x_{a,s,d,t}$ such that $\mathcal{T}_t \in \mathcal{T}(a)$, $\mathcal{T}_t \in \mathcal{T}(s)$, and $\mathcal{T}_t \in \mathcal{T}(d)$. Equalities (**FL**)(iii), (**FL**)(iv), and (**FL**)(v) control the number of vehicles used in the weekday and in the weekend periods, respectively, with the last using a pre-computed vector $(t^w_1, t^w_2, \ldots, t^w_t)$ generated from a solution of the WTCP associated with the instance.

6.4.2.3 Performance of the Sequential Approach

We tested an implementation of the sequential approach described in Algorithm 9 using (**FT**) to fix the number of active elements of each type during

n	t	e	m	b	p	w	Time
20	3	16	5	4	4	2	3
30	5	20	6	4	5	2	10
36	5	24	6	4	6	2	651
48	6	36	8	6	6	2	680
64	7	40	8	5	6	2	665
72	7	56	9	7	6	2	7150

Table 13: Computational results of (**FL**)

the weekend periods and (**FL**) to solve the first weekend period and the first weekday period in an integrated way. Table 13 shows the results of or computational tests in a PC with an Intel Core2 Quad 2.83GHz processor and 8GB RAM using `CPLEX` 12.2.0.2. The descriptions of the table and of the scheme of generation of the instances are similar to the ones we presented in Section 6.4.3.

All the solutions obtained are crossing-free and the time consumption for small- to medium-scale scenarios can be considered satisfactory. However, there are two serious barriers for the application of the sequential approach in practical scenarios. The first is the time consumption for large-scale scenarios. As our results show, the largest instance of our database is very challenging. We tested the method with an instance containing 100 vehicles, and it was not possible to obtain a solution after a week of computation. The second issue is presented in a formal way in the following proposition.

Proposition 6.16. *The sequential approach can produce arbitrarily bad solutions for the* VPP$^P_+$.

Proof. We present a family of instances of the VPP$^P_+$ for which Algorithm 9 can produce arbitrarily bad solutions. Each instance of our family is such that $t = n$, $e = n$, $b = m$, $\mathcal{T}(s) = \mathcal{T}$ for each track s, and $p+w = 3k$ for some positive integer k. We enumerate the periods from 0 to $3k-1$. Periods h with $(h \mod 3) \leq 1$ are called *free periods* and are such that $\mathcal{T}(d) = \mathcal{T}$ for every d in $\mathcal{D}_h$. Periods h with $(h \mod 3) = 2$ are called *fixed periods* and are such that $|\mathcal{T}(d)| = 1$ for every d in $\mathcal{D}_h$, i.e., $\mathcal{D}_h$ can be interpreted as an ordering of $\mathcal{T}$. All the fixed periods have the same departure sequence $\mathcal{D}_h = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$. Finally, we assume that $sync(d^h_i) = a^{h+1}_i$, $0 \leq h \leq 3k-1$ and $1 \leq i \leq n$, i.e., all the trips have the same duration and their respective vehicles arrive in the depot on period $h+1$ in the same order they left on period h.

n	t	e	m	b	p	w	Crossings
4	4	2	2	1	10	2	6
9	9	6	3	2	10	2	10
16	16	12	4	3	10	2	67
25	25	20	5	4	10	2	113

Table 14: Solving the VPP$_+^P$ using the Sequential Assignment

Any feasible solution for the free periods is crossing-free, but bad choices regarding assignments $\mathcal{D}_h \to \mathcal{T}$, $(h \mod 3) = 1$, may lead to an arrival sequence $\mathcal{A}_{h+1}$ for which solutions with crossings are unavoidable. Namely, if $\mathcal{D}_{h+1} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$ and $\mathcal{A}_{h+1} = \{\mathcal{T}_n, \mathcal{T}_{n-1}, \dots, \mathcal{T}_1\}$, each pair of units is involved in a semi-crossing. As a result, any feasible plan will require $m\beta(\beta-1)/2$ crossings. Conversely, crossing-free solutions can be obtained if the i-th arrival is assigned to the i-th departure, $1 \le i \le n$. □

We created some instances of the family described in Proposition 6.16 and used the basic version of Algorithm 9 to solve them. The results are presented in Table 14, and they show that the sequential approach does produce poor solutions in these cases.

We conclude that the sequential approach may produce good results if the improvements suggested in this section are incorporated to the problem, but very poor solutions can also be generated, which makes the method unreliable.

6.4.3 The Progressive Method Applied to (QP)

Algorithm 10 contains a solution method for the (**QP**) with two levels of iterations based on the Progressive Approach. M represents the original (**QP**) program, $LB(M)$ is a lower bound for M, $\mathcal{T}(a)'$ and $\mathcal{T}(d)'$ are subsets of $\mathcal{T}(a)$ and $\mathcal{T}(d)$, respectively, and M' is the projection of M which is being solved in the current iteration.

In the first level (or in the extern iteration), at step $iter$, we choose a type $\mathcal{T}^{iter}(d) \in \mathcal{T}(d) \setminus \mathcal{T}(d)'$ to be added to $\mathcal{T}(d)'$ and to $\mathcal{T}(sync(a))'$ for each departure d. Arrivals a and departures d of period k such that $\mathcal{T}^{iter}(d) = \mathcal{T}^{iter}(a) = \mathcal{T}_j$ compose the sequences $\mathcal{A}_k^{j,iter}$ and $\mathcal{D}_k^{j,iter}$, respectively. We remark that $|\mathcal{T}(d)'| = min(iter, |\mathcal{T}(d)|)$ is an invariant of this algorithm. When $iter = 1$, $|\mathcal{T}(d)'| = 1$ for each trip d, and in order to assert the existence of a feasible solution, these sets are chosen in a way that each $\mathcal{T}_i$

Algorithm 10 Progressive Approach applied to (**QP**)

```
for $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ do
    add $r_{a,s,d}$ to $v(M')$
end for
for $a \in \mathcal{A}$ and $d \in \mathcal{D}$ do
    $\mathcal{T}(a)' := \emptyset$ and $\mathcal{T}(d)' := \emptyset$
end for
$r(M') := r(M)$
$iter := 0$
while $iter < t$ do
    $iter := iter + 1$
    for $d \in \mathcal{D}$ do
        $\mathcal{T}^{iter}(d) := \mathcal{T}_i \in \mathcal{T}(d) \setminus \mathcal{T}(d)'$
        $\mathcal{T}(d)' := \mathcal{T}(d)' \cup \mathcal{T}^{iter}(d)$
        $\mathcal{T}(sync(d))' := \mathcal{T}(sync(d))' \cup \mathcal{T}^{iter}(d)$
    end for
    $iter_2 := 0$
    while $iter_2 < n$ do
        for $(a, s, d, k) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \times \mathcal{T}$ such that $k \in \mathcal{T}(a) \cap \mathcal{T}(s) \cap \mathcal{T}(d)$ do
            if $\mathcal{T}^{iter}(a) = \mathcal{T}_k$ and $\mathcal{T}^{iter}(d) = \mathcal{T}_k$ and $d^{iter}(a, d) = iter_2$ then
                add $x_{a,s,d,k}$ to $v(M')$
            end if
        end for
        if $c(solve(M')) = LB(M)$ then
            stop algorithm
        end if
        $iter_2 := iter_2 + 1$
    end while
end while
```

belong to exactly t_i sets $\mathcal{T}(d)'$. When $iter > 1$, types $\mathcal{T}^{iter}(d)$ are chosen randomly, so $|\mathcal{A}_k^{j,iter}|$ may be different from $|\mathcal{D}_k^{j,iter}|$ for some values of $iter$.

In the second level (or in the inner iteration), we use the same idea of Algorithm 7. In each step of the inner loop, some variables of M are chosen to be incorporated to M', which is initially empty. For each iteration $iter$, we define the function $d^{iter} : \mathcal{A} \times \mathcal{D} \to \mathbb{N}$, which denotes the absolute difference between the position of a in $\mathcal{A}_k^{j,iter}$ and the position of d in $\mathcal{D}_k^{j,iter}$. If $a \notin \mathcal{A}_k^{j,iter}$ or $d \notin \mathcal{D}_k^{j,iter}$, then $d^{iter}(a,d) = \infty$. We add variable $x_{a,s,d,t}$ to M' in the current step if and only if $d^{iter}(a,d) = iter_2$ and $\mathcal{T}^{iter}(d) = \mathcal{T}^{iter}(a) = \mathcal{T}_t$.

