O PROBLEMA DO ROTEAMENTO MAIS LUCRATIVO DE UM VEÍCULO COM REBOQUE E COM JANELAS DE TEMPO

HENRIQUE FAVARINI ALVES DA CRUZ

O PROBLEMA DO ROTEAMENTO MAIS LUCRATIVO DE UM VEÍCULO COM REBOQUE E COM JANELAS DE TEMPO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Alexandre Salles da Cunha

Belo Horizonte

Julho de 2020

HENRIQUE FAVARINI ALVES DA CRUZ

THE PROFITABLE SINGLE TRUCK AND TRAILER ROUTING PROBLEM WITH TIME WINDOWS

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: ALEXANDRE SALLES DA CUNHA

Belo Horizonte

July 2020

Cruz, Henrique Favarini Alves da

C957p The profitable single truck and trailer routing problem with time windows [manuscrito] / Henrique Favarini Alves da Cruz. – 2020.
xxiv, 100 f. il.

Orientador: Alexandre Salles da Cunha
Dissertação (mestrado) - Universidade Federal de Minas Gerais;Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.75-79
.
1. Computação – Teses. 2. Otimização combinatória – Teses. 3. Transporte rodoviário – Controle automático – Teses. 4. Logistica – Teses. 5. Empacotamento e cobertura combinatória – Teses. I. Cunha, Alexandre Salles da. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III.Título.

CDU 519.6*61(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

The Profitable Single Truck and Trailer Routing Problem with Time Windows

HENRIQUE FAVARINI ALVES DA CRUZ

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ALEXANDRE SALLES DA CUNHA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. HAROLDO GAMBINI SANTOS
Departamento de Ciência da Computação - UFOP

PROF. DILSON LUCAS PEREIRA
Departamento de Ciência da Computação - UFLA

PROF. CRISTIANO ARBEX VALLE
Departamento de Ciência da Computação - UFMG

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 30 de Julho de 2020.

Acknowledgments

I thank my parents, Virginia and Adelmo, for all the effort they made so that I had a good education. I thank my brother Bruno as well, for his constant friendship.

I am also thankful for my friends for all the support during this intense period. Specially, I thank my partner Guilherme, for all the companionship, patience, and for rooting for me more than anyone else.

Last but not the least, I would like to express my sincere gratitude to my advisor, Prof. Alexandre, for his continued support since before I joined this Masters program years ago, and for believing in me at all times. Thank you for the valuable advices, for the pleasant conversations, and for guiding me into learning so much.

Resumo

Investigamos neste trabalho o Problema do Roteamento Mais Lucrativo de Um Veículo com Reboque e com Janelas de Tempo. Nesse problema, são dados uma frota contendo caminhões e reboques, além de um conjunto de clientes, que fornecem uma recompensa caso sua demanda seja coletada por um veículo. Os reboques podem ser acoplados aos caminhões para aumentar a capacidade do veículo; em compensação, a velocidade do veículo é reduzida, e os custos de viagem são aumentados. Alguns clientes só podem ser atendidos por caminhões desacoplados, o que pode dar origem a rotas em que um reboque é desacoplado do caminhão em vértices satélite dados. O caminhão pode, então, viajar alguns trechos da rota desacoplado, após os quais retorna ao ponto onde o reboque foi estacionado, e a carga coletada pelo caminhão é transferida para o reboque. O objetivo do problema é encontrar uma escolha de veículo e uma rota a ser percorrida por ele, para a qual o lucro, dado pelas recompensas coletadas menos os custos de viagem, seja máximo. Propomos uma formulação de Programação Inteira para o problema, além de desigualdades válidas e estratégias de *lifting*, capazes de reforçar as relaxações lineares da formulação. A formulação desenvolvida emprega apenas variáveis binárias e garante o cumprimento das restrições de capacidade, janelas de tempo e sincronização das operações de acoplamento, transferência de carga e desacoplamento, por meio de cortes de Benders Combinatórios. Pelo que conhecemos, esta é a única formulação compacta no espaço de variáveis que permite múltiplas sub-rotas partindo do mesmo vértice satélite. Em trabalhos anteriores, diferentes variações do Problema de Roteamento de Veículo com Reboque (PRVR) têm sido tratadas. Essas variações são caracterizadas por considerar ou não algumas propriedades importantes na modelagem do problema, como janelas de tempo para os clientes, vértices satélite dedicados, frota heterogênea e tempos de transferência dependentes da carga. Nosso modelo lida com todos esses aspectos listados, além de considerar velocidades diferentes para veículos da frota e velocidades reduzidas para o caminhão quando acoplado a um reboque, o que, segundo nossa pesquisa, ainda não havia sido abordado na literatura sobre PRVRs. Desenvolvemos um algoritmo exato *Branch-and-Cut* para o problema e apresentamos

resultados computacionais. Os resultados sugerem que as desigualdades válidas aqui introduzidas não apenas melhoram os limites duais mas resultam em melhores algoritmos *Branch-and-Cut*.

Palavras-chave: Otimização Combinatória, Problema de Roteamento de Caminhão com Reboque, Janelas de Tempo, Branch-and-Cut.

Abstract

We investigate in this work the Profitable Single Truck and Trailer Routing Problem with Time Windows. In this problem, a fleet of trucks and trailers is given, along with a set of customers, that reward a profit if their demand is collected by a vehicle. Trailers can be attached to trucks in order to increase vehicle capacity, at the expense of reducing vehicle speed and increasing travel costs. Some customers can only be served by decoupled trucks, which may give rise to routes in which a trailer is decoupled from the truck at given satellite depots. The truck can, therefore, travel parts of the route detached, after which it returns to the point where the trailer was parked, and the load collected by the truck is transfered to the trailer. The goal is to find a choice of vehicle and route for which the collected profit minus the route cost is maximum. We propose an Integer Programming formulation for the problem, along with valid inequalities, that enforce Linear Programming Relaxation bounds for the formulation. The formulation developed employs binary variables only and ensures that capacity, time windows and synchronization for decoupling, load transfer and coupling operations are observed through Combinatorial Benders cuts. From our research, this is the only formulation that is compact in the variable space to allow multiple subtours from the same satellite vertex. Previous works have studied different variations of the Truck and Trailer Routing Problem (TTRP). These variations are characterized by some key characteristics of the problem being considered or not, such as customer time windows, dedicated satellite depots, heterogeneous fleet and load-dependent transfer times. Our model deals with all these aspects, plus considering different speed values for vehicles in the fleet, and a decrease in truck speed when it is attached to a trailer. To our knowledge, this modeling aspect has not been handled in the literature for TTRPs. We develop an exact Branch-and-Cut algorithm for the problem and present computational results. The experiments suggest that the valid inequalities introduced not only improve dual bounds, but also result in better Branch-and-Cut algorithms.

Keywords: Combinatorial Optimization, Truck and Trailer Routing Problem, Time

Windows, Branch-and-Cut.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Acronyms

BC	Branch-and-Cut
BP	Branch-and-Price
DCUT	Directed Cutset Constraint
DP	Dynamic Programming
IP	Integer Programming
IWEC	Infeasible Walk Elimination Constraint
LP	Linear Programming
LPR	Linear Programming Relaxation
PSTTRPTW	Profitable Single Truck and Trailer Routing Problem with Time Windows
TSP	Traveling Salesman Problem
TTRP	Truck and Trailer Routing Problem
TTRPTW	Truck and Trailer Routing Problem with Time Windows
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Contents

Chapter 1

Introduction

A common activity that takes place in the dairy industry involves collecting milk from farmyards and bringing it to a dairy, where the product is processed and later distributed. The choice of route to be used in this task has to consider some particularities arisen from the nature of the product being transported. Milking takes place at certain times of the day only, and, for each farmyard, there is a time interval within which it should be collected and properly stored, so it does not lose its properties. Also, collection points may sit at remote locations that cannot be reached by large vehicles, due to accessibility limitations. A usual solution for this is to employ trucks and trailers. A truck is a relatively small vehicle able to access remote locations. A trailer is a larger unit, but it needs to be decoupled from the truck and parked at certain points of the route, so the truck can visit remote locations. After the truck returns to the parking spot, milk can be transferred from truck to trailer, so the truck capacity is released for visiting other remote locations.

Consider an autonomous driver who is hired by a dairy to perform the task described, with a rented vehicle. The driver is paid a given prize for each farmyard serviced by them. They can pick a truck and a trailer from a fleet of distinct vehicles, with different capacity, travel cost and maximum speed values, available from a rental company. This driver would be interested in identifying the choice of vehicle to rent and route for which they would receive the largest profit, considering rent and travel costs. The problem faced by the driver is known as the *Profitable Single Truck and Trailer Routing Problem with Time Windows* (PSTTRPTW).

The PSTTRPTW is a variation of the Truck and Trailer Routing Problem (TTRP) [Chao, 2002], in which the goal is to find a minimum cost set of routes for multiple vehicles, visiting all customers. Applications that inspire TTRPs involve several particularities that can be taken into account when modeling a real-life scenario. Some

of these properties include: time windows for customers to be visited; a heterogeneous fleet of trucks and trailers (i.e., the fact that distinct vehicles may have different capacities and travel costs); the existence of specific spots where trailers can be parked; the time taken for coupling and decoupling operations; the time taken for transferring load from truck to trailer, and the dependency of this time on the amount of load to be transferred. Several combinations of these modeling aspects have been studied in the literature. In this thesis, we consider all the properties listed above for solving the PSTTRPTW.

1.1 Contributions

The main contributions of this thesis are listed in the following, in the order they appear in the text:

- Introduction of an Integer Programming Formulation for the PSTTRPTW. Our model involves a compact set of integer variables, and an exponential number of constraints. Restrictions related to truck capacity in subtours, customer time windows and satellite scheduling are handled through Combinatorial Benders cuts, that eliminate infeasible portions of a route. A truck is allowed to perform any number of subtours at satellite depots; to our knowledge, this is the first model that is compact in the variable space with such properties. Also, we take into account the fact that distinct vehicles may travel at different speeds. This includes a speed decrease when a truck is coupled to a trailer, in comparison to the speed reached by the same truck when decoupled. To our knowledge, this assumption has also not been previously considered.

- Introduction of valid inequalities for the formulation, and lifting strategies for some of the model constraints.

- Development of a Branch-and-Cut algorithm based on our formulation. Two separation algorithms for the Combinatorial Benders cuts are developed. One of them is a heuristic, that identifies violated cuts for fractional solutions of the Linear Programming Relaxations. The other algorithm is an exact one, employed for integer solutions of the relaxations.

1.2 Organization of this Thesis

This thesis is structured in six chapters. The remaining of the text is organized as follows. In Chapter 2, the PSTTRPTW is formally defined; we also present a review of previous works in the literature that deal with related problems. Chapter 3 describes the modeling that was developed for the problem, and introduces an Integer Programming formulation based on it. A formal proof of the correctness of the formulation is shown. We also discuss preprocessing and variable fixing strategies, valid inequalities for the formulation, and how some constraints can be lifted. In Chapter 4, we introduce a Branch-and-Cut algorithm based on the formulation. In Chapter 5, we describe and discuss computational experiments involving different versions of the proposed algorithm. Chapter 6 concludes this work, with final remarks and suggestions for future research.

Chapter 2

The Profitable Single Truck-and-Trailer Routing Problem with Time Windows

In this chapter, we formally introduce the Profitable Single Truck-and-Trailer Routing Problem with Time Windows (PSTTRPTW). We also review previous studies that addressed related problems, in order to place this work amidst the literature.

2.1 Problem Definition

Let $\mathcal{K}$ be a set of trucks and $\mathcal{L}$, a set of trailers. Each truck $k \in \mathcal{K}$ has an associated capacity $Q_k > 0$. Each trailer $l \in \mathcal{L}$ also has a capacity $Q_l > 0$. A truck k can be coupled to a trailer l. This combined vehicle will be denoted by (k, l) and we refer to it as a *complete vehicle*. Truck k can also travel without a trailer attached to it, in which case we will call it a *single truck* and denote it by $(k, 0)$. Therefore, the set of all possible vehicles is $\mathcal{V} := \mathcal{K} \times (\{0\} \cup \mathcal{L})$.

For each vehicle $(k, l) \in \mathcal{V}$, some associated quantities are defined. The vehicle *capacity*, $Q_k + Q_l$, represents the maximum amount of load that the vehicle can carry at any time. For $l = 0$, we define $Q_l = 0$, since the capacity of a single truck is just Q_k. The vehicle *speed*, $s_{kl} > 0$, gives the distance the vehicle travels per time unit. The *travel cost*, $\phi_{kl} > 0$, is related to the amount of resources spent per distance unit traveled by the vehicle. For a given truck $k \in \mathcal{K}$, we will typically have $s_{k0} > s_{kl}$ and $\phi_{k0} < \phi_{kl}$ for any trailer $l \in \mathcal{L}$, meaning that a truck can travel faster and spend less resources when decoupled. Each truck $k \in \mathcal{K}$ also has a corresponding *rent cost*, a_{k0}.

If trailer $l \in \mathcal{L}$ is rent together with truck k, an additional cost a_{kl} is paid.