Algorithm 10 is an exact method, and therefore delivers an optimal solution if executed until its termination. We applied it to a small-scale instance, containing 4 weekday periods, 2 weekend periods, 20 trips per weekday periods, 15 per weekend period, 4 tracks, and 5 vehicle types. After 160000 seconds, we had already a matching containing only one crossing, but the optimality of this solution could not be proved. Besides, the time needed by the algorithm to close optimality gaps increased drastically at each step. We interrupted the execution of the algorithm after noticing that the whole computation could take at least two months.

However, a solution for the same instance requiring three crossings was obtained after the optimization of the fourth reduced model, and this was accomplished in five seconds. For this reason, we decided to sacrifice the optimality and modify Algorithm 10 in order to obtain a heuristic procedure that computes feasible solutions quickly. The program is described in Algorithm 11.

The main differences between the two algorithms are concentrated on the criterion to stop the iterations of the second level. Because solutions for the problem typically describe matchings which are not very different from the sequential assignment, the inner iteration stops when $iter_2 = 3$. Moreover, if the optimal solution of the current M' has the same cost as the best matching computed so far, the inner iteration is interrupted and Algorithm 11 proceeds to the next step of the first level. The idea of this procedure is to reduce the number of iterations and also the size of the last programs it has to solve.

Tables 15 shows the results of computational experiments of Algorithm 11 with artificial instances of the VPP$^P_+$. Our tests were conducted on a 64-bits Intel(R) Core(TM)2 Quad with 2.83 GHz, 8 GB of RAM memory, running openSuse Linux 11.2. Our code is implemented in `C++` and was compiled

Algorithm 11 Heuristic for (**QP**)

```
$lastSolution := \infty$
**for** $(a, s, d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}$ **do**
  add $r_{a,s,d}$ to $v(M')$
**end for**
**for** $a \in \mathcal{A}$ and $d \in \mathcal{D}$ **do**
  $\mathcal{T}(a)' := \emptyset$ and $\mathcal{T}(d)' := \emptyset$
**end for**
$r(M') := r(M)$
$iter := 0$
**while** $iter < t$ **do**
  $iter := iter + 1$
  **for** $d \in \mathcal{D}$ **do**
    $\mathcal{T}^{iter}(d) := \mathcal{T}_i \in \mathcal{T}(d)$
    $\mathcal{T}(d)' := \mathcal{T}(d)' \cup \mathcal{T}^{iter}(d)$
    $\mathcal{T}(sync(d))' := \mathcal{T}(sync(d))' \cup \mathcal{T}^{iter}(d)$
  **end for**
  $iter_2 := 0$
  **while** $iter_2 < 3$ **do**
    **for** $(a, s, d, k) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \times \mathcal{T}$ such that $k \in \mathcal{T}(a) \cap \mathcal{T}(s) \cap \mathcal{T}(d)$ **do**
      **if** $\mathcal{T}^{iter}(a) = \mathcal{T}_k$ and $\mathcal{T}^{iter}(d) = \mathcal{T}_k$ and $d^{iter}(a, d) = iter_2$ **then**
        add $x_{a,s,d,k}$ to $v(M')$
      **end if**
    **end for**
    **if** $c(solve(M')) = LB(M)$ **then**
      stop algorithm
    **end if**
    **if** $solution = lastSolution$ **then**
      break
    **end if**
    $iter_2 := iter_2 + 1$
    $lastSolution := solution$
  **end while**
**end while**
```

n	t	e	m	b	p	w	Time	Sol
20	5	15	4	3	4	2	30	3
25	5	20	5	4	5	3	75	7
36	5	30	6	5	7	2	181	0
48	7	36	8	6	5	2	6683	4
64	7	40	8	5	10	2	10581	0
81	6	63	9	7	10	2	1213	0
100	7	70	10	7	10	4	4282	0
126	7	90	14	10	10	4	15954	0

Table 15: Solving the VPP$^P_+$ using Algorithm 11

using g++ 4.4.1 and with the callable library of CPLEX 12.1.0 ILOG (2010) [57]. Columns n, t, e, m, b, p, and w describe the parameters of the examples according to the notation we are using in this text. Given these parameters, an arrival sequence $\mathcal{A}$ containing vehicles with only one allowed vehicle type is randomly built, and departure sequence $\mathcal{D}$ is obtained with Algorithm 1. Finally, the first $\lfloor t_1/\beta \rfloor$ tracks will allow only vehicles of type $\mathcal{T}_1$. If t_1 mod $\beta \neq 0$, track $s_{\lfloor t_1/\beta \rfloor+1}$ will allow (at least) types $\mathcal{T}_1$ and $\mathcal{T}_2$. Then we repeat this step for the t_2 units of $\mathcal{T}_2$, starting from track $s_{\lfloor t_1/\beta \rfloor+1}$, and we follow this procedure until all the units have been distributed in the depot. At this point, it is clear that the current instance admits a feasible solution. We complete the description of sets $\mathcal{T}(s)$, $\mathcal{T}(d)$, and $\mathcal{T}(a)$ by adding the other types (each type is added with probability 0.5). Column Time describes the time consumption and column Sol the number of crossings of the solution obtained for the respective instance.

The results show that the VPP$^P_+$ is tractable for large-scale scenarios with Algorithm 11. It is also interesting to observe that some results for large-scale scenarios were better than results for medium-sized instances. It is clear that the dimension of the problem plays a very significative role, but these results suggest that the difficulty of the problem is strongly correlated to the ratio β/m.

Finally, we conclude that Algorithm 11 describes a satisfactory approach to the VPP$^P_+$. It solves the problem in an integrated way, thus avoiding the arbitrarily bad solutions that can be produced by the sequential approach. Moreover, its computational performance is satisfactory for practical applications.

Chapter 7

Uncertainty in the Vehicle Positioning Problem

In this chapter, we give an overview of uncertainty in different situations and investigate it in the context of the VPP. We introduce an online version of the VPP, present a generic competitive analysis of the problem, and suggest an online procedure to solve it. A criterion to evaluate robustness and a model able to produce matchings which are less sensitive to disruptions are introduced as well. Our computational experiments show that a software implementing the techniques presented here is suitable for real-world applications.

7.1 Introduction

One aspect that plays a significant role in many real-world scenarios of the VPP is uncertainty in the input data. Namely, uncontrollable events like extreme climatic conditions, traffic accidents, and erratic behavior of passengers and pedestrians can cause delays of vehicles. Therefore, the expected arrival order may suffer some modifications. Such changes are potentially harmful to originally good (and even to optimal) solutions, as unpredicted crossings may occur if the initial plan is maintained.

Our objective in this chapter is to characterize and to compute solutions which are not so sensitive to disruptions and to propose an online algorithm that can adjust a matching when modifications in the arrival order take place.

This chapter is organized as follows. In Section 7.2 we discuss how uncertainty appears in different contexts, giving special attention to combinatorial online optimization and to robust optimization. In Section 7.3 we investigate an online version of the VPP. Namely, we propose an online algorithm and present a generic competitive analysis of the problem. In Section 7.4 we suggest a criterion to evaluate the robustness of solutions and suggest an ILP formulation for the generation of robust plans. Finally, computational results are reported in Section 7.5.

7.2 The Different Forms of Uncertainty

Typically, it is assumed that the input data for a problem is precisely known in advance. However, in many cases, theoretical formulations that do not take uncertainty into account are too unrealistic, and therefore unsatisfactory for practical application. We can classify problems of this nature in three (non-mutually exclusive) groups.

The first group contains problems for which the acquisition of data is hard and/or imprecise. For instance, a work that analyzes the changes of temperatures in Antarctica during the pre-instrumental era (i.e., before 1900) is based on a series of measurements whose values are not 100% reliable due to the (im)precision of the materials that were used and due to human failures. Eventually, there is even absence of data for some periods. In this problem, untrustworthy values can be indirectly estimated with the help of ”proxies” like the water stable isotopes content of ice-cores retrieved in the investigated region (for more details, we refer to the article of Fernandoy et al. (2010) [34]). Scenarios like this can be found in every field of the so-called physical sciences, and each area has its own specific tools to deal with these obstacles. This group of problems is certainly important, but we will not consider it in this text.

The other groups contain online and robust problems. We investigate in this chapter how these aspects appear in the context of the VPP, so we introduce them in a more detailed way.

7.2.1 Online Combinatorial Optimization

In many contexts, the complete input data can not be determined a priori, as it is produced in real-time. A classical example is the ADAC Problem, which is about the support given by the German automobile club ADAC to its members when they have problems with their cars on the road. Basically, service requests come from unpredictable places in Germany at unpredictable moments, and dispatchers have to assign units to service them almost instantly (for more details, we refer to Grötschel, Hiller & Tuchscherer (2007) [49]). Problems of this nature require methods with the capacity to yield solutions in a very short period of time which are not only satisfactory, but also flexible enough to allow for possible modifications that might be necessary after the arrival of new input data.

The ADAC Problem is a typical example of an *Online Combinatorial Problem*. Comprehensive overviews of this topic are given by Borodin & El-Yaniv (1998) [17] and Fiat & Woeginger (1998) [35], and some recent surveys are presented by Winter & Zimmermann (1998) [92] and by Hiller (2010) [55]. The work of Ascheuer et al. (1998) [4] and the book of Grötschel, Krumke & Rambau (2001) [47] show practical applications of combinatorial online optimization, as do many recent Ph.D. Theses (Ascheuer (1995) [3]; Hiller (2010) [55]; Kamin (1998) [60]; Winter (1998) [91]).

There is a strong connection between online combinatorial optimization and combinatorial approximation algorithms. For example, the first approximation algorithm, developed by Graham (1966) [42], can be seem as an online algorithm, as it has to take decisions based on the current input without informations about future incoming data. For an overview of approximation algorithms, we refer to Vazirani (2003) [87].

The most important method used to evaluate online algorithms is the *competitive analysis*. This technique was initially proposed by Sleator & Tarjan (1985) [84] and analyzed in other works (see Karlin et al. (1988) [63] and Karlin (1998) [62]). In a competitive analysis, given an arbitrary instance of the problem, we want to calculate an upper bound for the ratio between the value of the solution produced by an online algorithm and the value of an optimal solution. This upper bound must be valid for every instance of the problem, so worst-case scenarios are used in the estimate. As a consequence, the superiority of algorithms which perform well in practice towards others which are typically unsatisfactory may not be theoretically certified by the competitive analysis. This phenomena happens with the Online Bin-Coloring Problem, for example (see Krumke et al. (2001) [71]).

The weakness of the competitive analysis is the main motivation for recent works proposing alternative evaluation methods of online algorithms. These new techniques try to obtain better results by considering weaker adversary algorithms, well-behaved inputs, etc. For example, a new method for probabilistic analysis of online algorithms using the concept of stochastic dominance was proposed recently by Hiller & Vredeveld (2008) [56].

7.2.2 Robust Optimization

For many problems, some of the parameters are not associated with fixed numbers, but with sets or intervals containing the values that they can assume. When probability distributions describing the behavior of the uncertain parameters are known, we have a *Stochastic Programming Problem*. If this is not the case, we have a *Robust Optimization Problem*.

The first works that investigated uncertainty in mathematical models introduce the fundamentals of Stochastic Programming. It is not easy to apply Stochastic Programming to real-world scenarios, as models for such problems are very challenging from a computational point of view and knowledge about the probability distributions followed by uncertain parameters is typically limited in practical situations. For further information, we refer to the Stochastic Programming Bibliography (van der Vlerk (1996-2007) [86]) and to the Stochastic Integer Programming Bibliography (van der Vlerk (1996-2007) [85]).

Similarly to Stochastic Programming, Robust Programming is also about the optimization of problems with uncertainty in the parameters, but in this case, the probability distributions of the parameters are unknown. The ultimate objective in Robust Programming is to find a satisfactory solution which is feasible not only for the expected input, but also for all its possible variations (or eventually only for the ones which are more likely to occur). Optimality is obviously desired, but it traditionally must be sacrificed in exchange for robustness.

Initially, Robust Programming was investigated in the context of convex constrained optimization (Ben-Tal & Nemirovski (2002) [7]). The methods proposed for problems of this nature can not be directly applied to discrete optimization scenarios, though, so many recent works have been dedicated to close this gap, as the article of Bertsimas & Sim (2003) [9], the monograph of Kouvelis & Yu (1997) [69], and the Ph.D. Thesis of Salazar-Neumann (2009) [80]. Especially interesting for the VPP is the concept of *limited recovery from infeasibilities*, proposed in some recent articles (Ben-Tal, Boyd & Nemirovski (2006) [8]; Liebchen et al. (2007) [73]). For a survey on Robust Optimization, we refer to the article of Beyer & Sendhoff (2007) [10].

7.3 Online Approach to the VPP

The first investigations involving online versions of the VPP were presented by Winter (1998) [91] and by Winter & Zimmermann (2000) [93]. In these works, the authors introduced the concept of (c, d)-competitiveness, which is reproduced below:

Definition 7.1. *An online algorithm ALG is said to be* (c,d)-competitive *if and only if the following inequalities hold for all problem instances I:*

$$cost_{ALG}(I) \leq \begin{cases} c.cost_{OPT}(I) & \text{if } cost_{OPT}(I) \neq 0; \\ d & \text{if } cost_{OPT}(I) = 0, \end{cases}$$

where $cost_{ALG}(I)$ denotes the cost of the solution produced by ALG for the instance I and $cost_{OPT}(I)$ denotes the optimal cost computed by an optimal algorithm OPT knowing the complete instance I in advance, i.e., OPT can be any optimal offline algorithm*.*

This generalization of the traditional definition of competitive ratio (see, e.g., Karlin (1998) [62]) is necessary because some instances of the VPP have optimal solutions whose cost is zero, and in this case $\frac{cost_{\mathrm{ALG}}(I)}{cost_{\mathrm{OPT}}(I)}$ is not defined. Based on this fact, Winter and Zimmermann proposed to partition the instances of the VPP in parts $\mathcal{I}_0$ and $\mathcal{I}_1$, where:

$$\mathcal{I}_0 = \{I \in \mathcal{I} : cost_{\mathrm{OPT}}(I) = 0\}; \quad \text{and}$$
$$\mathcal{I}_1 = \{I \in \mathcal{I} : cost_{\mathrm{OPT}}(I) \neq 0\}.$$

We say that there is no competitive algorithm for an online optimization problem P if there is no constant c and no online algorithm ALG such that

$$cost_{\mathrm{ALG}}(I) \leq c.cost_{\mathrm{OPT}}(I)$$

for every instance I of P. For the VPP, an algorithm ALG is said to be non-competitive if there is no pair (c, d) of constants such that algorithm ALG is (c, d)-competitive.

Winter (1998) [91] investigates two real-time versions of the problem. In both cases, the dispatcher controlling the assignments in the depot is sure only about which element a_{i+j} will be the next to arrive. Moreover, he knows that a_{i+j} belongs to the sequence $\mathcal{A}' = (a_{i+1}, a_{i+2}, \ldots, a_{i+\Delta})$ describing the Δ vehicles which are supposed to come next. In one case, it is assumed that $\mathcal{A}'$ should be substituted by $(a_{i+j}, a_i, a_{i+1}, \ldots, a_{i+j-1}, a_{i+j+1}, \ldots, a_{i+\Delta})$, i.e., a_{i+j} moves to the front of the sequence, while the other elements do not have their order changed. In the second case, the expected sequence is substituted by $(a_{i+j}, a_{i+j-1}, a_{i+j-2}, \ldots, a_i, a_{i+j+1} \ldots, a_{i+\Delta})$, i.e., the first $i+j$ arrivals of $\mathcal{A}'$ have their order inverted. An initial plan is elaborated based on the original expected data, and a new solution is computed whenever modifications in $\mathcal{A}$ are confirmed using the current plan as warm start. The time-limit for the operation is 5 minutes, which is a challenging constraint for real-world instances. Finally, Winter also examines some theoretical aspects of an online version of the VPP.

Demange, Di Stefano & Leroy-Beaulieu (2006) [28] analyze scenarios where all the assignments $\mathcal{A} \rightarrow \mathcal{D}$ are fixed and the objective is to determine the assignments $\mathcal{A} \rightarrow \mathcal{S}$ in a way that the number of tracks employed in the solution is minimized. For these cases, they show that the problem is similar to the online bounded coloring of a permutation graph if the last arrival occurs before the first departure and present an algorithm with the best possible competitive ratio.

Cornelsen & Di Stefano (2007) [24] assume that the assignments $\mathcal{A} \rightarrow \mathcal{D}$ are fixed and that the tracks have infinite size. In another variation, their objective is to obtain cyclic timetables, i.e., to compute plans that can be applied to all the periods. The authors propose algorithms for offline and online scenarios of these problems, also based on coloring algorithms for some specific families of graphs.