Let $G = (V, A)$ be a complete, directed graph, of which V is the set of vertices, and A is the set of arcs. Vertices in the subset $C \subset V$ are *customers* and represent locations where some *supply* of a given product is available to be collected by a vehicle. The customer set is partitioned into two subsets: the *vehicle customer* set, $C_{\mathcal{V}}$, and the *truck customer* set, $C_{\mathcal{K}}$. Vehicle customers may be serviced by any vehicle, complete or not, whereas, due to accessibility limitations, truck customers can only be visited by single trucks. Vertex $0 \in V \setminus C$ is the *main depot*, and $S := V \setminus (\{0\} \cup C)$ is the set of *satellite depots*.

The supply of customer $i \in C$ will be denoted by $q_i > 0$. Customer i also has an associated *profit* $p_i > 0$ to be rewarded if that customer is serviced. *Time windows* $[e_i, f_i]$ represent time intervals within which the goods are available to be collected. So, if a customer is serviced, that service must start within that customer time window. If the vehicle arrives at customer i before its time window opening e_i, it must wait at that point until instant e_i, and only then start servicing i. A vehicle cannot arrive at a customer after f_i. If $i \notin C$, we establish that $e_i := 0$ and $f_i := \infty$.

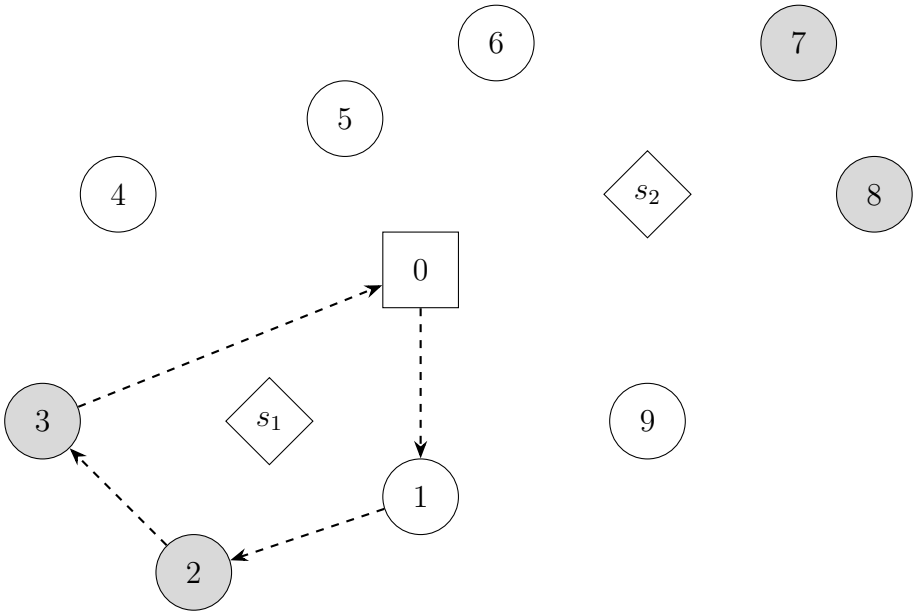
We also consider supply-dependent *service times*, i.e., the time necessary to service a customer is considered to be proportional to its supply. When servicing customer i, q_i product units are loaded from this customer into a vehicle. For truck k, this operation can happen at a rate of o_k units of load per unit of time, hence the service time for customer i by vehicle $(k, l) \in \mathcal{V}$ is given by $\frac{q_i}{o_k}$. If the collected product is, for instance, a fluid like milk, quantity o_k may represent the rate at which a pump, with which truck k is equipped, works.

For each arc $(i, j) \in A$, the distance $d_{ij} > 0$ to travel from i to j is given. In this work, distances are assumed to satisfy the triangular inequality property. We also assume that vehicles travel at constant speed, so the time required for vehicle $(k, l) \in \mathcal{V}$ to travel through arc (i, j) is given by $\frac{d_{ij}}{s_{kl}}$. This arc travel has an associated cost, that depends on the arc distance and the travel cost of the vehicle that crosses (i, j), and is given by $c_{ij}^{kl} := d_{ij}\phi_{kl}$.

We define a *walk* in a directed graph as a finite sequence of adjacent arcs, in which vertices and arcs can appear more than once. A feasible *route* for the PSTTRPTW is defined as a walk in G that is assigned to a vehicle, starting and ending at depot vertex 0, in which customer vertices can be visited at most once. If a customer is visited in a route, it must be serviced, within its time window. The sum of the supplies of the customers that are visited in a route cannot exceed the capacity of the assigned vehicle. If a route is assigned to a complete vehicle, it can also include satellite depots. At these special vertices, the trailer can be decoupled and the truck can continue the

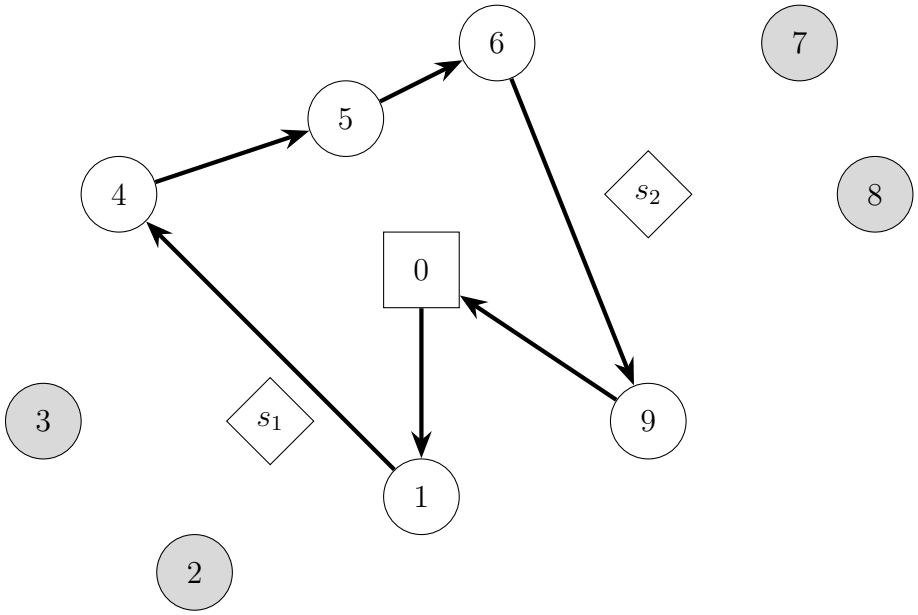
route by itself, visiting truck or vehicle customers. If this happens, the truck must then return to the same depot where the trailer was parked, and the load from the truck can be transferred to the trailer. A portion of the route between two consecutive visits to a satellite depot, traveled by a single truck, is referred to as a *subtour*. After a trailer decoupling at a satellite depot, the route may include any number of subtours from that satellite. For each subtour, the sum of the supplies of the visited customers cannot exceed the capacity of the truck alone. Immediately after the last subtour from satellite depot $i \in S$, the trailer must be coupled to the truck again at i. A trailer can be decoupled and re-coupled any number of times in a route, in any satellite depot. Naturally, a route that includes subtours cannot be assigned to a single truck, but a route without subtours (and, therefore, no satellite depots) can be assigned to any vehicle. Note that truck customers can only be serviced either in routes that are assigned to single trucks, or within subtours.

Figure 2.1 illustrates three possible routes in a given graph. The square labeled with "0" represents the depot vertex, diamonds s_1 and s_2 are satellite depots, and numbered circles are customers, of which the white ones are vehicle customers, and the gray ones represent truck-only customers. In each subfigure, a route is represented by arrows. A continuous arrow means that the corresponding arc is traveled by a complete vehicle, while a dashed one is used for arcs traversed by single trucks.

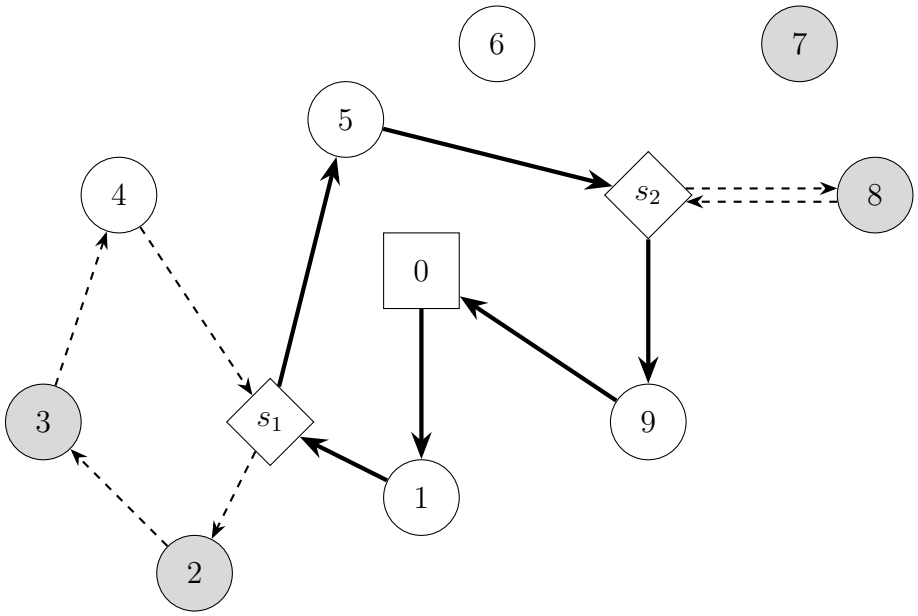


(a) A truck-only route.

In Figure 2.1a, a truck-only route, visiting two truck customers and one vehicle customer, is depicted. Figure 2.1b shows a truck-and-trailer route that does not contain subtours (and therefore no satellite vertices). Note that this route could not include



(b) A truck-and-trailer route without subtours.



(c) A truck-and-trailer route with subtours.

Figure 2.1: Examples of feasible routes.

truck customers. Finally, a truck-and-trailer route with two subtours is illustrated by Figure 2.1c. In this route, customers 2, 3 and 4 are serviced by a decoupled truck in a subtour, after decoupling at s_1. The vehicle is decoupled again later, at s_2, starting another subtour to service customer 8.

In order to attach (detach) trailer $l \in \mathcal{L}$ to (from) truck $k \in \mathcal{K}$, a *coupling/decoupling time* τ_{kl} is spent. After finishing a subtour, when a truck returns

to a satellite depot, the supply that was collected in that subtour is transferred from the truck to the trailer, so that the truck can perform other possible subtours later. Like for the customer service time, this *load transfer* time also depends on the amount of load to be transferred and on the parameter o_k. Thus, if the truck load at the end of a subtour is q', the load transfer time is given by $\frac{q'}{o_k}$. In this work, we consider that a load transfer time is spent after every subtour in the solution route, corresponding to the total supply in the subtour. Note that this assumption is a simplification of a real world scenario, and it may forbid solutions that could be feasible in practice. For example, by choosing not to perform a load transfer at some point, or to perform a partial load transfer, it might after be possible to service a customer with a tight time window, that cannot be reached in time under our assumption. Furthermore, if, at a given point of a route, the trailer is full, a load transfer operation would not even be possible, and the load collected after this point must be stored in the truck.

Therefore, given a route assigned to $(k, l) \in \mathcal{V}$ that visits a customer $i \in C$, the time at which the vehicle arrives at i in this route is given by the sum of the travel times of all arcs that appear before i, the service times of all customers that appear before i, the possible waiting times at customers that appear before i, and the coupling, load transfer and decoupling times that happen before i in the route. Note that the travel time for arcs in subtours is calculated using the single truck speed, instead of the complete vehicle speed.

The profit of a route assigned to a given vehicle is given by the sum of the profits associated to the customers visited in that route, minus the sum of the rent costs, and the costs of the arcs in the route for that vehicle. In the *Profitable Single Truck and Trailer Routing Problem with Time Windows*, the goal is to find a feasible route and vehicle assignment, satisfying all the above restrictions, whose profit is maximum.

To summarize all the PSTTRPTW input data, Table 2.1 shows all the data that define an instance for the problem, along with their respective notation, as discussed in this section, and that will be used through the rest of this thesis.

2.2 Literature Review

The PSTTRPTW is a variation of the Vehicle Routing Problem (VRP) [Dantzig and Ramser, 1959; Cordeau et al., 2001], in which, given a set of vehicles, a minimum cost set of routes servicing every customer must be found. The VRP is, in turn, a generalization of the classic Traveling Salesman Problem (TSP) [Dantzig et al., 1954] in the sense that, in the TSP, there is only one vehicle available.

Table 2.1: PSTTRPTW input data.