7.3.1 An Online Algorithm for the VPP

The online extension of the VPP investigated in this chapter is defined as follows:

Definition 7.2. *The* VPPO *(the online* VPP*) is the version of the* VPP *where the arrival sequence is given a priori, but it may undergo modifications. An algorithm ALG for the* VPPO *always has a current plan, which is based on the current expected sequence* $\mathcal{A} = \{a_i, a_{i+1}, \ldots, a_j, \ldots, a_n\}$*. If ALG is informed about the next incoming arrival and it happens to be* a_j *instead of* a_i*,* $\mathcal{A}$ *is substituted by* $\{a_j, a_i, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n\}$ *and the algorithm can modify the current solution by changing the assignments involving vehicles that have not been parked yet.*

Basically, VPPO is similar to the first real-time problem investigated in Winter (1998) [91]. However, differently from Winter, we do not assume that the next incoming vehicle is one of the next Δ expected arrivals for some fixed value of Δ.

We propose the following online algorithm for the VPPO.

Algorithm 12 REPLAN

```
$x := solve(\mathcal{A})$
for $i := 1$ to $n$ do
  $a :=$ next arriving vehicle
  if $a$ comes unexpectedly and a significant change takes place then
    $update(x)$
  end if
end for
```

An unexpected arrival causes *significant changes in* $\mathcal{A}$ if it leads to one of the following situations:

- if the current plan is maintained, an unexpected crossing will occur;
- if the current plan is modified, an expected crossing can be avoided.

In the first step of REPLAN, an optimal solution x is computed based on the original arrival sequence $\mathcal{A}$. In order to update x after the occurrence of a significant change, we need a fast algorithm that does not modify irreversible assignments.

For the initial optimal solution, both the branch-and-price algorithm proposed in Chapter 4 and the Progressive Method proposed in Chapter 5 are satisfactory. For the updates, though, we consider the Progressive Method as more appropriate, as it has better computational performance.

7.3.2 Nonexistence of Competitive Algorithms for the VPPO

The following description of the Online Scheduling Problem, presented by Grötschel et al. (2001) [48], shows that this problem is very similar to the online versions of the VPP.

> In scheduling, one is concerned with the distribution of jobs (activities) to a number of machines (the resources). In our example, one is given m identical machines and is faced with the task of scheduling independent jobs on these machines. The jobs become available at their release dates, specifying their processing times. An online algorithm learns the existence of a job only at its release date. Once a job has been started on a machine

the job may not be preempted and has to run until completion. However, jobs that have been scheduled but not yet started may be rescheduled. The objective is to minimize the average flow time of a job, where the flow time of a job is defined to be the difference between the completion time of the job and its release date

The Online Scheduling Problem is very challenging from the theoretical point of view. Namely, there is no competitive algorithm for it even when restricted to instances containing only one machine (see Fiat & Woeginger (1998) [35]). Consequently, the nonexistence of competitive algorithms for the original online versions of the VPP, proved by Winter (1998) [91], can not be considered surprising.

In his proofs, Winter uses families of instances containing tracks of different sizes. We show that there is no competitive algorithms for the problem even if all the tracks have the same size.

Theorem 7.3. *There is a family of instances of $\mathcal{I}_0$ for which any online algorithm for the* VPPO *produces solutions with $O(\sqrt{n})$ crossings.*

Proof. Let us assume that $\beta = m$ and that $\tau(a_j) = \tau(d_j) = \mathcal{T}_0$, $0 \leq j < m$. Departure d_i is of type $\mathcal{T}_{i-m}$ for $m \leq i < n' = n - m$. Therefore, sequence $\mathcal{D}$ can be described as follows:

$$\mathcal{D} = \{0, 0, ..., 0, 1, 2, 3, ..., n'\}.$$

If the first m arrivals are assigned to different tracks, i.e., if there is no pair of arrivals of type $\mathcal{T}_0$ assigned to the same track, we use the arrival sequence $\mathcal{A}_1$, whose last n' elements are such that $\tau(a_{qm+l}) = \tau(d_{(m-q)m+l})$, $0 \leq l < m$ and $1 \leq q < m$. In other words, if we partition the last n' departures of $\mathcal{D}$ into $m-1$ subsequences of size m and invert the order of the sequences, we obtain the description of the last n' elements of $\mathcal{A}_1$. Therefore, $\mathcal{A}_1$ is as follows:

$$\begin{aligned}\mathcal{A}_1 = \{\ & 0, 0, \ldots, 0,\\ & n'-m+1, n'-m+2, \ldots, n',\\ & n'-2m+1, n'-2m+2, \ldots, n'-m,\\ & \vdots\\ & 1, 2, \ldots, m\}.\end{aligned}$$

Clearly, there is a semi-crossing involving each pair of units belonging to different subsequences. A solution without crossings can be obtained if each arrival a_j is assigned to track $s_{\lceil j/m \rceil}$, $1 \leq j \leq n$. Conversely, if an online algorithm assigns the first m arrivals to different tracks, it will produce a solution where each of the last $m-1$ subsequences contains an element that will be assigned to a track containing an element of another subsequence, as not all the elements of a subsequence can be assigned to the same track. Consequently, these solutions contain $O(m)$ crossings.

Suppose now that at least two of the first m arrivals are assigned to the same track. In this case, we will use the arrival sequence $\mathcal{A}_2$ whose last n' elements are such that $\tau(a_{m+q(m-1)+l}) = \tau(d_{m+(m-q-1)(m-1)+l})$, $0 \leq l < m-1$ and $0 \leq q < m$. In this case, the last n' departures of $\mathcal{D}$ are divided in m subsequences of size $m-1$, and similarly to $\mathcal{A}_1$, we invert the order of the subsequences in order to obtain $\mathcal{A}_2$. Therefore, $\mathcal{A}_2$ is as follows:

$$\begin{aligned}
\mathcal{A}_2 = \{ \quad & 0, 0, \ldots, 0, \\
& n' - (m-1) + 1, n' - (m-1) + 2, \ldots, n', \\
& n' - 2(m-1) + 1, n' - 2(m-1) + 2, \ldots, n' - (m-1), \\
& \vdots \\
& 1, 2, \ldots, m-1\}.
\end{aligned}$$

Again, there is a semi-crossing involving each pair of vehicles belonging to different subsequences, and a solution without crossings can be obtained if each arrival a_j is assigned to track $s_{(j \mod m)+1}$, $0 \leq j < n$. Conversely, we remark that there will be at least one track which will remain empty after the assignment of the first m arrivals, and this track will contain elements of at least two of the m subsequences. As a consequence, it is clear that a solution provided by any online algorithm will contain $O(m)$ crossings.

We conclude that every online algorithm produces solutions requiring $O(m) = O(\sqrt{n})$ crossings for instances where crossings are not necessary, i.e., there is no competitive online algorithm for instances $\mathcal{I}_0$ of VPPO. $\square$

For instances $\mathcal{I}_1$, Winter also shows that there is no competitive online algorithm using a family of instances containing tracks of different sizes. It is possible to make a small modification of his proof in order to show that this claim is also valid if all the tracks have the same size.

Corollary 7.4. *There is a family of instances of $\mathcal{I}_1$ whose optimal solutions require one crossing and for which any online algorithm for the* VPPO *produces solutions with $O(n)$ crossings.*

Proof. Winter (1998) [91] constructed a family I of instances of $\mathcal{I}_1$ whose optimal solutions have one crossings and for which online algorithms will always produce matchings that require $O(n)$ crossings. Instances of I contain two tracks, one with l parking positions and the other with $l+1$. There are three vehicle types, with $t_1 = t_2 = l$ and $t_3 = 1$. The departure sequence used in the proof is as follows:

$$\tau(d_i) = \begin{cases} \mathcal{T}_1 & \text{if } l+2 \le i \le n; \\ \mathcal{T}_2 & \text{if } 1 \le i \le l; \\ \mathcal{T}_3 & \text{if } i = l+1. \end{cases}$$

Sequences $\mathcal{A}_1 = \{a_1^1, a_2^1, \dots, a_n^1\}$ and $\mathcal{A}_2 = \{a_1^2, a_2^2, \dots, a_n^2\}$ are such that

$$\tau(a_i^1) = \begin{cases} \mathcal{T}_1 & \text{if } i = 1 \le i \le l; \\ \mathcal{T}_2 & \text{if } l+1 \le i \le n-2 \vee i = n; \\ \mathcal{T}_3 & \text{if } i = n-1, \end{cases}$$

and

$$\tau(a_i^2) = \begin{cases} \mathcal{T}_1 & \text{if } i = 1 \vee 3 \le i \le l+1; \\ \mathcal{T}_2 & \text{if } l+2 \le i \le n; \\ \mathcal{T}_3 & \text{if } i = 2. \end{cases}$$

Winter proved his theorem for LIFO tracks. In order to adapt his proof to ours, we inverted his original sequences $\mathcal{A}_1$ and $\mathcal{A}_2$, as we are assuming that the tracks are FIFOs.