Notation	Description
$G = (V, A)$	The complete, directed input graph
$C_{\mathcal{K}}$	Truck customer set
$C_{\mathcal{V}}$	Vehicle customer set
$C = C_{\mathcal{K}} \cup C_{\mathcal{V}}$	Customer set
S	Satellite depots set
q_i	Demand of customer $i \in C$
p_i	Profit rewarded for serving customer $i \in C$
e_i	Customer $i \in C$ time window opening time
f_i	Customer $i \in C$ time window closing time
d_{ij}	Distance of arc $(i, j) \in A$
$\mathcal{K}$	Set of trucks
$\mathcal{L}$	Set of trailers
a_{k0}	Rent cost of truck $k \in \mathcal{K}$
a_{kl}	Rent cost of trailer $l \in \mathcal{L}$ with truck $k \in \mathcal{K}$
s_{kl}	Speed of vehicle $(k, l) \in \mathcal{V}$
ϕ_{kl}	Fuel consumption of vehicle $(k, l) \in \mathcal{V}$
τ_{kl}	Coupling/decoupling time of truck $k \in \mathcal{K}$ and trailer $l \in \mathcal{L}$
o_k	Product transfer rate in vehicle with truck $k \in \mathcal{K}$
Q_k	Capacity of truck $k \in \mathcal{K}$
Q_l	Capacity of trailer $l \in \mathcal{L}$

The VRP has been extensively studied over the years and has a number of variations. We highlight some of them, for their relation to the PSTTRPTW. In the Heterogeneous Fleet Vehicle Routing Problem [Ferland and Michelon, 1988], the available vehicles may have different sizes and capacities, and, in each route, the collected load must not exceed the capacity of the respective vehicle. In the Vehicle Routing Problem with Time Windows (VRPTW) [Kolen et al., 1987; Kallehauge, 2008], the customers have a time interval in which the service must happen. The VRPTW has some versions itself. For instance, time windows may be "hard", in the sense that the vehicle cannot arrive late at any customer at all, or "soft", in which case a penalty value is paid when a time window is violated [Koskosidis et al., 1992]. In the Vehicle Routing Problem with Satellite Facilities [Bard et al., 1998], a set of satellite vertices is available. Along a route, a vehicle can stop at these points and reload, instead of returning to the starting vertex.

Cuda et al. [2015] provide a survey on two-echelon routing problems, which are routing problems involving two levels of routing: from central depots to intermediate facilities, and then to customers. For instance, in the Capacitated Two-Echelon Vehicle Routing Problem (2E-VRP) [Hemmelmayr et al., 2012; Baldacci et al., 2013; Santos

et al., 2015], a set of primary vehicles is available at a central depot, and a set of customers must be serviced, but they are only accessible for a set of secondary vehicles. These vehicles are available at given satellite facilities, each one of which has a given capacity. The problem consists, then, in finding optimal routes for the primary vehicles to the satellites, and for the secondary ones to the customers, while satisfying the capacity constraints.

In the Truck and Trailer Routing Problem (TTRP), a set of capacitated trucks and trailers are available at a central depot and must service a set of customers. Trailers are non-autonomous vehicles that can be pulled by trucks in order to raise capacity, but a subset of customers cannot be visited by these complete vehicles. Trailers can be parked at vehicle customers, and trucks can perform subtours from there, visiting a subset of customers, including trailer customers. The goal is to find an optimal set of feasible routes that services all customers. Like the 2E-VRP, the TTRP also involves two phases of routing: the first level corresponds to sending the complete vehicles to vehicle customers, and the second level consists in finding subtours for decoupled trucks. Chao [2002] formally introduces the problem for the first time, although variations have appeared in earlier works [Semet and Taillard, 1993; Gerdessen, 1996; Bodin and Levy, 2000].

Several studies propose metaheuristic approaches for the standard TTRP outlined above, often introducing variants to the original problem [Hoff, 2006; Scheuerer, 2006; Tan et al., 2006; Lin et al., 2009; Caramia and Guerriero, 2010; Villegas et al., 2010, 2011, 2013; Pasha et al., 2014; Batsyn and Ponomarenko, 2014; Wang et al., 2018]. In [Lin et al., 2011], the Truck and Trailer Routing Problem with Time Windows (TTRPTW) is addressed, combining elements from the VRPTW and the TTRP. Derigs et al. [2013] describe a metaheuristic based on local search and large neighborhood search. This algorithm considers, for the first time, the flexibility of servicing a vehicle customer at any visit, if this customer is visited multiple times (due to subtours). The authors also consider a TTRP variant in which load transfer is not possible, meaning that load collected by a decoupled truck cannot be transferred to a trailer after re-coupling. This limitation is compatible with certain real-life scenarios, either because of the nature of the load, or for security reasons. Non-deterministic versions of the TTRP have also been considered. Torres et al. [2015] handle a TTRP with fuzzy constraints. Mirmohammadsadeghi and Ahmed [2015] address stochastic demands and time windows.

Exact approaches for the TTRP family of problems have received relatively less attention than heuristic ones, although this seems to be changing more recently. Exact algorithms can be classified according to the solution method they use to generate

bounds and prune the search space. *Branch-and-Price* (BP) [Barnhart et al., 1998] approaches are Branch-and-Bound algorithms that generate valid dual bounds by means of column generation. As such, they involve a pricing subproblem, typically solved by Dynamic Programming (DP), when the problem is a VRP or one of its variations that are defined by multiple vehicles. Among this group, Drexl [2011] deals with a "generalized" TTRP, which includes transshipment locations. In these vertices, a trailer can be parked to perform load transfers. This work considers a heterogeneous fleet, and both customers and transshipment locations have time windows.

Parragh and Cordeau [2017] introduce a set-partitioning formulation and a BP algorithm with variable neighborhood search for a TTRPTW in which, like in [Derigs et al., 2013], it is possible to choose when to service vehicle customers that are visited multiple times. Rothenbächer et al. [2018] also consider this possibility, along with load-dependent transfer times.

Another category of exact approaches, *Branch-and-Cut* (BC) algorithms tackle the problem formulations, that involve a usually large set of constraints, by means of cutting planes generation, and try to devise inexpensive ways of dynamically adding some of them to Linear Programming Relaxations (LPRs) of the model. In [Drexl, 2014], BC algorithms are proposed for a TTRPTW variant in which a trailer is not strictly assigned to a single truck, meaning it can be pulled by different trucks along the route. Also, there can be load transfers from different vehicles to a trailer, and transfer times depend on the load. Belenguer et al. [2016] also propose a BC algorithm, this time for a Single TTRP with Satellite Depots, in which there is only one vehicle available and only truck customers to be serviced, and satellite depots can be used to park the trailer.

Very recently, Bartolini and Schneider [2020] introduced a flow formulation and a BC algorithm for a TTRP variant without time windows, considering scenarios in which load transfers are allowed, as well as scenarios where they are forbidden. The authors also use their method to handle the Single TTRP with Satellite Depots, as in [Belenguer et al., 2016].

The TTRP variants solved by the works mentioned in this section are defined by the presence or not of some key characteristics, that may depend on the particular real life application that inspired that work, and on the limitations of the model that is employed. For example, in [Drexl, 2014] and [Rothenbächer et al., 2018], the time taken by a load transfer operation depends on the amount of load to be transferred. In the other exact approaches listed here, load transfer times are treated as constant or ignored. Table 2.2 summarizes some of these characteristics of the TTRP variants treated by each of the exact approaches that we mentioned, as well as the model in-

troduced in this thesis. A horizontal line separates the three works that used a BP algorithm (above the line) from the ones that used a BC method (below). In each column of this table, a modeling characteristic is represented, and a check mark symbol indicates that the characteristic is considered by the corresponding work. The first column represents the possibility of a heterogeneous fleet in the instance, that is, if the model considers vehicles with possibly distinct capacities. The second column represents the possibility of considering different speed values for each vehicle in the fleet. The third column indicates if a solution for the considered TTRP variation consists of multiple routes, for multiple vehicles in the fleet. The fourth column indicates the presence of time windows for customers. The fifth column corresponds to the existence of dedicated satellite depots among the vertices in an instance of the problem. The sixth column represents the possibility for a satellite depot to be used as the starting point of multiple subtours in a solution route. Finally, the last column of the table indicates if the model considers the amount of load to be transfered in the calculation of load transfer times.

Table 2.2: Previous works that handle TTRP variations with exact approaches and comparison of modeling characteristics considered in each one.

	Heterogeneous fleet	Distinct vehicle speeds	Multiple routes	Time windows	Dedicated satellite depots	Multiple subtours per vertex	Load-dependent transfer times
(1)	✓		✓	✓	✓	✓	
(2)			✓	✓		✓	
(3)	✓		✓	✓	✓	✓	✓
(4)	✓		✓	✓	✓		✓
(5)					✓	✓	
(6)			✓		✓	✓	
(7)	✓	✓		✓	✓	✓	✓

Works referred by each row of this table:

(1) Drexl [2011]

(2) Parragh and Cordeau [2017]

(3) Rothenbächer et al. [2018]

(4) Drexl [2014]

(5) Belenguer et al. [2016]

(6) Bartolini and Schneider [2020]

(7) This thesis

The model developed in this work considers different speed values for vehicles in the fleet, as well as a decrease in truck speed when it is coupled to a trailer. To our knowledge, this has not been handled by exact approaches previously introduced for the TTRP. In general, an existing BP model could be adapted for this aspect, since time windows are usually tackled by the DP algorithm. For an existing BC approach, however, this would require significant changes in the formulation.

Our model also allows for a truck to perform any number of subtours from each satellite depot. From our research in the literature, among exact BC approaches, no previous model has considered that aspect for a TTRPTW variation. The model introduced in [Drexl, 2014] for a TTRPTW variation also has a polynomial number of variables; however, each truck can only perform one subtour at each satellite vertex. Belenguer et al. [2016] and Bartolini and Schneider [2020] also solve TTRPs with models from the same category. In those works, the number of subtours departing from the same satellite depot is not restricted, but they do not consider customer time windows. When that aspect is dropped, the model does not need to handle the satellite scheduling, i.e., the order at which satellite vertices are visited, which allows for it to be considerably simpler.

The PSTTRPTW is a variation of the TTRPTW that involves profits. Actually, it is possible to define generalized versions with profits for the TSP [Feillet et al., 2005] and for any vehicle routing problem variation. In such versions, it is not necessary to visit all customers, and a profit is associated to each of them. There is, then, a trade-off relationship between the arc travel costs and the customer service profits. This relationship can be tackled with different *optimization senses*, giving rise to distinct variations of routing problems involving profits. For example:

1. In a *prize collecting* variation of a routing problem, the goal is to find a minimum cost set of routes with profit not smaller than a given predefined value. Some works that handle prize collecting vehicle routing problems also consider a penalty value in the objective function associated to non-visited customers, like in the original work regarding the Prize Collecting TSP [Balas, 1989].

2. In an *Orienteering* Problem [Chao et al., 1996], the aim is to find a set of routes with maximum profit, while the total travel cost cannot exceed a given budget.

3. If the goal is to find a set of routes that maximizes the collected profit minus the travel cost, the routing problem variation often receives the designation *profitable*.

Profitable Vehicle Routing Problems, besides having a clear economical meaning on their own, may be thought as subproblems in column generation schemes for solving

more general vehicle routing problems. In these subproblems, customer profits typically originate from dual variables associated to constraints that impose the customer service in the master problem. From our research in the literature, all BP approaches for multiple vehicle Truck-and-Trailer Routing Problems resort on DP algorithms to solve such pricing subproblems. Our goal is also to contribute with an approach based on Integer Programming (IP) techniques. The first step in this direction is the presentation of an IP formulation for the PSTTRPTW. This subject is the theme of the next chapter.

Chapter 3

An Integer Programming Formulation and Valid Inequalities for the PSTTRPTW

In this chapter, we introduce an Integer Programming formulation for the PSTTRPTW. The formulation makes use of an auxiliary graph, created from the problem input graph. Before presenting the formulation, the construction of this graph is described. We also highlight some properties obtained with the auxiliary graph that are explored in the formulation. A proof for the formulation correctness is also provided, as well as preprocessing procedures and valid inequalities, designed to reinforce the formulation's LPR bounds.

3.1 Auxiliary Graph Creation

Given an instance for the PSTTRPTW, we create an *auxiliary graph* $\bar{G} = (\bar{V}, \bar{A})$. One of the main benefits we look for with this transformation is to separate the three roles a satellite vertex can play in the problem description: decoupling, load transfer and coupling. The transformation makes these three operations easier to express through linear constraints in the model. Another purpose of using $\bar{G}$ is to help translating some structural restrictions imposed by the problem definition, for instance, that only single trucks can reach or leave truck customers, or that only complete vehicles can arrive at a satellite depot to decouple a trailer.

Let $G = (V, A)$ be the input graph for the problem. The vertex set of the auxiliary graph $\bar{G}$ is given by $\bar{V} = \{0\} \cup C \cup \bar{S}$, where, for each satellite vertex $i \in S$, we create

three *specialized satellites* in $\bar{S}$: i_d, i_t and i_c. i_d is the *decoupling satellite* associated to i, whereas i_t is the *load transfer satellite*, and i_c, the *coupling satellite*. The specialized satellite sets S_d, S_t and S_c, that partition $\bar{S}$, are defined as

$$S_d = \{i_d : i \in S\},$$
$$S_t = \{i_t : i \in S\},$$
$$S_c = \{i_c : i \in S\}.$$

Each of the specialized satellite vertices is used to represent a specific role that a satellite depot can perform. A complete vehicle visiting a vertex $i_d \in S_d$ in a route in $\bar{G}$ represents a decoupling operation, after which that vehicle leaves i_d as a single truck. When a truck visits vertex $i_t \in S_t$, this represents a load transfer operation performed in the original satellite $i \in S$. Likewise, a visit to $i_c \in S_c$ by a single truck corresponds to a coupling operation at i, after which the vehicle leaves this satellite depot with the trailer attached. A similar transformation was described in [Drexl, 2014].

When two vertices of $\bar{S}$ playing different roles have the same original satellite in S, we will refer to them as *related*. If two satellites are related, we will always use the same letter to reference them. For example, if, in a context, we are denoting a decoupling satellite as i_d, and refer to a satellite as i_c, we mean the coupling satellite that was created from the same original satellite as i_d, that is, $i \in S$.