Both $\mathcal{A}_1$ and $\mathcal{A}_2$ admit optimal assignments to $\mathcal{D}$ and to $\mathcal{S}$ containing only one crossing, but Winter shows that online algorithms will produce solutions with $O(n)$ crossings if the arrival sequence is correctly chosen after the assignment of the first arrival. Namely, we choose $\mathcal{A}_1$ if the first arrival is assigned to the track with $l+1$ positions and $\mathcal{A}_2$ if the first arrival is assigned to the track with l positions.

Based on I, we construct a family I' of instances of the VPPO as follows. There are two tracks, both with $l+1$ parking positions. There are four vehicle types, with $t_1 = t_2 = l$ and $t_3 = t_4 = 1$. The arrival and the departure of type $\mathcal{T}_4$ are added to the beginning of their respective sequences, while the other elements appear in the sequences in the same order described in the original proof. After the assignment of the unit of type $\mathcal{T}_4$ to one of the tracks, we have the same scenario described by Winter. Consequently, both his examples and analyses hold in this case, which allows us to conclude that any online algorithm for the VPPO will produce solutions with $O(\sqrt{n})$ crossings for instances in $\mathcal{I}_1$ whose optimal solutions require just one crossing. $\square$

It follows from these results that REPLAN is not competitive for the VPPO, but it is important to reinforce that competitive analysis is based on the performance of online algorithms in worst-case scenarios that may be unlikely to happen in real-world scenarios.

7.4 The Robust Vehicle Positioning Problem

Roughly speaking, a robust solution for an optimization problem is a solution which remains feasible even if some parameters suffer limited and predictable modifications. This description is purposely inaccurate, as robustness may have different meanings when applied to different scenarios.

Hamdouni et al. (2004) [51] and Hamdouni, Soumis & Desaulniers (2005) [52] presented the first investigations of robustness in the context of the VPP. In these works, the authors observed that configurations containing vehicles of several types are typically more sensitive to disruptions. Based on this fact, they introduced the concept of *uniform tracks* and proposed formulations for the problem that limit the number of different vehicle types that can be assigned to each track. Kroon, Lentink & Schrijver (2006) [70] suggested a model whose solutions must have a certain number of uniform tracks. Because uniform tracks do not contain crossings, several constraints and variables can be removed from the formulation, making the resulting ILPs more compact and easier to solve. The formulations proposed by Hamdouni et al. and Kroon, Lentink, and Schrijver have good computational performances, but it is clear that the solutions produced by them are not necessarily optimal for the original problem.

There are some articles dedicated to robustness in the context of the Shunting Problem Over a Hump (see Cicerone et al. (2007) [20] and Cicerone et al. (2009) [21]). These works employ the *recoverable robustness methodology*, proposed by Liebchen et al. (2007) [73] (see also Liebchen et al. (2009) [74]). The main idea of this technique is to adjust a solution that became infeasible after modifications in the input. Ideally, polynomial-time algorithms are employed in these operations, but this goal usually can only be achieved if the changes in data are limited.

In our opinion, neither of these approaches is completely satisfactory for the VPP. Restricting the number of different vehicle types assigned to tracks is an artificial constraint that may interfere with the quality of the solutions, and the recoverable robustness methodology is also not suitable, as disruptions never lead to infeasibility in our case.

7.4.1 A Robust Model for the VPP

In the previous chapters, crossings were equally penalized and based on the expected arrival sequence $\mathcal{A}$. In real-world scenarios of the VPP, though, changes in $\mathcal{A}$ may happen, and these modifications usually do not have a completely erratic and unpredictable behavior. Typically, we know that some crossings are more likely to appear (and, in some cases, to dissappear) than others.

Our idea is not only to minimize the problems caused by disruptions in $\mathcal{A}$, but also to eventually benefit from them if they happen. Namely, if a vehicle a is not involved in crossings, then it is undesirable to assign it to a track where other vehicles which are very likely to change their relative arrival order with a will be parked, as this would increase the probability of unexpected crossings. Conversely, if crossings are unavoidable, then it would be better if the existing ones involved pairs of arrivals which are more likely to have their relative arrival order exchanged, as the original plan would improve if these modifications happen. A penalization system able to prioritize matchings with the characteristics above seems to be more appropriate and satisfactory for scenarios where disruptions are expected.

Let $p : \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$ be a *penalty function.* The following model of the VPP is a variation of (**LU**) that uses p in the crossing inequalities:

$$\begin{array}{lll}
(\mathbf{RU}) & \min \sum\limits_{(a,s,d)} r_{a,s,d} & \\
\text{(i)} & \sum\limits_{(s,d)} x_{a,s,d} = 1 & a \in \mathcal{A} \\
\text{(ii)} & \sum\limits_{(a,s)} x_{a,s,d} = 1 & d \in \mathcal{D} \\
\text{(iii)} & \sum\limits_{(a,d)} x_{a,s,d} = \beta & s \in \mathcal{S} \\
\text{(iv)} & \sum\limits_{a' \neq a} p(a',a) x_{a',s,d} + \sum\limits_{d' \leq d} x_{a,s,d'} \leq 1 + r_{a,s,d} & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D} \\
 & x_{a,s,d} \in \{0,1\} & (a,s,d) \in \mathcal{F} \\
 & r_{a,s,d} \in \mathbb{R}^+ & (a,s,d) \in \mathcal{A} \times \mathcal{S} \times \mathcal{D}.
\end{array}$$

This model uses binary variables $x_{a,s,d}$, with $a \in \mathcal{A}$, $s \in \mathcal{S}$, $d \in \mathcal{D}$, and $\tau(a) = \tau(d)$, where $x_{a,s,d} = 1$ if and only if arrival a is assigned to departure d and to track s. Equalities (**RU**)(i) and (**RU**)(ii) are assignment constraints for arrivals and departures, respectively, while Equalities (**RU**)(iii) are capacity restrictions for each track in $\mathcal{S}$. Formulations (**LU**) and (**RU**) differ in their crossing constraints, i.e., Inequalities (**LU**)(iv) and (**RU**)(iv), respectively. If sums $\sum\limits_{a'<a} x_{a',s,d}$ are substituted for sums $\sum\limits_{a'\neq a} p(a,a')x_{a',s,d}$, given a vehicle a, it is possible to penalize crossings involving vehicles a', $a' > a$, and to reduce the penalties involving vehicles a'', $a'' < a$, in order to consider eventual changes in the relative arrival order of a, a', and a''. Moreover, because the image of penalty function p is $\mathbb{R}^+$, the crossings variables $r_{a,s,d}$ also belong to $\mathbb{R}^+$, and not to $\{0,1\}$ as in (**LU**).

Let x be a solution of an instance of VPP and p be a penalty function. We will denote the *robustness of x according to p* by

$$\mathcal{R}(p,x) = \sum_{a',a} f(x,p,a',a),$$

where

$$f(x,p,a',a) = \begin{cases} 0 & \text{if } a' = a; \\ 0 & \text{if } a' \text{ and } a \text{ are not in the same track in } x; \\ p(a',a) & \text{else.} \end{cases}$$

Typically, one is interested in a penalty function p such that $p(a_i,a_j) \geq p(a_k,a_l)$ if and only if the probability with which a_i arrives before a_j is greater than the probability with which a_k arrives before a_l for $a_i, a_j, a_k, a_l \in \mathcal{A}$. We say that a function p with this property is a *satisfactory penalty function*.