This specialization of satellites allows for graph $\bar{G}$ not being complete, unlike the input graph G. Given two vertices $i, j \in \bar{V}$, with $i \neq j$, arc (i, j) is *not* included in the auxiliary graph in the following cases:

1. $i = 0$ and $j \in S_t \cup S_c$. A vehicle cannot transfer load or couple a trailer right after leaving the depot (it should decouple first).

2. $i \in C_{\mathcal{K}}$ and $j \in S_d$. If a vehicle leaves a truck only customer, it has to be a single truck. So, it cannot perform a decoupling operation right after.

3. $i \in S_d \cup S_t$ and $j \in \{0\} \cup \bar{S}$. If a truck has just decoupled or transferred load in a satellite depot, it cannot return to the main depot, since it has to recouple first. It also should not immediately visit another satellite vertex without visiting a customer first, as this is a suboptimal choice. Since there is always an optimal solution in which such arcs are not needed, they are not created.

4. $i \in S_c$ and $j \in C_{\mathcal{K}} \cup S_t \cup S_c$. If a vehicle has just coupled a trailer, it is not allowed to visit a truck customer, since it has a trailer attached. It also cannot transfer load or couple again.

5. $i = v_c$ and $j = v_d$, for any $v \in S$. This is another suboptimal arc. If a vehicle has just coupled at v, it should not decouple again at the same satellite depot. In such a situation, it is always preferable to only perform a load transfer at v.

For all pairs of vertices $i, j \in \bar{V}$ not matching any excluding item above, an arc (i, j) is included in $\bar{A}$.

Besides the cases above, there are also situations in which an arc can never be used by a single truck, and others in which it cannot be used by a complete vehicle. In such cases, the arc is still included in $\bar{A}$, and we describe in the next section how to prevent it from being used by forbidden vehicles. We now list every situation that forbids an arc from being traversed by a specific type of vehicle.

An arc $(i, j) \in \bar{A}$ cannot be used by a single truck in the two following cases:

1. $i \in S_c$. A vehicle that leaves a coupling satellite should be coupled.

2. $j \in S_d$. A single truck can never arrive at a decoupling satellite, since it is already decoupled.

Conversely, arc $(i, j) \in \bar{A}$ cannot be used by a complete vehicle in the following cases:

1. $i \in C_{\mathcal{K}} \cup S_d \cup S_t$. If a vehicle is leaving a truck customer, a decoupling satellite, or a load transfer satellite, it should not be pushing a trailer.

2. $j \in C_{\mathcal{K}} \cup S_t \cup S_c$. For similar reasons as above.

Table 3.1 summarizes the arc usage rules introduced so far. For $i, j \in \bar{V}$, each row in the table represents a possibility for i, whereas the central columns correspond to a possibility for j. A dash means the arc is not created. Symbols "$\{0\}$" and "$\mathcal{L}$" mean the arc is created, but can only be used by single trucks and by complete vehicles, respectively. For the case $i \in S_c, j \in S_d$, the arc is created only if i and j are not related, as explained earlier. Finally, "$\{0\} \cup \mathcal{L}$" means the arc can be used by any type of vehicle.

As the PSTTRPTW definition states, each vehicle has its own travel cost. Hence, arcs have an associated cost for each vehicle. As defined in Section 2.1, the cost associated to arc $(i, j) \in \bar{A}$, for vehicle $(k, l) \in \bar{V}$, is given by $c_{ij}^{kl} = d_{ij}\phi_{kl}$.

We now define the time associated to an arc. Like for the costs, time-related parameters like speed and coupling time also depend on the vehicle. Therefore, time t_{ij}^{kl} associated to an arc (i, j) is also defined for each vehicle (k, l). Besides the travel time

Table 3.1: Rules for creation of arc (i, j) and its usage by vehicles in the auxiliary graph, and time associated to (i, j) for vehicle (k, l).

i \ j	$\{0\}$	$C_{\mathcal{V}}$	$C_{\mathcal{K}}$	S_d	S_t	S_c	t_{ij}^{kl}
$\{0\}$	-	$\{0\} \cup \mathcal{L}$	$\{0\}$	$\mathcal{L}$	-	-	$\frac{d_{ij}}{s_{kl}}$
$C_{\mathcal{V}}$	$\{0\} \cup \mathcal{L}$	$\{0\} \cup \mathcal{L}$	$\{0\}$	$\mathcal{L}$	$\{0\}$	$\{0\}$	$\frac{d_{ij}}{s_{kl}} + \frac{q_i}{o_k}$
$C_{\mathcal{K}}$	$\{0\}$	$\{0\}$	$\{0\}$	-	$\{0\}$	$\{0\}$	$\frac{d_{ij}}{s_{kl}} + \frac{q_i}{o_k}$
S_d	-	$\{0\}$	$\{0\}$	-	-	-	$\frac{d_{ij}}{s_{kl}} + \tau_{kl}$
S_t	-	$\{0\}$	$\{0\}$	-	-	-	$\frac{d_{ij}}{s_{kl}}$
S_c	$\mathcal{L}$	$\mathcal{L}$	-	$\mathcal{L}^*$	-	-	$\frac{d_{ij}}{s_{kl}} + \tau_{kl}$

* for $i \in S_c$ and $j \in S_d$, (i, j) is created only if i and j are not related. In this case, (i, j) can only be used by a complete vehicle.

$\frac{d_{ij}}{s_{kl}}$, t_{ij}^{kl} also considers time-consuming operations that take place at vertex i, namely, customer service, coupling and decoupling, depending on whether i is a customer or a specialized satellite depot. The last column of Table 3.1 shows the value of t_{ij}^{kl} for arc (i, j), for each possible type that endpoint i may assume. If vehicle (k, l) is not allowed to use arc (i, j), we define $t_{ij}^{kl} := \infty$. Note that there are still two types of time-consuming events that are not considered by t_{ij}^{kl}: the waiting when a vehicle arrives at a customer before its time window opening, and the load transfer after a subtour. The time taken by these events depend on vertices previously visited, and therefore must be dynamically calculated, as seen later in this chapter.

3.2 Properties of the PSTTRPTW

In this section, we present properties that hold for a feasible solution for the PSTTRPTW, arisen from the problem definition and the rules for the auxiliary graph creation. These properties are explored in the formulation and algorithm for the PSTTRPTW, described later on. First, we introduce the definitions and the notation that will be used throughout this thesis. For the remaining of this section, consider a given instance, for which an auxiliary graph $\bar{G} = (\bar{V}, \bar{A})$ was created.

Definition 1. A *walk* in $\bar{G}$ is defined as a finite sequence $\Gamma = (\gamma_1, \ldots, \gamma_{|\Gamma|})$ of adjacent vertices in $\bar{G}$, possibly including repeated vertices and arcs.

Despite the usual definition of a walk in a graph using a sequence of arcs, in order to simplify notation, we use a different, but equivalent, definition. The capital letter Γ will be used to denote walks, and γ_i (in lower case) will be used to reference the

i-th vertex in Γ. The length $|\Gamma|$ of a walk is the number of elements in the sequence, which counts possibly repeated vertices that appear in the walk. We use s_i in indexes to reference the i-th satellite vertex that appears in the walk in context. For example, $\gamma_{s_1} \in \bar{S}$ is the first satellite that appears in Γ. $C(\Gamma)$ is defined as the set of customers that appear in Γ. Also, $s(\Gamma)$ is the number of times satellite vertices appear in Γ, counting possible repeated entries. A subtour in Γ is any portion of Γ that starts at a decoupling or load transfer satellite $i \in S_d \cup S_t$, ends at the related load transfer or coupling satellite (i_t or i_c, respectively), and contains only customer vertices in between. A subtour of Γ that ends at satellite vertex i will be denoted as $\hat{\Gamma}(i)$.

A vehicle in $\mathcal{V}$ can be assigned to traverse a walk. This assignment represents a choice of vehicle and route in our problem. The following definition reflects the problem condition by which a complete vehicle is decoupled after visiting a decoupling or load transfer satellite.

Definition 2. Consider that a vehicle $(k, l) \in \mathcal{V}$ was assigned to a walk Γ. For any γ_i,

- If $l = 0$, or if the last satellite vertex in Γ before γ_i is a decoupling or load transfer satellite, we say (k, l) is *decoupled* at γ_i.
- Otherwise, that is, if (k, l) is a complete vehicle and the last satellite vertex in Γ before γ_i is a coupling satellite, or if no satellite appears before γ_i, we say (k, l) is *coupled* at γ_i.

We now define two important quantities associated to a walk in $\bar{G}$.

Definition 3. The *total supply of a walk* Γ in $\bar{G}$ is given by the sum of customer supplies in Γ, that is,

$$q(\Gamma) := \sum_{i \in C(\Gamma)} q_i.$$

Definition 4. Consider a walk Γ and a vehicle $(k, l) \in \mathcal{V}$, assigned to Γ. The *travel time* t^i_{kl} that vehicle (k, l) takes to reach the i-th element of Γ is given by the following recursion:

$$t^1_{kl}(\Gamma) := e_{\gamma_1}$$

$$t^i_{kl}(\Gamma) := \begin{cases} \max\{t^{i-1}_{kl}(\Gamma) + \Delta_i, e_{\gamma_i}\}, & \text{if } \gamma_i \in C \\ t^{i-1}_{kl}(\Gamma) + \Delta_i, & \text{if } \gamma_i \in \{0\} \cup S_d \\ t^{i-1}_{kl}(\Gamma) + \Delta_i + \dfrac{q(\hat{\Gamma}(\gamma_i))}{o_k}, & \text{if } \gamma_i \in S_t \cup S_c \end{cases} \qquad i = 2, \ldots, |\Gamma|$$

where the quantity Δ_i is given by

$$\Delta_i = \begin{cases} t^{kl}_{\gamma_{i-1}\gamma_i}, \text{if } (k,l) \text{ is coupled at } \gamma_i \\ t^{k0}_{\gamma_{i-1}\gamma_i}, \text{if } (k,l) \text{ is decoupled at } \gamma_i \end{cases}$$

Also, we define the travel time of Γ for vehicle (k,l) as

$$t_{kl}(\Gamma) := t^{|\Gamma|}_{kl}(\Gamma)$$

Note that the travel time definitions consider the coupling state of the vehicle at any vertex of Γ, as well as the possibility that the vehicle may have to wait for a customer time window to open, if it arrives too early at it. Also, after finishing a subtour, at a load transfer or coupling satellite, the load transfer time is considered.

In the sequence, we define three desirable properties for a walk, that will be used afterwards to characterize feasible solutions.

Definition 5. Given a walk Γ and a vehicle (k,l) assigned to Γ, we say this assignment is *time window consistent* if, for every $\gamma_i \in C(\Gamma)$,

$$t^i_{kl}(\Gamma) \leq f_{\gamma_i}$$

That is, vehicle (k,l) is able to arrive at every customer in Γ before its time window closing.

Definition 6. Given a walk Γ and a vehicle (k,l) assigned to Γ, we say this assignment is *capacity consistent* if:

i) $q(\Gamma) \leq Q_k + Q_l$, and

ii) $q(\hat{\Gamma}(\gamma_{s_i})) \leq Q_k$, for every $i = 1, \ldots, s(\Gamma)$ such that $\gamma_{s_i} \in S_t \cup S_c$.

That is, the total supply to be collected along Γ does not exceed the capacity of vehicle (k,l), and the total supply in subtours does not exceed the truck capacity.

Definition 7. We say that a walk Γ is *satellite consistent* if Γ contains no satellite vertices, or, if it does,

i) $\gamma_{s_1} \in S_d$

ii) If $\gamma_{s_k} = i \in S_d \cup S_t$, then $\gamma_{s_{k+1}} \in \{i_t, i_c\}$, for $k = 1, \ldots, s(\Gamma) - 1$.

iii) If $\gamma_{s_k} \in S_c$, then $\gamma_{s_{k+1}} \in S_d$, for $k = 2, \ldots, s(\Gamma) - 2$.

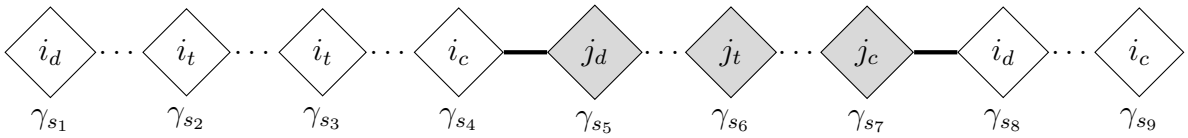


Figure 3.1: Example of a satellite consistent walk.
Customers visited between satellites are omitted. Dotted lines between satellites represent subtours, and strong, continuous lines are portions of the walk that must be traveled by a complete vehicle.

iv) $\gamma_{s_k} \in S_c$ for $k = s(\Gamma)$.

This definition translates the ordering that satellite vertices in a feasible solution must follow: the first satellite must be a decoupling one; after that, the related load transfer satellite can be visited zero or more times, until the walk visits the related coupling satellite. Then, another decoupling satellite can be visited, and the same pattern repeats until the walk visits the last coupling satellite. This property is illustrated by Figure 3.1, in which a walk visiting nine satellite vertices is represented. For simplicity, customers visited between satellites are not shown. Dotted lines between two satellites represent subtours, and continuous lines are portions of the walk that must be traveled by a complete vehicle.