7.4.2 Defining Function p

Initially, we defined p as a function in $\mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$. However, as our idea is to employ (**RU**) to solve the problem, we can restrict p to the set of functions in $\mathcal{A} \times \mathcal{A} \to [0,1]$. Namely, if (x,r) is an integer solution of (**RU**), (a^*,a) is a pair of arrivals, s is the track assigned to a^* in x, and $p(a^*,a) > 1$, we would have the following situation:

$$\begin{aligned} r_{a,s,d} &\geq \sum_{a'\neq a} p(a',a)x_{a',s,d} + \sum_{d'\leq d} x_{a,s,d'} - 1 \\ &\geq p(a^*,a)x_{a^*,s,d} + \sum_{\substack{a'\neq a \\ a'\neq a^*}} x_{a',s,d} + \sum_{d'\leq d} x_{a,s,d'} - 1 \\ &\geq p(a^*,a)x_{a^*,s,d} - 1 \\ &> 0. \end{aligned}$$

Basically, if a^* is assigned to d, $r_{a,s,d}$ will be bigger than zero even if a is not assigned to track s. In this case we will penalize a crossing that does not occur, which is clearly unnecessary. For this reason, we will assume from now on that $p(a', a) = P(a' < a)$, where $P(a' < a)$ is the probability with which a' will arrive before a.

Ideally, the value of $P(a' < a)$ for every pair of vehicles a, a' in $\mathcal{A}$ is determined with the help of a representative dataset describing the historical events, i.e., the real arrival sequences in several past situations. However, information of this nature are not always available. Besides, an analysis focused on a specific scenario would probably lead to results that lack generality. Therefore, we will make the following assumption regarding the probability function P: given arrivals a_i, a_j, and a_k, if $a_i < a_j < a_k$, then $P(a_k < a_i) < P(a_j < a_k)$. It is clear that this estimate is not 100% accurate (this claim may be false in the cases where a_i frequently has long delays while a_j and a_k are almost always punctual, for example), but it is suitable for a generic approach.

We will say that a solution describes a *perfectly robust matching* if each arrival a_i is assigned to the same track as arrivals a_{i-m} and a_{i+m}, and if $i - m < 0$ $(i + m > n)$, then a_i is the first (last) vehicle assigned to its track.

Proposition 7.5. *A solution describing a perfectly robust matching may produce an arbitrarily bad solution for an instance of the* VPP *which admits a crossing-free solution.*

Proof. Consider the family of instances with n vehicles, $t = n$, $m = \beta$, arrival sequence

$$\begin{aligned}\mathcal{A} = \{ \quad & n-m+1, n-m+2, \ldots, n, \\ & n-2m+1, n-2m+2, \ldots, n-m, \\ & \vdots \\ & 1, 2, \ldots, m\},\end{aligned}$$

and departure sequence

$$\mathcal{D} = \{1, 2, \ldots, n\}.$$

Instances of this family belong to $\mathcal{I}_0$. Namely, if we assign each arrival a_i to track $s_{\lceil (i-1)/m \rceil}$, $1 \leq i \leq m$, we will obtain a crossing-free solution. This solution is very sensitive to disruptions, though. If $a_i < a_j$ but a_j comes before a_i and $\lfloor (i-1)/m \rfloor = \lfloor (j-1)/m \rfloor$, an unexpected crossing will occur.

Conversely, a perfectly robust matching requires the assignment of each arrival a_i to the $(((i-1) \mod m)+1)$-th position of some track. With this distribution, each pair of vehicles assigned to the same track is involved in a crossing, which shows that the perfect robust matching produces an arbitrarily bad solution. $\square$

Finally, we recall the robustness criterion of Hamdouni et al. (2004) [51] and remember that if a and a' are assigned to the same track and $\tau(a)=\tau(a')$, then we do not have to worry about crossings involving them. Consequently, having two vehicles of same type with very close arriving times assigned to the same track does not reduce the robustness of a matching.

We define the following penalty function p' based on these observations:

$$p'(a',a) = \begin{cases} 0 & \text{if } \tau(a)=\tau(a'), \\ 2^{a-a'-1} & \text{if } a<a', \\ 1-2^{a'-a-1} & \text{if } a>a'. \end{cases}$$

It is clear that p' is a satisfactory penalty function. Moreover, it explicitly favors uniform tracks. Namely, a solution x describing a matching containing only uniform configurations is such that $\mathcal{R}(p',x)=0$.

As $|a-a'|$ increases, though, the value of $p'(a',a)$ converges exponentially to zero if $a<a'$ and to one if $a>a'$. As such small values may bring numerical difficulties to the resulting ILPs, we propose the following function p^*:

$$p^*(a',a) = \begin{cases} 0 & \text{if } \tau(a)=\tau(a'), \\ 2^{a-a'-1} & \text{if } 0<a'-a<\gamma(m+1), \\ 0 & \text{if } a'-a\geq\gamma(m+1), \\ 1-2^{a'-a-1} & \text{if } 0<a-a'<\alpha(m+1), \\ 1 & \text{if } a-a'\geq\alpha(m+1). \end{cases}$$

Numbers α and γ are *robustness factors* that should be calibrated and tested by the user. In our computational results, only values of α and γ bigger than zero and smaller then or equal to one were considered.

For every solution x of (**RU**), it is clear that $\mathcal{R}(p^*,x)\leq\mathcal{R}(p',x)$. We remark that, with this penalty function, if x describes a perfectly robust matching, then $\mathcal{R}(p^*,x)=0$.

7.5 Computational Results

We present now some computational results involving model (**RU**). Our tests were conducted on a 64-bits Intel(R) Core(TM)2 Quad with 2.83 GHz, 8 GB of RAM memory, running openSuse Linux 11.2. Our code is implemented in `C++` and was compiled using `g++` 4.4.1 and with the callable library of CPLEX 12.2.0.2 ILOG (2010) [57].

Tables 16 and 17 show the results of computations using an algorithm based on the Exact Heuristic Method applied to the (**RU**) similar to Algorithm 7 (described in Chapter 5) for the VPP$_+$ and for the VPP$_+^P$, respectively. We used the function p^* to penalize crossings.

The first column of Table 16 contains the name n-m-t of the instances. Following our notation, n is the number of arrivals, m is the number of tracks, and t is the number of types. In Table 17, the names p-w-n-m-e-t of the instances indicate the number p of weekday periods, the number w of weekend periods, the number n of departures on weekday periods, the number m of tracks, the number e of trips on weekend periods, and the number t of vehicle types. Based on these parameters, the arrival sequences $\mathcal{A}$ were randomly generated (i.e., the type of each vehicle was uniformly chosen among the t possibilities), while their respective departure sequences $\mathcal{D}$ were obtained after the application of Algorithm 1 in $\mathcal{A}$. For instances of VPP^P, we used the scheme described in Chapter 2, i.e., using Algorithm 2. Finally, columns Iterations, V_{IP}, α, γ, and Time indicate the number of iterations, the cost value of the best solution obtained, the robustness factors α and γ, and the time consumed for the computation.

For each instance, we chose the biggest values of α and γ which generated a model that could be solved in few hours. Typically, the running time increases together with both robustness factors, but γ is clearly the most challenging parameter. In our tests, good solutions could be quickly obtained for $\alpha = 1$ and $\gamma = 0.35$. It is important to remark that good matchings could be obtained both for the VPP and for the VPP^P after few minutes of computation even when larger values of α and γ were employed.

Finally, we also simulated some disruptions in these scenarios in order to test the online algorithm REPLAN. Basically, based on an original plan, we considered modifications of the arrival sequence that would result in the occurrence of unexpected crossings and computed a new plan using Algorithm 7. It is obviously hard to evaluate the applicability of REPLAN based

Instance	Iterations	V_{IP}	Crossings	α	γ	Time
20-4-12	6	4.875	4	1.0	1.0	15
63-7-8	2	0	0	0.5	0.5	9
64-8-12	1	0	0	0.5	0.5	4
96-12-7	1	0	0	0.5	0.5	5
150-15-6	1	0	0	0.5	0.5	39
150-15-15	1	0	0	0.5	0.5	312
160-20-10	1	0	0	0.5	0.5	75
200-20-10	1	0	0	0.5	0.5	172

Table 16: Solving the VPP using (**RU**)

Instance	Iterations	V_{IP}	Crossings	α	γ	Time
4-2-20-4-15-5	31	3.5625	3	0.5	0.5	28271
5-2-36-6-24-5	21	0.875	1	1.0	0.35	210
6-3-30-5-18-5	26	4.875	4	1.0	0.35	130
10-3-64-8-48-6	14	0	0	1.0	0.35	78
10-4-80-10-56-8	14	0	0	1.0	0.25	948
10-4-120-12-90-8	14	0	0	1.0	0.35	323
10-4-150-15-100-10	14	0	0	1.0	0.35	2223

Table 17: Solving the VPP^P using (**RU**)

on these random tests, but we believe that this algorithm has a very good potential when combined with the Progressive Method, as it was frequently able to compute new matchings that avoided crossings very quickly.

Index

Bibliography

[1] K. Aardal, G. Nemhauser & R. Weismantel. *Handbook of Discrete Optimization*. Elsevier, Amsterdam, 2005.

[2] T. Achterberg. *Constraint integer programming*. Ph.D. thesis, TU Berlin , 2007.

[3] N. Ascheuer. *Hamiltonian path problems in the on-line optimization of flexible manufacturing systems*. PhD thesis, TU Berlin, 1995.

[4] N. Ascheuer, M. Grötschel, N. Kamin & J. Rambau. *Combinatorial online optimization in practice*. OPTIMA - Mathematical Programming Society Newsletter, 1998.

[5] E. Balas & M. J. Saltzman. *Facets of the three-index assignment polytope*. Discrete Appl. Math. **23**(3):201–229, 1989. ISSN 0166-218X.

[6] R. Baldacci, P. Toth & D. Vigo. *Recent advances in vehicle routing exact algorithms*. 4OR: A Quarterly Journal of Operations Research **5** (4):269–298, 2007.

[7] A. Ben-Tal & A. Nemirovski. *Robust optimization - methodology and applications*. Mathematical Programming **92**(3):453–480, 2002.

[8] A. Ben-Tal, S. Boyd & A. Nemirovski. *Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems*. Mathematical Programming B **107**:63–890, 2006.

[9] D. Bertsimas & M. Sim. *Robust discrete optimization and network flows*. Mathematical Programming Series B **98**:49–71, 2003.

[10] H. Beyer & B. Sendhoff. *Robust optimization - a comprehensive survey*. Computer Methods in Applied Mechanics and Engineering **196** (33-34):3190–3218., 2007.

[11] A. Billionnet & S. Elloumi. *Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem*. Math. Program. **109**(1):55–68, 2007.

[12] U. Blasum, M. Bussieck, W. Hochstättler, C. Moll, H. Scheel & T. Winter. *Scheduling trams in the morning*. Mathematical Methods of Operations Research **49**:137–148, 1999.

[13] H. L. Bodlaender & K. Jansen. *Restrictions of graph partition problems. part i*. Theor. Comput. Sci. **148**(1):93–109, 1995.

[14] H. L. Bodlaender, K. Jansen & G. J. Woeginger. *Scheduling with incompatible jobs*. In *Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 37–49, 1992. URL `citeseer.ist.psu.edu/bodlaender94scheduling.html`.

[15] J. Bondy & U. Murty. *Graph Theory with Applications.* Elsevier North-Holland, 1976.

[16] R. Borndörfer. *Combinatorial packing problems.* In M. Grötschel, (Ed.), *The Sharpest Cut - The Impact of Manfred Padberg and His Work*, vol. 4 of *Series on Optimization*, pp. 19–32. MPS-SIAM, 2004.

[17] A. Borodin & R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[18] R. Burkard, M. Dell'Amico & S. Martello. *Assignment Problems.* SIAM, 2009.

[19] V. Chvátal. *Linear Programming.* A Series of Books in the Mathematical Sciences. Freeman, 1983. ChVA v 83:1 P-Ex.

[20] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni & A. Navarra. *Robust algorithms and price of robustness in shunting problems.* In C. Liebchen, R. K. Ahuja & J. A. Mesa, (Eds.), *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. ISBN 978-3-939897-04-0. URL http://drops.dagstuhl.de/opus/volltexte/2007/1175.

[21] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni & A. Navarra. *Recoverable robustness for train shunting problems.* Algorithmic Operations Research **4**:102–116, 2009.

[22] S. Cook. *The complexity of theorem proving procedures.* Proceedings of the 3rd Annual ACM Symposium on Theory of Computing pp. 151–158, 1971.

[23] T. Cormen, C. Leiserson, R. Rivest & C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[24] S. Cornelsen & G. Di Stefano. *Track assignment.* J. of Discrete Algorithms **5**(2):250–261, 2007. ISSN 1570-8667.

[25] Z. J. Czech. *Parallel simulated annealing for the set-partitioning problem.* In *Proceedings of the 8th Euromicro conference on Parallel and distributed processing*, EURO-PDP'00, pp. 343–350, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0500-7. URL http://portal.acm.org/citation.cfm?id=1897179.1897229.

[26] G. Dantzig. *Activity Analysis of Production and Allocation*, chap. Maximization of a linear function of variables subject to linear inequalities, pp. 339–347. Wiley, 1951.

[27] G. Dantzig. *Linear programming under uncertainty.* Management Science **1**(3):197–206, 1955.

[28] M. Demange, G. Di Stefano & B. Leroy-Beaulieu. On the online track assignment problem. Technical report, ARRIVAL Project, December 2006.

[29] G. Di Stefano & Koči. *A graph theoretical approach to the shunting problem.* Eletronic Notes in Theoretical Computer Science **92**:16–33, 2004.

[30] G. Diepen, J. van den Akker & J. Hoogeveen. *Integrated gate and bus assignment at amsterdam airport schiphol*, 2009.

[31] J. Edmonds. *Paths, trees, and flowers.* Canad. J. Math. **17**:449–467, 1965. ISSN 0008-414X.

[32] W. Eisenberg & D. Prigge. *Wertschöpfunspotenziale in betribshof besser nutzen.* Der Betrieb , 2002.

[33] L. Euler. *Solutio problematis ad geometriam situs pertinentis.* Commentarii academiae scientiarum Petropolitanae **8**:128–140, 1741.

[34] F. Fernandoy, H. Oerter, H. Meyer, F. Wilhelms, W. Graf & J. Schwander. *Temporal and spatial variation of stable isotope ratios and accumulation rates in the hinterland of neumayer station, east antarctica.* Journal of Glaciology **56**, 2010.

[35] A. Fiat & G. Woeginger. *Online Algorithms - The State of the Art. Lecture Notes in Computer Science 1442.* Springer, 1998.

[36] F. Fisher, R.A.; Yates. *Statistical tables for biological, agricultural and medical research.* Oliver & Boyd, 1948.

[37] P. Føns. Decision support for depot planning in the railway industry. Master's thesis, Technical University of Denmark, 2006.

[38] G. Froyland, T. Koch, N. Megow & H. Wren. Optimizing the landside operation of a container terminal. Technical Report ZIB Report 06-06, Zuse-Institute Berlin, 2006.

[39] G. Gallo & F. Di Miele. *Dispatching buses in parking depots.* Transportation Science **35**(3):322–330, 2001.

[40] M. R. Garey & D. S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. ISBN 0-7167-1045-5. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[41] R. Gomory. *Outline of an algorithm for integer solutions to linear programs.* Bulletin of the American Mathematical Society **64**:275–278, 1958.

[42] R. Graham. *Bounds for certain multiprocessor anomalies.* Bell Systems Technical Journal **45**:1563–1581, 1966.

[43] M. Grohe. *Computing crossing numbers in quadratic time.* J. Comput. Syst. Sci. **68**:285–302, March 2004. ISSN 0022-0000. URL http://portal.acm.org/citation.cfm?id=985660.985664.

[44] M. Grötschel. *Discrete mathematics in manufacturing.* In *Proc. of the Second Int. Conf. on Industrial and Applied Mathematics*, pp. 119–145. Society for Industrial and Applied Mathematics, SIAM, 1991.

[45] M. Grötschel, L. Lovász & A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, Berlin, second edition, 1988. ISBN 3-540-56740-2.

[46] M. Grötschel, R. Graham & L. Lovász. *Handbook of Combinatorics.* Elsevier, MIT Press, 1995.

[47] M. Grötschel, S. Krumke & J. Rambau, (Eds.). *Online Optimization of Large Scale Systems.* Springer-Verlag, Berlin, 2001. ISBN 3-340-42459-8.

[48] M. Grötschel, S. Krumke, J. Rambau, T. Winter & U. Zimmermann. *Online Optimization of Large Scale Systems*, chap. Combinatorial online optimization in real time, pp. 679–704. Springer-Verlag, 2001.

[49] M. Grötschel, B. Hiller & A. Tuchscherer. Combinatorial online optimization: Elevators and yellow angels. Technical Report ZIB Report 07-36, Zuse-Institute Berlin, 2007.

[50] H.-O. Günther & K. Kim, (Eds.). *Container Terminals and Automated Transport Systems.* Springer, Berlin, 2005.

[51] M. Hamdouni, G. Desaulniers, O. Marcotte, F. Soumis & M. van Putten. *Dispatching buses in a depot using block patterns.* GERAD's thematic workshop on Optimization in Public Transit, Montreal, Canada, 2004.