The next statement about a feasible solution for the PSTTRPTW follows directly from the problem description and the definitions above.

Remark 1. *A feasible solution for the PSTTRPTW consists of a choice of a walk* Γ *in* $\bar{G}$*, referred to as a route, and a vehicle* $(k, l) \in \mathcal{V}$*, assigned to* Γ*, for which the following holds:*

i) Γ *starts and ends at vertex* $0 \in \bar{V}$*, and visits each customer in* C *at most once;*

ii) Γ *is time window consistent for* (k, l)*;*

iii) Γ *is capacity consistent for* (k, l)*;*

iv) Γ *is satellite consistent.*

Now, we introduce results that allow to establish bounds on the number of times certain vertices and arcs in $\bar{G}$ can be visited. Let $W \subseteq C$ be a subset of customers for which $\sum_{i \in W} q_i \leq \max\{Q_k + Q_l : (k, l) \in \mathcal{K} \times \mathcal{L}\}$, that is, W is a set of customers whose total supply does not exceed the capacity of at least one vehicle in $\mathcal{V}$. Let $\bar{c}$ be the maximum cardinality of a possible set W. That is, $\bar{c}$ is the Dantzig bound [Dantzig, 1957] for a Knapsack Problem in which we aim to select a subset of customers whose

total supply fit a vehicle of maximum capacity. Quantity $\bar{c}$ is a bound for the number of customers that can be visited by a vehicle in a feasible route. It can be calculated by sorting the customers in C by supply in ascending order and counting the number of customers until the total supply exceeds the biggest vehicle capacity in $\mathcal{V}$.

While the depot and customer vertices are visited at most once in a feasible solution for the problem, satellite vertices can be visited multiple times. The satellite consistency property allows to establish bounds for the number of times satellite vertices and certain arcs in $\bar{G}$ are used.

Proposition 1. *The number of times a trailer is decoupled (or coupled) in an optimal solution is at most* $\bar{c}$.

Proof. Consider an optimal solution that selects a complete vehicle. For each time the selected trailer is decoupled, the truck visits at least one customer. Because of the satellite consistency, the truck also does not visit a different satellite vertex until re-coupling at the satellite where the trailer was parked. Therefore, the number of decoupling operations is equal to the number of couplings, and limited by the number $\bar{c}$ of customers that can be visited in a route. □

Proposition 2. *The number of times a vehicle visits satellite vertices in an optimal solution is at most* $2\bar{c}$.

Proof. Consider a solution that visits $\bar{c}$ customers, and in which the route is built according to the following: for each visited customer, the trailer is decoupled, the customer is serviced, and then the trailer is coupled again, as shown in Figure 3.2. This solution visits satellite vertices $2\bar{c}$ times. This is the maximum number of satellite visits in a feasible solution, since, if a customer is visited while the vehicle is coupled, or if more than one customer are visited between two consecutive decoupling and coupling operations, the number of visited satellite vertices is always smaller than in the solution considered. Also, note that any solution that uses load transfer satellites requires less than $2\bar{c}$ satellite visits. □

Proposition 3. *Let* $(i, j) \in \bar{A}$. *If* $i \in S_c$ *and* $j \in S_d$, *then* (i, j) *can be traversed at most* $\lfloor \frac{\bar{c}}{2} \rfloor$ *times in an optimal solution. Otherwise,* (i, j) *is traversed at most once.*

Proof. If i or j is the depot or a customer vertex, then (i, j) is used at most once, since these vertices cannot be visited more than once. Suppose now that $i, j \in \bar{S}$. By the rules for the auxiliary graph arcs, summarized in Table 3.1, the only arcs connecting satellites that are created are for $i \in S_c$ and $j \in S_d$. We consider again the extreme-case

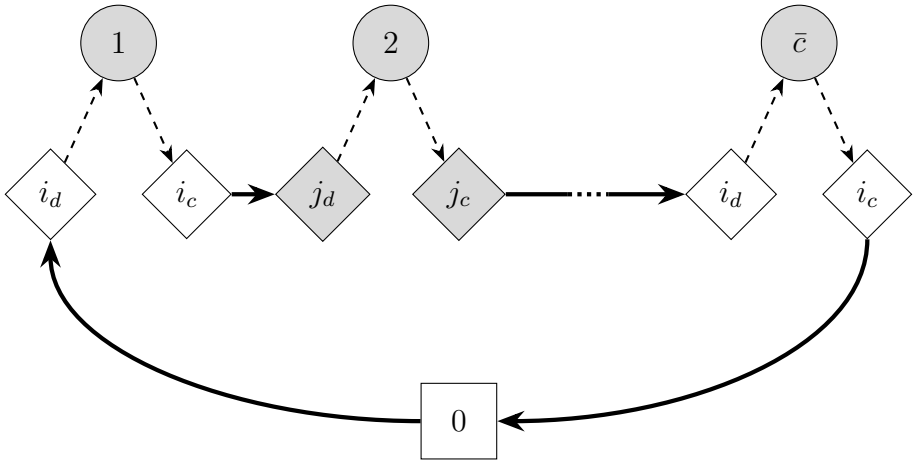


Figure 3.2: Route with the maximum number of satellite visits.

solution that visits each customer between two decoupling and coupling operations, as in Figure 3.2. In the worst case, satellites i and j are alternated in this route. Since at most $\bar{c}$ customers are visited, the number of times this route can go from i to j (and from j to i) is $\lfloor \frac{\bar{c}}{2} \rfloor$. □

3.3 Integer Programming Formulation

Before presenting the formulation, we introduce some definitions and the notation adopted hereafter. Let $\mathbb{Z}$ be the set of integer numbers, and $\mathbb{R}$, the set of real numbers. Given an auxiliary graph $\bar{G} = (\bar{V}, \bar{A})$ and a subset of vertices $W \subseteq \bar{V}$, we define $A(W) := \{(i,j) \in \bar{A} : i,j \in W\}$, $\delta^+(W) := \{(i,j) \in \bar{A} : i \in W, j \notin W\}$, $\delta^-(W) := \delta^+(\bar{V} \setminus W)$ and $\delta(W) := \delta^+(W) \cup \delta^-(W)$. For unitary sets, we simplify the notation by defining $\delta^+(i) := \delta^+(\{i\})$, $\delta^-(i) := \delta^-(\{i\})$ and $\delta(i) := \delta(\{i\})$. If, additionally we have $Y \subseteq \bar{V}$, we define $\delta^+(W:Y) := \delta^+(W) \cap \delta^-(Y)$ and $\delta(W:Y) := \delta^+(W:Y) \cup \delta^+(Y:W)$. $\mathcal{W}(\bar{G})$ is defined as the set of all walks in $\bar{G}$. We also define $n := |\bar{V}|$ and $m := |\bar{A}|$.

We consider the following integer decision variables:

- x_{ij} : number of times arc $(i,j) \in \bar{A}$ is traversed in the solution
- $y_i = \begin{cases} 1, \text{if vertex } i \in \bar{V} \text{ is visited in the solution} \\ 0, \text{otherwise} \end{cases}$
- $z_{k0} = \begin{cases} 1, \text{if truck } k \in \mathcal{K} \text{ is used in the solution (with or without a trailer)} \\ 0, \text{otherwise} \end{cases}$

- $z_{kl} = \begin{cases} 1, \text{if truck } k \in \mathcal{K} \text{ is used in the solution with trailer } l \in \mathcal{L} \\ 0, \text{otherwise} \end{cases}$

- $w_{ij}^{k0} = \begin{cases} 1, \text{if arc } (i,j) \in \bar{A} \text{ is traversed by truck } k \in \mathcal{K} \text{ without a trailer} \\ 0, \text{otherwise} \end{cases}$

- w_{ij}^{kl} : number of times arc $(i,j) \in \bar{A}$ is traversed by truck $k \in \mathcal{K}$ with trailer $l \in \mathcal{L}$

- $r_{ij}^{\kappa} = \begin{cases} 1, \text{ if } (i,j) \in \delta^+(\bar{S}) \text{ is traversed after leaving a satellite depot for the } \kappa\text{-th} \\ \quad \text{time, if } \kappa \text{ or more satellites are used in the solution, for } \kappa = 1, \ldots, 2\bar{c} \\ 0, \text{otherwise} \end{cases}$

Variable r is indexed both by the arc (i,j) leaving a satellite depot, and by the number of times κ a satellite depot is visited in the solution. By Proposition 2, index κ can be limited by $2\bar{c}$ for this set of variables.

Let $H \subseteq \bar{A}$ be a set of arcs in $\bar{G}$ and $\mathcal{S} \subseteq \mathcal{V}$, a set of vehicles. In order to shorten the presentation, we define the following real-valued sums:

$$x(H) := \sum_{(i,j)\in H} x_{ij}$$
$$w^{kl}(H) := \sum_{(i,j)\in H} w_{ij}^{kl}$$
$$w_{ij}(\mathcal{S}) := \sum_{(k,l)\in \mathcal{S}} w_{ij}^{kl}$$

where x and w are the vectors of variables introduced in this section. Additionally, if $H \subseteq \delta^+(\bar{S})$ and $\kappa = 1, \ldots, 2\bar{c}$,

$$r^{\kappa}(H) := \sum_{(i,j)\in H} r_{ij}^{\kappa}$$

where r is the vector of decision variables previously introduced.

For variable x, the short hand $x(\Gamma) := \sum_{i=1}^{|\Gamma|-1} x_{\gamma_i\gamma_{i+1}}$ is the sum of that variable for all arcs in Γ. If $W \subseteq \bar{V}$, the notation $x(\Gamma|W)$ represents the sum of x restricted to arcs in Γ that belong to $\delta^+(W)$, that is:

$$x(\Gamma|W) := x(\{(\gamma_i, \gamma_{i+1}) : \gamma_i \in W\})$$

Our IP Formulation for the PSTTRPTW is given by:

$$\max \left\{ \sum_{i \in C} p_i y_i - \sum_{(k,l) \in \mathcal{V}} a_{kl} z_{kl} - \sum_{(i,j) \in \bar{A}} \sum_{(k,l) \in \mathcal{V}} c_{ij}^{kl} w_{ij}^{kl} : (x, y, z, w, r) \in \mathcal{P} \cap \mathbb{Z}^{\sigma} \right\}, \quad (3.1)$$

where σ is the total number of variables, and $\mathcal{P} \subset \mathbb{R}^{\sigma}$ is the polyhedron defined by the linear constraints (3.2)-(3.16), introduced in the sequence.

Connectivity constraints

$$x_{ij} = w_{ij}(\mathcal{V}) \qquad (i,j) \in \bar{A} \qquad (3.2)$$

$$x(\delta^+(i)) = x(\delta^-(i)) \qquad i \in \bar{V} \qquad (3.3)$$

$$x(\delta^+(W)) \geq y_i \qquad W \subseteq \bar{V} \setminus \{0\}, i \in W \qquad (3.4)$$

Constraints (3.2) link variables x and w, ensuring that the number of times an arc is used is exactly the number of times a vehicle traverses it. Flow balance constraints (3.3) establish that, the number of times a vehicle arrives at a vertex is exactly the number of times it leaves that vertex. Inequalities (3.4) are the Directed Cutset Constraints (DCUTs) [Chopra et al., 1992]. They ensure that a feasible solution must be connected and arrive at the depot vertex, whenever a nonempty route is implemented by a vehicle, i.e., a non-trivial solution is chosen.

$$x(\delta^+(0)) \leq 1 \qquad (3.5a)$$

$$x(\delta^+(i)) = y_i \qquad i \in C \qquad (3.5b)$$

Inequalities (3.5a) establish that a route can leave the depot at most once, and (3.5b), that the vehicle leaves a customer vertex if and only if it is visited.

$$y_i \geq x_{ij} \qquad (i,j) \in \delta^+(\bar{V} \setminus S_c) \qquad (3.6a)$$

$$y_i \geq x_{ji} \qquad (j,i) \in \delta^-(S_c) \qquad (3.6b)$$

Constraints (3.6) establish that, if an arc is traversed in the solution, its endpoints must be used. Since arcs of the type $(i_c, j_d) : i \in S_c, j \in S_d$ can be traversed

multiple times, (3.6a) might not hold for them. For these cases, constraints (3.6b) are used instead.

Total vehicle capacity constraint

$$\sum_{i \in C} q_i y_i \leq \sum_{k \in \mathcal{K}} Q_k z_{k0} + \sum_{(k,l) \in \mathcal{V}} Q_l z_{kl} \tag{3.7}$$

Constraint (3.7) ensures that the total supply collected in a feasible route does not exceed the capacity of the selected vehicle, be it a single truck or a complete vehicle.

Truck selection constraint

$$\sum_{k \in K} z_{k0} \leq 1 \tag{3.8}$$

By constraint (3.8), at most one truck is selected in a feasible solution. Note that a "trivial" solution, in which no vehicle is selected and no vertex is visited, is allowed and has objective value equal to zero.

Arc-vehicle assignment constraints

$$w^{k0}(\delta^+(0)) = z_{k0} - \sum_{l \in \mathcal{L}} z_{kl} \qquad k \in \mathcal{K} \tag{3.9a}$$

$$w^{kl}(\delta^+(0)) = z_{kl} \qquad k \in \mathcal{K}, l \in \mathcal{L} \tag{3.9b}$$

$$w^{k0}(\delta^+(i)) \leq z_{k0} \qquad i \in C, k \in \mathcal{K} \tag{3.9c}$$

$$w^{kl}(\delta^+(i)) \leq z_{kl} \qquad i \in C_{\mathcal{V}}, k \in \mathcal{K}, l \in \mathcal{L} \tag{3.9d}$$

$$w^{kl}_{ji} \leq z_{kl} \qquad i \in S_d, j \notin S_c, (j,i) \in \delta^-(i), k \in \mathcal{K}, l \in \mathcal{L} \tag{3.9e}$$

$$w^{kl}_{ij} \leq z_{kl} \qquad i \in S_c, j \notin S_d, (i,j) \in \delta^+(i), k \in \mathcal{K}, l \in \mathcal{L} \tag{3.9f}$$

$$w^{kl}(\delta^+(S_c)) \leq \bar{c} z_{kl} \qquad k \in \mathcal{K}, l \in \mathcal{L} \tag{3.9g}$$

Constraints (3.9) determine which vehicles are allowed to traverse each type of arc, according to the selected vehicle. By (3.9a), a single truck $(k, 0)$ can leave the depot if, and only if, truck k is selected and no trailer is selected together with k. At the same time, this constraint also enforces that, if k is selected, at most one trailer can be selected to be used coupled with k. If $z_{k0} = 1$ and $\sum_{l \in \mathcal{L}} z_{kl} = 0$, the corresponding solution is a truck-only route. Constraints (3.9b) establish that, if the complete vehicle

(k, l) is selected in the solution, it must leave the depot vertex.

A customer can be visited by single truck k only if this truck is selected (constraints (3.9c)), and vehicle customers can be visited by the complete vehicle (k, l) only if this vehicle is selected (constraints (3.9d)). For arcs that arrive at decoupling satellites or leave coupling satellites, vehicle (k, l) is allowed only if it is selected, as determined by (3.9e)-(3.9g). Constraints (3.9e) and (3.9f) consider arcs that can only be used once, while (3.9g) also considers arcs that can be visited multiple times, using the bound expressed in Proposition 1. Note that it is not necessary to have a similar constraint for arcs adjacent to transfer satellites, because these vertices are only visited by single trucks.

Trailer consistency constraints

$$w^{k0}(\delta^+(i)) = w^{k0}(\delta^-(i)) \qquad i \in \{0\} \cup C_{\mathcal{V}}, k \in \mathcal{K} \tag{3.10}$$

Constraints (3.10) express the fact that, if i is the depot or a vehicle customer vertex, the truck that arrives at i must be the same that leaves this vertex. Together with (3.2) and (3.9), these constraints also ensure that, if a complete vehicle arrives at i, the same truck and trailer pair must leave i.

Satellite consistency constraints

$$\sum_{\kappa=1}^{2\bar{c}} r_{ij}^{\kappa} = x_{ij} \qquad (i, j) \in \delta^+(\bar{S}) \tag{3.11}$$

Constraints (3.11) link variables x and r, establishing that, for each time an arc that leaves a satellite vertex is used, a variable r for this arc is active for some index κ.

$$r^1(\delta^+(S_t \cup S_c)) = 0 \tag{3.12a}$$

$$r^1(\delta^+(S_d)) \leq 1 \tag{3.12b}$$

$$r^{\kappa}(\delta^+(i_t) \cup \delta^+(i_c)) \leq r^{\kappa-1}(\delta^+(i_d) \cup \delta^+(i_t)) \qquad i \in S, \kappa = 2, \ldots, 2\bar{c} \tag{3.12c}$$

$$r^{\kappa}(\delta^+(S_d)) \leq r^{\kappa-1}(\delta^+(S_c)) \qquad \kappa = 2, \ldots, 2\bar{c} \tag{3.12d}$$

Constraints (3.12) enforce that the satellite vertices corresponding to the r

variables set to 1 provide a sequence, considering index κ, that is consistent with the coupling rules given by the problem, as in Figure 3.1. More precisely, by (3.12a), the first satellite vertex to be visited in a route can be neither a transfer nor a coupling satellite. Inequality (3.12b) establishes that at most one decoupling satellite is assigned to index $\kappa = 1$. For $\kappa \geq 2$, a transfer or coupling satellite can be assigned to index κ only if the related decoupling or transfer satellite is assigned to $\kappa - 1$ (constraints (3.12c)), and a decoupling satellite can be assigned to index κ only if a coupling satellite is assigned to $\kappa - 1$ (constraints (3.12d)). Note that this set of constraints ensures that at most one arc is assigned to each κ, and that an arc is assigned to index κ only if some arc is assigned for indexes $1, \ldots, \kappa-1$.

Subtour capacity constraints

$$x(\Gamma) \leq |\Gamma| - 1 - z_{k0} \quad \Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 \in S_d \cup S_t \\ \gamma_2, \ldots, \gamma_{|\Gamma|} \in C \end{cases}, k \in \mathcal{K} : Q_k < q(\Gamma) \quad (3.13)$$

Constraints (3.13) consider walks that start at decoupling or load transfer and then visit only customers. If the selected truck capacity is smaller than the supply of the customers visited in a given walk, then the corresponding solution is forbidden. Therefore, these constraints ensure that the supply collected in a subtour does not exceed the selected truck capacity.

Time window constraints

$$x(\Gamma|\{0\} \cup C) + \sum_{i=1}^{s(\Gamma)} r^i_{\gamma_{s_i}\gamma_{s_i+1}} \leq |\Gamma| - 1 - z_{kl} \quad (3.14)$$

$$\Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 = 0 \\ \gamma_{|\Gamma|} \in C \end{cases}, (k,l) \in \mathcal{V} : t_{kl}(\Gamma) > f_{\gamma_{|\Gamma|}}$$

Constraints (3.14) guarantee that the time windows of customers visited in a feasible solution are observed. If a given solution uses vehicle (k, l) and a route that begins with walk Γ, and (k, l) is unable to reach the end of Γ before the time window of customer $\gamma_{|\Gamma|}$ closes, then this solution must be forbidden. Note that, regardless of Γ including or not satellite vertices, the sum in the left side has $|\Gamma|$ terms, and this constraint enforces that (k, l) and Γ are not simultaneously picked in a feasible solution.

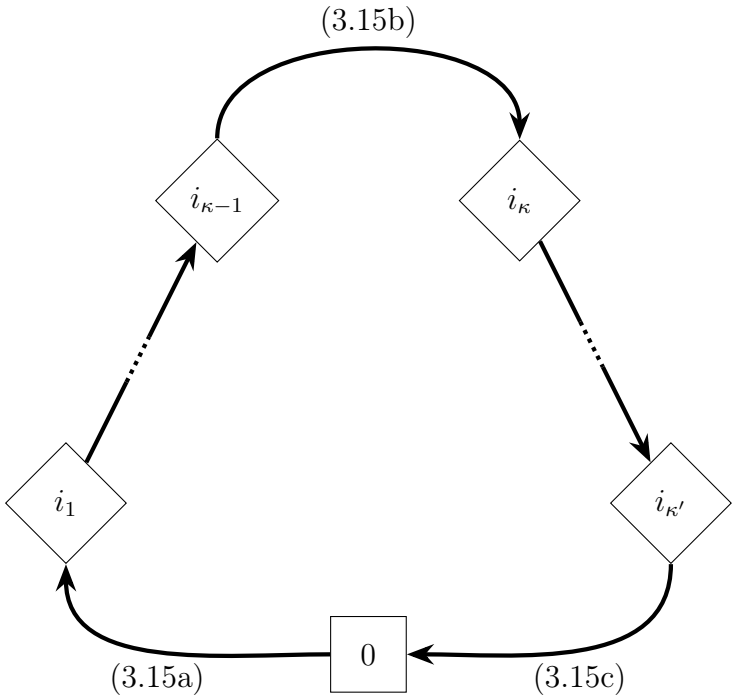


Figure 3.3: Satellite scheduling constraints.

Satellite scheduling constraints

$$x(\Gamma) \leq r^1(\delta^+(\gamma_{|\Gamma|})) + |\Gamma| - 2 \quad \Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 = 0, \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C, \\ \gamma_{|\Gamma|} \in S_d \end{cases} \tag{3.15a}$$

$$r^{\kappa-1}_{\gamma_1\gamma_2} + x(\Gamma|C) \leq r^\kappa(\delta^+(\gamma_{|\Gamma|})) + |\Gamma| - 2 \quad \Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 \in S, \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C, \\ \gamma_{|\Gamma|} \in S \end{cases} \tag{3.15b}$$

$$\kappa = 2, \ldots, 2\bar{c}$$

$$r^\kappa_{\gamma_1\gamma_2} + x(\Gamma|C) + r^{\kappa+1}(\delta^+(\bar{S})) \leq |\Gamma| - 1 \quad \Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 \in S_c, \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C, \\ \gamma_{|\Gamma|} = 0 \end{cases} \tag{3.15c}$$

$$\kappa = 1, \ldots, 2\bar{c} - 1$$

Inequalities (3.15) aim to ensure that the satellite sequence given by variables r

is compatible with the route defined by variables x. Figure 3.3 shows a generic route that visits κ' satellite vertices. Customers visited by this route are not displayed, but possibly appear in the route in portions represented by arrows. Constraints (3.15a) consider walks Γ with the following pattern: Γ starts at the depot vertex, visits any number of customers and finishes at a decoupling satellite i_1. In the figure, this walk corresponds to the portion of the route between 0 and i_1. This set of constraints states that, since i_1 is the first satellite vertex in the route, if the arcs in Γ are selected in a solution, we must have $r^1_{i_1 j} = 1$ for some $(i_1, j) \in \delta^+(i_1)$.

Consider now any portion Γ of a route starting at a satellite $i_{\kappa-1}$, then visiting zero or more customers and ending at another satellite vertex i_κ, like in the portion of the route of Figure 3.3 between $i_{\kappa-1}$ and i_κ. Constraints (3.15b) establish that, if the arcs in Γ are selected by a solution, and $r_{i_{\kappa-1} j} = 1$ for some j (that is, $i_{\kappa-1}$ is the $(\kappa-1)$-th satellite in the solution), then we must have $r^\kappa_{i_\kappa v} = 1$ for some $(i_\kappa, v) \in \delta^+(i_\kappa)$, that is, i_κ must be the κ-th satellite in the solution.

Finally, constraints (3.15c) handle the portion Γ of the route between the last visited satellite, $i_{\kappa'}$, and the depot vertex, like shown in Figure 3.3. These constraints state that, if the arcs in Γ are selected in a solution, and $r_{i_{\kappa'} j} = 1$ for some j, then no satellite vertex must be set to be the $(\kappa'+1)$-th one, that is, we cannot have $r^{\kappa'+1}_{ij} = 1$ for any $(i, j) \in \delta^+(\bar{S})$.

Variable bounds

$$0 \leq x_{ij} \leq \begin{cases} \lfloor \frac{\bar{c}}{2} \rfloor, & \text{if } i \in S_c \text{ and } j \in S_d \\ 1, & \text{otherwise} \end{cases} \qquad (i,j) \in \bar{A} \qquad (3.16a)$$

$$0 \leq y_i \leq 1 \qquad i \in \bar{V} \qquad (3.16b)$$

$$0 \leq z_{kl} \leq 1 \qquad (k,l) \in \mathcal{V} \qquad (3.16c)$$

$$0 \leq w^{kl}_{ij} \leq \begin{cases} \lfloor \frac{\bar{c}}{2} \rfloor, & \text{if } i \in S_c \text{ and } j \in S_d \\ 1, & \text{otherwise} \end{cases} \qquad (i,j) \in \bar{A}, (k,l) \in \mathcal{V} \qquad (3.16d)$$

$$0 \leq r^\kappa_{ij} \leq 1 \qquad (i,j) \in \delta^+(\bar{S}), \kappa = 1, \ldots, 2\bar{c} \qquad (3.16e)$$

Variable bounds are provided by (3.16). Upper bounds for variables x and w are given by the number of times each arc can be traversed, and are in accordance to Proposition 3.

3.3.1 Proof of the Formulation Correctness

We now present a proof that the IP model introduced in this section is in fact a formulation for the PSTTRPTW. The proof requires to show that there is a bijective mapping between integer vectors in $\mathcal{P}$ and feasible solutions for the problem.

Consider a feasible solution for the problem. From the exposition given in this section, it was already shown that it is possible to build an integer vector $(x, y, z, w, r) \in \mathcal{P}$ whose objective value equals the solution cost, and for which constraints (3.2)-(3.16) hold. Therefore, this set of constraints does not cut off any feasible solution. The opposite direction of this statement, i.e., that the set of constraints that define $\mathcal{P}$ are not satisfied by any infeasible solution, remains to be proved. The next lemma demonstrates this result.

Lemma 1. *For each $(x, y, z, w, r) \in \mathcal{P} \cap \mathbb{Z}^{\sigma}$, there is a feasible solution for the PSTTRPTW with cost equal to the objective value for (x, y, z, w, r).*

Proof. Let (x, y, z, w, r) be an integer vector satisfying constraints (3.2)-(3.16). We now describe how a feasible solution for the problem can be obtained.