[52] M. Hamdouni, F. Soumis & G. Desaulniers. *Dispatching buses in a depot minimizing mismatches.* 7th IMACS, Scientific Computing Toronto, Canada, 2005.

[53] P. Hammer & A. Rubin. *Some remarks on quadratic programming with 0-1 variables.* Revue Francaise d'Informatique et de Recherche Operationelle **4**(3):67–79, 1970.

[54] R. S. Hansmann. *Optimal Sorting of Rolling Stock.* PhD thesis, Institute of Mathematical Optimization, Braunschweig Technical University Carolo-Wilhelmina, 2010.

[55] B. Hiller. *Online Optimization: Probabilistic Analysis and Algorithm Engineering.* PhD thesis, TU Berlin, 2010.

[56] B. Hiller & T. Vredeveld. *Probabilistic analysis of online bin coloring algorithms via stochastic comparison.* In *Proceedings of ESA*, pp. 528–539, 2008.

[57] ILOG. *CPLEX website.* http://www.ilog.com/products/cplex/, 2010.

[58] R. Jacob. *On shunting over a hump.* Manuscript, 2007.

[59] K. Jansen. *The mutual exclusion scheduling problem for permutation and comparability graphs.* Inf. Comput. **180**(2):71–81, 2003. ISSN 0890-5401.

[60] N. Kamin. *On-line optimization of order picking in an automated warehouse.* PhD thesis, TU Berlin, 1998.

[61] L. Kantorovich. *Mathematical methods of organizing and planning production.* Management Science **6**(4):366–422, 1960.

[62] A. Karlin. *On the performance of competitive algorithms in practice. Lecture Notes in Computer Science 1442.* Springer, 1998.

[63] A. Karlin, M. Manasse, L. Rudolph & D. D. Sleator. *Competitive snoopy caching.* Algorithmica **3**:79–119, 1988.

[64] N. Karmarkar. *A new polynomial time algorithm for linear programming.* Combinatorica **4**(4):373–395, 1984.

[65] R. M. Karp. *Reducibility among combinatorial problems.* Complexity of Computer Computations pp. 85–103, 1972.

[66] L. Kaufmann & F. Broeckx. *An algorithm for the quadratic assignment problem.* European J. Oper. Res. **2**:204–211, 1978.

[67] L. Khachiyan. *A polynomial algorithm in linear programming.* Dokl. Akad. Nauk SSSR **244**(5):1093–1096, 1979.

[68] T. Koch. *Rapid Mathematical Programming.* PhD thesis, 2004.

[69] P. Kouvelis & G. Yu. *Robust Discrete Optimization and Its Applications.* Springer, 1997.

[70] L. Kroon, R. Lentink & A. Schrijver. *Shunting of passenger train units: an integrated approach.* ERIM Report Series Reference No. ERS-2006-068-LIS , 2006.

[71] S. Krumke, W. de Paepe, J. Rambau & L. Stougie. *Online bin-coloring.* In *Proc. of the 9th European Symposium on Algorithms*, pp. 74–84, 2001.

[72] R. Lentink. *Algorithmic Decision Support for Shunt Planning.* PhD thesis, Erasmus Research Institute of Management, Erasmus University Rotterdam, 2006.

[73] C. Liebchen, M. Lübbecke, R. Möhring & S. Stiller. Recoverable robustness. Technical report, ARRIVAL-Project, August 2007.

[74] C. Liebchen, M. Lübbecke, R. Möhring & S. Stiller. *Robust and Online Large-Scale Optimization*, vol. 5868 of *Lecture Notes in Computer Science*, chap. The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications. Springer, 2009.

[75] M. Lübbecke & J. Desrosiers. *Selected topics in column generation.* Operations Research **53**(6):1007–10023, 2005.

[76] M. Lübbecke & J. Desrosiers. *A primer in column generation.* In G.Desaulniers, J.Desrosiers & M. Solomon, (Eds.), *Column Generation*, pp. 1–32. Springer, Berlin, 2005.

[77] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming.* Birkhäuser Verlag, 2005.

[78] C. Ribeiro & F. Soumis. *A column generation approach to the multiple-depot vehicle scheduling problem.* OPERATIONS RESEARCH **42**(1):41–52, 1994.

[79] D. Ryan & B. Foster. *An integer programming approach to schedulling.* In A. Wren, (Ed.), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280. North Holland, Amsterdam, 1981.

[80] M. Salazar-Neumann. *Advances in robust combinatorial optimization and linear programming.* PhD thesis, Universite Libre de Bruxelles, 2009.

[81] M. Savelsbergh. *A branch-and-price algorithm for the generalized assignment problem.* Operations Research **45**(6):831–841, 1997.

[82] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986. SchRI a 86:1.

[83] A. Schrijver. *Combinatorial optimization : polyhedra and efficiency.* Springer, 2003.

[84] D. Sleator & R. Tarjan. *Amortized eciency of list update and paging rules.* Communications of the ACM 28 pp. 202–208, 1985.

[85] M. H. van der Vlerk. *Stochastic integer programming bibliography.* World Wide Web, http://www.eco.rug.nl/mally/biblio/sip.html, 1996-2007.

[86] M. H. van der Vlerk. *Stochastic programming bibliography.* World Wide Web, http://www.eco.rug.nl/mally/spbib.html, 1996-2007.

[87] V. V. Vazirani. *Approximation Algorithms.* Springer, 2003.

[88] S. Vigerske. *Nonconvex mixed-integer nonlinear programming.* http://www.math.hu-berlin.de/∼stefan/B19/, 2009.

[89] D. Villeneuve, J. Desrosiers, M. Lübbecke & F. Soumis. *On*

compact formulations for integer programs solved by column generation. Ann. Oper. Res **139**(1):375–388, 2005.

[90] J. VON NEUMANN. *Discussion of a maximum problem.* unpublished working paper, 1947.

[91] T. WINTER. *Online and Real-Time Dispatching Problems.* PhD thesis, TU Braunschweig, 1998.

[92] T. WINTER & U. ZIMMERMANN. *Discrete online and real-time optimization*, 1998.

[93] T. WINTER & U. ZIMMERMANN. *Real-time dispatch of trams in storage yards.* Annals of Operations Research **96**:287–315, 2000.

[94] L. WOLSEY. *Integer Programming.* John Wiley & Sons, 1998.

Eidesstattliche Versicherung

Ich versichere an Eides statt, dass ich die von mir vorgelegte Dissertation selbstständig angefertigt habe und alle benutzten Quellen und Hilfsmittel vollständig angegeben habe. Die Zusammenarbeit mit anderen Wissenschaftlern habe ich kenntlich gemacht. Diese Personen haben alle bereits ihr Promotionsverfahren abgeschlossen.

Erklärung

Teile dieser Arbeit sind bereits veröffentlicht und zwar:

- *A set partitioning approach to shunting*
  zusammen mit Ralf Borndörfer
  To appear in: Discrete Applied Mathematics.
- *A set partitioning approach to shunting*
  zusammen mit Ralf Borndörfer
  ZIB Report 09-18, Zuse Institute Berlin, 2009.
- *A set partitioning approach to shunting*
  zusammen mit Ralf Borndörfer
  V Latin-American Algorithms, Graphs and Optimization Symposium
  Electronic Notes in Discrete Mathematics, pp. 359-364, Vol. 35, 2009.
- *A Binary Quadratic Programming Approach to the Vehicle Positioning Problem*
  zusammen mit Ralf Borndörfer
  ZIB Report 09-12, Zuse Institute Berlin, 2009.
- *A Binary Quadratic Programming Approach to the Vehicle Positioning Problem*
  zusammen mit Ralf Borndörfer
  To appear in: Proceedings of the Fourth International Conference on High Performance Scientific Computing, March 2-6, 2009, Hanoi, Vietnam, Springer.

Eine Anmeldung der Promotionsabsicht habe ich an keiner anderen Fakultät oder Hochschule beantragt.

Lebenslauf

Carlos Cardonha

geboren am 04.09.1982 in São Paulo

1985 bis 1997:	Besuch der Grundschule in São Paulo
1998 bis 2000:	Besuch des Escola Tecnica Federal de São Paulo in São Paulo
2001 bis 2004:	Studium der Informatik an der Universidade de São Paulo
2005 bis 2006:	Masterstudium der Informatik an der Universidade de São Paulo
Seit 2007:	Gast Wissenschaftlicher Mitarbeiter am Zuse Institute Berlin (ZIB)