If $z_{k0} = 0$ for every k, then the solution is empty with objective value zero. Otherwise, select the truck k for which $z_{k0} = 1$. Note that, by (3.8), there can be at most one such $k \in \mathcal{K}$. For this index k, if $z_{kl} = 1$ for some $l \in \mathcal{L}$, then the solution uses trailer l with truck k. Otherwise, the solution consists of a truck-only route. Note that, by (3.9a), and because $w \geq 0$, there can be at most one such l, and only for the selected k.

The solution route is built according to the following procedure:

1. **Set** $i \leftarrow 0$ and $\kappa \leftarrow 1$.
2. **If** $i \in \{0\} \cup C$, **then** add to the route the arc (i, j) for which $x_{ij} = 1$, and **go to** step 5.
3. Add to the route the arc (i, j) for which $r_{ij}^{\kappa} = 1$.
4. **Set** $\kappa \leftarrow \kappa + 1$.
5. **Set** $i \leftarrow j$.
6. **If** $i \neq 0$, **then go to** step 2.

The following six statements highlight properties for the route built by this procedure. They will be used to demonstrate that this solution is feasible for the PSTTRPTW.

Statement 1. *Whenever steps 2 or 3 are reached, there is an arc (i, j) that can be added to the route.*

Proof. The proof is by induction on the procedure iterations. In the first iteration, we have $i = 0$. Let k be the selected truck. First, suppose that $z_{kl} = 0$ for all l. Then, constraint (3.9a) provides $w^{k0}(\delta^+(0)) = 1$. Therefore, there is exactly one arc (i, j) for which $w_{ij}^{k0} = 1$. By (3.2), $x_{ij} = 1$, and this will be the arc that the procedure adds. If, otherwise, $z_{kl} = 1$ for some $l \in \mathcal{L}$, then, (3.9b) gives $w^{kl}(\delta^+(0)) = 1$, and, with the same argument, we obtain an arc (i, j) for which $x_{ij} = 1$.

Now, suppose that the procedure arrives at a given iteration with a connected walk, and we show that there is an arc to be added at the next iteration. First, assume that $i \in C$, which means the condition in step 2 is true. Since this iteration is not the first, i was set in step 5 of the previous iteration, which means there is an arc (v, i) for which $x_{vi} = 1$. By (3.3), this implies that there is an arc (i, j) for which $x_{ij} = 1$. This is the next arc to be added by the procedure.

In the case that $i \in \bar{S}$, we reach step 3. If no satellite vertex was added until this point, let Γ be the route built so far by the procedure, with $\gamma_1 = 0$ and $\gamma_{|\Gamma|} = i$. Consider constraint (3.15a) for Γ. We have $x(\Gamma) = |\Gamma|$, since $x_{ij} = 1$ for all arcs in Γ. This leads to $r^1(\delta^+(i)) \geq 1$, which means there is an arc (i, j) to be added to Γ.

If, otherwise, i is not the first satellite that the procedure adds, a similar argument can be used. Let γ_1 be the previous satellite vertex in the route before i, and let Γ be the portion of the route between γ_1 and i. Applying constraint (3.15b) for Γ and the current value of κ leads to $r^\kappa(\delta^+(i)) \geq 1$, and we also have an arc to be added by the procedure. □

Statement 2. *The procedure always finishes. Therefore, the generated route is connected, starting and ending at the depot vertex.*

Proof. We have to show that the procedure eventually reaches step 6 with $i = 0$. Suppose that this does not happen. Since the set of vertices is finite, there is at least one vertex v that appears an infinite number of times in the procedure route. Vertex v cannot be a satellite vertex, as this would require that $r_{vj}^\kappa = 1$ for infinite indexes κ, and these are defined only for $1, \ldots, 2\bar{c}$. Thus, $v \in C$. Note that, because of (**??**), the choice of arc in step 2 is unique. This means that, after reaching v, the procedure loops indefinitely between the same sequence of customer vertices. Let W be the set containing these vertices. For each $i \in W$, there is exactly one j for which $x_{ij} = 1$, and, since the vertices in W form a loop, we have $x(\delta^+(W)) = 0$. Furthermore,

because of (**??**), we have $y_v = 1$. Applying constraint (3.4) for W and v, we have a contradiction. □

Statement 3. *Each customer i for which $y_i = 1$ is visited in the route exactly once.*

Proof. Let $i \in C$ be a customer such that $y_i = 1$. As already noted, if i is reached by the procedure, the choice of arc (i, j) in step 2 is unique, by (**??**). Therefore, i cannot appear in the route more than once, since this would lead to an infinite loop.

Suppose, by contradiction, that the procedure finishes without adding i. First, assume that the generated route does not include satellite vertices. In this case, let W be the set of vertices in the route. Note that $x(\delta^+(W)) = 0$, and, because of (3.3), $x(\delta^-(W)) = x(\delta^+(W)) = 0$. Applying constraint (3.4) for $V \setminus W$ and i leads to a contradiction.

Now, suppose that the route that does not include i has $\bar{\kappa} \geq 1$ satellite vertices. Since $y_i = 1$, by (3.4), (3.3) and (**??**), there is exactly one arc (j, i) for which $x_{ji} = 1$. Note that $j \neq 0$, because otherwise i would have been included in the route. If $j \in C$, then j was also not included in the route, and, by (**??**), $y_j = 1$ too. In this case, apply the same operation for j, renaming this vertex to i. We can repeat this until we reach a satellite vertex v (note that this operation cannot reach repeated customer vertices, as this would violate (3.4) again). We have, then, an arc (v, i) for which $x_{vi} = 1$. By constraint (3.11), $r^{\kappa}_{vi} = 1$ for some κ. We cannot have $\kappa \leq \bar{\kappa}$, since, in this case, i would have been included in the route. Then, $r^{\kappa}_{vi} = 1$ for $\kappa > \bar{\kappa}$. Applying (3.12c) and (3.12d) repeatedly for $\kappa, \kappa - 1, \ldots, \bar{\kappa} + 2$ leads to the conclusion that there is a satellite vertex w for which $r^{\bar{\kappa}+1}(\delta^+(w)) = 1$. Let Γ be the portion of the generated route after the last visited satellite until the depot. Considering constraint (3.15c) for Γ and $\bar{\kappa}$, we get $r^{\bar{\kappa}+1}(\delta^+(S)) \leq 0$, which is another contradiction. □

Statement 4. *The route built by the procedure is satellite consistent.*

Proof. The procedure ensures that, if the generated route includes satellites, their order is given by indexes κ for which some variable r is active. Constraints (3.12), in turn, enforce that this order agrees with Definition 7, as follows: i) (3.12a)-(3.12b) state that the first satellite to be visited is a decoupling satellite; ii) (3.12c) grants that, after visiting i_d or i_t, the next satellite must be i_t or i_c, and iii) by (3.12d), only a decoupling satellite can follow a coupling one in the sequence.

In order to verify that condition iv) of Definition 7 also holds, suppose that it does not, that is, the last satellite s in the route is a decoupling or load transfer satellite. Since the procedure arrived at s, $x_{is} = 1$ for some vertex i. Because of the rules for $\bar{G}$, only complete vehicles are allowed to use arcs in $\delta^-(s)$, so, by (3.2), we must have

$w_{is}^{kl} = 1$ for some complete vehicle (k, l), which, from (3.9e) or (3.9g), implies that $z_{kl} = 1$. Thus, the solution involves a complete vehicle.

Now, because of step 3, we have $r_{si}^{\kappa} = 1$ for some $i \in \delta^+(s)$ and some κ. By (3.11), $x_{si} \geq 1$ holds. Because of the construction rules for $\bar{G}$, only single trucks are allowed to use arcs in $\delta^+(s)$; therefore, using (3.2), it follows that $w_{si}^{k0} = 1$. If $i \in C$, applying (3.10) for i leads to $w^{k0}(\delta^+(i)) = 1$, which means $w_{ij}^{k0} = 1$ for some arc (i, j). If j is also a customer, we can rename j to i and apply (3.10) again, until we reach the depot vertex, for which we will have $w_{i0}^{k0} = 1$. On the other hand, reminding that $z_{kl} = 1$, constraint (3.9a) states that $w^{k0}(\delta^+(0)) = 0$. Using now (3.10) for vertex 0, we arrive at a contradiction. Therefore, the last satellite vertex must be a coupling one, and the statement holds. □

Statement 5. *The route built by the procedure is capacity consistent for the selected vehicle.*

Proof. Statement 3, along with (**??**), establish that the customers visited in the generated route are exactly those for which $y_i = 1$. Then, the left side of (3.7) is exactly the total supply collected, which, by (3.7), is no greater that the selected vehicle capacity.

Now, consider a subtour starting at satellite $i \in S_d \cup S_t$ whose supply exceeds the selected truck capacity, and let Γ be the portion of the route from i to the last customer in this subtour. Then, constraint (3.13) is violated for Γ and the selected truck k. Therefore, the supply in all subtours is compatible with the truck capacity. □

Statement 6. *The route built by the procedure is time window consistent for the selected vehicle.*

Proof. Suppose that the time taken by the selected vehicle (k, l) to reach a visited customer i in the route exceeds f_i, and let Γ be the portion of the route starting at the depot vertex and ending at i. Consider constraint (3.14) for Γ and (k, l). The left side of this constraint equals $|\Gamma|$, which leads to $z_{kl} \leq 0$. However, since (k, l) is the selected vehicle, $z_{kl} = 1$ must hold, and a contradiction follows. Therefore, time windows are always observed in the generated route. □

Together, the statements above show that the vehicle selection and the route described produce a solution that matches all the properties that characterize a feasible solution in Remark 1: by Statement 1, the route is a walk that starts at the depot vertex and, by Statement 2, also ends at the depot. From Statement 3, no customer is visited more than once, and the following three statements establish the consistency properties.

As for the cost of this solution, we define it the following way: for each time arc (i, j) is added to the route, if the vehicle is coupled, we add c_{ij}^{kl} to the cost, and if it is decoupled, we add c_{ij}^{k0}. It is possible to verify that constraints (3.9) ensure that this cost equals the objective value of vector (x, y, z, w, r). □

The results discussed in this section allow us to state the formulation correctness, formalized in Theorem 1.

Theorem 1. *The Integer Program* (3.1) *correctly formulates the PSTTRPTW.*

□

3.3.2 Handling the Constraint Sets

The formulation introduced in this section has a polynomial number of integer variables, and does not rely on continuous variables for quantities like load or time. The inclusion of such types of variables to impose time windows and capacity consistencies would require the use of big-M disjunctive constraints, and these tend to generate poor LPR bounds. We avoided their use by imposing exponentially many constraints (3.13)-(3.15) that are applicable to walks in $\bar{G}$. Constraints (3.13)-(3.15) can be seen as *Combinatorial Benders cuts* [Codato and Fischetti, 2006]. In order to handle the large number of constraints, we adopt a BC approach. DCUTs (3.4) are an exponential set of constraints that can be efficiently separated with well known max-flow routines [Padberg and Rinaldi, 1991]. Constraints (3.13), (3.14) and (3.15), on the other hand, are also numerous, and require another separation strategy, which is presented in Chapter 4.

Consider a solution for an integer relaxation of the formulation in which constraints (3.13)-(3.15) are dropped. This solution consists of a connected route, starting and ending at the depot vertex. Due to the structure of the auxiliary graph, the physical restrictions of the problem, that impose that a complete vehicle cannot visit a truck-only customer, and coupling/decoupling rules, are observed by this solution. Compatibility between the total collected supply and vehicle capacity also hold. However, there are three types of infeasibilities that such a solution might carry:

- Satellite inconsistencies: the order at which satellite vertices appear in the route, given by variables x, may not be consistent with the ordering given by variables r. For instance, in this solution, it is possible that the vehicle decouples at satellite i and, after visiting some customers, visits non-related coupling satellite j. This

type of violation is addressed by constraints (3.15), and we will refer to this set of constraints as *scheduling cuts*.

- Truck capacity in subtours: in this relaxed solution, after a decoupling or load transferring operation, the truck may collect an amount of supply greater than its capacity without transferring its load before. A violation of this nature is eliminated with the introduction of constraints (3.13) to the model, which we call *capacity cuts*.

- Time windows inconsistencies: the selected vehicle in the solution may arrive at a customer after its time window closing. Constraints (3.14) can eliminate this type of violation, and we refer to them as *time window cuts*.

Based on this analysis, we can expect that, in a BC approach, in which constraints (3.13)-(3.15) are not initially included in the model, the number of generated cuts will directly depend on some characteristics of the instance being solved. For example, if the number of satellites that need to be visited in an optimal solution is close to zero, the number of scheduling cuts is likely smaller than for an instance that requires multiple decoupling and load transfer operations. If customer time windows are relatively loose, fewer time window cuts will be required, in contrast with an instance with tighter time windows. Also, if truck capacities are relatively large compared to customer supplies, fewer capacity cuts are expected to be needed. Therefore, although the sets of cuts to be added is theoretically exponential, this approach may be effective under certain conditions.

3.4 Preprocessing and Variable Fixing

Given a PSTTRPTW instance, it is often possible to preprocess it in order to build an equivalent, "lighter", instance, in which at least one optimal solution is not lost. It is also possible to state that some variables are never active in an optimal solution, making it possible to fix their values at zero. The objective of this operation is to reduce the number of variables and constraints in the model. In some cases, as described in the next section, preprocessing also allows to introduce valid inequalities for the formulation.

Preprocessing performs verifications over the input instance as follows. In the expressions below, for simplicity, whenever $i \notin C$, we define $e_i := 0$ and $q_i := 0$.

1. For each $(i, j) \in \bar{A}$ with $i \in C$ or $j \in C$, and each vehicle $(k, l) \in \mathcal{V}$ that can use arc (i, j), check if $q_i + q_j > Q_k + Q_l$. If so, the capacity of vehicle (k, l) is not

sufficient to consecutively visit i and j without performing a load transfer, and we can fix $w_{ij}^{kl} = 0$. In practice, we can remove (i, j) for vehicle (k, l) from $\bar{G}$, and simply remove variable w_{ij}^{kl} from the model.

2. For each $(i, j) \in \bar{A}$ with $j \in C$, and each vehicle $(k, l) \in \mathcal{V}$ that can use arc (i, j), check if $f_j - e_i < t_{ij}^{kl}$. If so, vehicle (k, l) can never reach customer j in time after leaving i. In this case, we also remove (i, j) for vehicle (k, l) from $\bar{G}$, and delete w_{ij}^{kl} from the model.

3. For each $(i, j) \in \bar{A}$, if arc (i, j) was removed for every vehicle that could use it, then this arc will not be used at all, and $x_{ij} = 0$. Arc (i, j) can be removed from $\bar{G}$.

4. For each customer $i \in C$, if arc $(0, i)$ was removed, this means that customer i cannot appear in any feasible solution for any chosen vehicle, either because of capacity or time window constraints. Then, y_i can be fixed at zero, and we can remove from $\bar{G}$ vertex i and all arcs adjacent to i.

3.5 Strengthening the Formulation

The formulation described in this chapter for the PSTTRPTW can be strengthened with the addition of valid inequalities and lifting some of the constraints introduced earlier, which enhances its LPR bounds.

3.5.1 Valid Inequalities

We now introduce valid inequalities for the problem in the hope of strengthening the LPR bounds provided by $\mathcal{P}$.

2-matching inequalities [Edmonds, 1965]. 2-matching inequalities were proven valid for the matching polytope and have been used to reinforce LPR for the TSP and several of its variants. They can be adapted for the PSTTRPTW as follows. We consider only vertices that can be visited once, thus satellite vertices are not involved in the definition of these inequalities. Given $H \subseteq \{0\} \cup C$ and $T \subseteq \delta(H : \{0\} \cup C \setminus H)$,

$$\sum_{(i,j)\in A(H)} x_{ij} + \sum_{(i,j)\in T} x_{ij} \leq |H| + \left\lfloor \frac{|T|}{2} \right\rfloor \tag{3.17}$$

is valid for $3 \leq |H| \leq n-1$ and non-redundant for $|T| \geq 3$, odd.

Mutually excluding vertices. Let $i, j \in \bar{V}$, and assume that both arcs (i, j) and (j, i) violate the criterion in item 2 of preprocessing for every vehicle (k, l), that is, $f_j - e_i < t_{ij}^{kl}$ and $f_i - e_j < t_{ji}^{kl}$. Then, since the distances in G satisfy the triangular inequality, even if the service to customer i starts when its time window opens, customer j cannot be reached in time by any vehicle after i, and the opposite also holds: if j is serviced, no vehicle can arrive at i in time. This means customers i and j cannot be simultaneously part of any feasible solution, and the following constraint can be added to the formulation:

$$y_i + y_j \leq 1 \tag{3.18}$$

Infeasible sets elimination. This set of inequalities derives from one of the preprocessing criteria in [Ascheuer et al., 2001], a paper dedicated to the Asymmetric TSP with Time Windows. Differently from the PSTTRPTW, for that problem, all customers have to be visited. In order to handle this difference, we adapted their preprocessing routine into a set of valid inequalities.

Let $i \in \{0\} \cup C, j \in C$ and $W \subset C \setminus \{i, j\}$. We try to create a feasible path by concatenating arc (i, j) with all permutations of set W. If all possible resulting paths are infeasible due to customer time windows, set W and arc (i, j) are mutually exclusive in a feasible solution, and the following constraint is valid for the formulation:

$$x_{ij} + \sum_{k \in W} y_k \leq |W|. \tag{3.19}$$

In order to calculate travel times in these paths, the speed of the fastest vehicle in $\mathcal{V}$ is used. In our implementation, for each arc, we consider all customer sets with $|W| \leq 2$.

Related satellites. In order to reinforce that related satellite vertices are visited consistently in a solution, the following sets of constraints are used:

$$y_{i_d} = y_{i_c} \qquad i \in S, \tag{3.20}$$

$$y_{i_t} \leq y_{i_d} \qquad i \in S. \tag{3.21}$$

By (3.20), if satellite vertex $i \in S$ is used in the solution for decoupling, it must

also be used for coupling. Inequalities (3.21) establish that a satellite can only be used for load transfer if it is used for decoupling.

3.5.2 Lifting Constraints

In this subsection, we describe how some inequalities can be lifted in order to better approximate the convex hull of PSTTRPTW integer feasible solutions and, thus, to provide better LPR bounds for the problem.

Opposite direction arcs. *Applicable for constraints* (**??**). If $i, j \in C$, and arcs (i, j) and (j, i) both exist in $\bar{G}$, that is, they were not removed by the preprocessing routine, we can state that no feasible solution can use both arcs. Therefore, constraint (**??**) can be rewritten for vertex i (and similarly for j) as

$$y_i \geq x_{ij} + x_{ji} \tag{3.22}$$

Cutting off multiple vehicles. *Applicable for constraints* (3.13) *and* (3.14). Capacity cuts (3.13) and time window cuts (3.14) both forbid solutions that select a certain vehicle (k, l) and a given walk Γ, provided that the combination of (k, l) and Γ is not consistent. If multiple vehicles are forbidden for Γ, they can all be considered in a single, lifted cut.

For capacity cuts, let Γ be a walk matching this constraint condition, and let $\mathcal{K}' \subseteq \mathcal{K}$ be the set of all trucks with capacity bigger than $q(\Gamma)$. Then, (3.13) can be replaced by

$$x(\Gamma) \leq |\Gamma| - 1 - \sum_{k \in \mathcal{K}'} z_{k0}. \tag{3.23a}$$

If $\mathcal{K}' = \mathcal{K}$, then Γ can be simply cut off, regardless of the truck choice:

$$x(\Gamma) \leq |\Gamma| - 2. \tag{3.23b}$$

Now, consider time window cuts, for which a similar argument can be used. Let $\mathcal{K}' \subseteq \mathcal{K}$ be the set of trucks that cannot travel through Γ in time, even decoupled. Then, any truck $k \in \mathcal{K}'$ cannot be used with path Γ with or without a trailer, and we can subtract z_{k0} from the right side of (3.14), like above. Furthermore, for $k \in \mathcal{K} \setminus \mathcal{K}'$, let $\mathcal{L}'_k \subseteq \mathcal{L}$ be a set of trailers l for which $t_{kl}(\Gamma) > f_{\gamma_{|\Gamma|}}$. Then, these complete vehicles

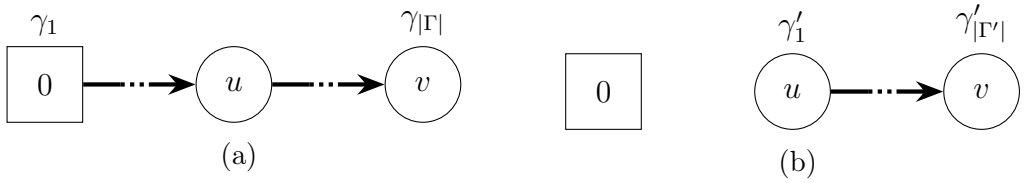


Figure 3.4: Contracting a walk Γ, from vertex 0 to v (a), into a walk Γ', from u to v (b).

(k, l) can also be eliminated, and the time window cuts can be lifted as

$$x(\Gamma|\{0\} \cup C) + \sum_{i=1}^{s(\Gamma)} r^i_{\gamma_{s_i}\gamma_{s_i+1}} \leq |\Gamma| - 1 - \sum_{k \in \mathcal{K}'} z_{k0} - \sum_{k \in \mathcal{K} \setminus \mathcal{K}'} \sum_{l \in \mathcal{L}'_k} z_{kl}. \tag{3.24a}$$

Note that, if Γ cannot be traveled by single trucks (because of the presence of a coupling satellite), then set $\mathcal{K}'$ is not applicable, and it is considered to be empty. If no vehicle can travel through Γ in time, we can again replace the z sums by 1, resulting in the stronger inequality

$$x(\Gamma|\{0\} \cup C) + \sum_{i=1}^{s(\Gamma)} r^i_{\gamma_{s_i}\gamma_{s_i+1}} \leq |\Gamma| - 2. \tag{3.24b}$$

Contracting walks. *Applicable for* (3.14). Let Γ be a walk starting at the depot vertex, that cannot be traveled in time by vehicle $(k, l) \in \mathcal{V}$, and define $v = \gamma_{|\Gamma|}$, as in Figure 3.4a. Let $u \in C(\Gamma)$ be the last customer in Γ for which vehicle (k, l) cannot reach v in time, that is, $f_v - e_u$ is smaller than the time spent by (k, l) to travel the portion of Γ between u and v. If there is no such customer u, we cannot contract walk Γ, so assume u exists, and let Γ' be the subwalk of Γ starting at u and ending in v, as illustrated in Figure 3.4b.

If Γ' has no satellite vertices, time window cuts can be taken considering only Γ', and we have

$$x(\Gamma') \leq |\Gamma'| - 1 - z_{kl} \tag{3.25a}$$

If, however, Γ' contains satellite vertices, we cannot simply use Γ', as this could cut off possibly feasible solutions that visit v before u. We search for the last satellite vertex u' that appears before u in Γ. If it exists, we consider subwalk Γ'', starting at u' and ending at v, and (3.14) becomes

$$x(\Gamma''|\{0\} \cup C) + \sum_{i=\kappa}^{s(\Gamma'')+\kappa-1} r^i_{\gamma''_{s_i}\gamma''_{s_i+1}} \leq |\Gamma''| - 1 - z_{kl} \quad \kappa = 1, \ldots, 2\bar{c} - s(\Gamma'') \tag{3.25b}$$

Note that we consider all possible κ indexes, since now any number of satellites may be visited before walk Γ'' in a solution. We also remark that walk contraction is compatible with the vehicle elimination rule described before, so both lifting strategies can be applied for the same time window cut.

Unrelated satellite vertices in subtours. *Applicable for constraints* (3.15b). If a solution includes a subtour starting at $i \in S_d \cup S_t$ and ending at an unrelated satellite, it can be eliminated regardless of the κ indexes. In this case, scheduling cuts (3.15b) can be simplified to

$$x(\Gamma) \leq |\Gamma| - 2 \qquad \Gamma \in \mathcal{W}(\bar{G}) : \begin{cases} \gamma_1 = i \in S_d \cup S_t, \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C, \\ \gamma_{|\Gamma|} \in \{0\} \cup \bar{S} \setminus \{i_t, i_c\} \end{cases} \tag{3.26}$$

Although they are a special case of scheduling cuts, this set of constraints is stronger, since, for each Γ, all the original constraints, for each index κ, are replaced by a single one. Constraints (3.15b) are, however, more general, and are still necessary when Γ does not match the criteria in (3.26).

Cutting off infeasible arcs after a walk. *Applicable for constraints* (3.15a) *and* (3.15b). In these scheduling cuts, if $i \in C(\Gamma)$, then arc $(\gamma_{|\Gamma|}, i)$ does not need to be considered on the right side of the constraint, since no feasible solution can include this arc and Γ at the same time. The lifted versions of these constraints become, respectively,

$$x(\Gamma) \leq \sum_{(\gamma_{|\Gamma|},i) \in \delta^+(\gamma_{|\Gamma|}) : i \notin C(\Gamma)} r^1_{\gamma_{|\Gamma|}i} + |\Gamma| - 2 \tag{3.27a}$$

$$r^{\kappa-1}_{\gamma_1\gamma_2} + x(\Gamma|C) \leq \sum_{(\gamma_{|\Gamma|},i) \in \delta^+(\gamma_{|\Gamma|}) : i \notin C(\Gamma)} r^{\kappa}_{\gamma_{|\Gamma|}i} + |\Gamma| - 2. \tag{3.27b}$$

3.6 Comments

In this chapter, we introduced an IP formulation for the PSTTRPTW. A BC algorithm based on this formulation, that handles the exponential-sized sets of constraints (3.13)-(3.15), is discussed next, in Chapter 4. One of the possible applications for this algorithm is to solve the pricing subproblem in a column generation scheme for a multiple vehicle TTRPTW, in place of a DP algorithm, which, as discussed earlier, is the method usually adopted in the literature for this type of problems.

A BC approach to solve this subproblem has properties, as discussed in Subsection 3.3.2, that offers a contrast with DP. The DP algorithm is able to prune infeasible choices. For example, if the time window of customer i closes before the time window of customer j opens, the algorithm can discard every solution in which j is visited before i. In a BC approach, however, a number of time window cuts may be needed to eliminate such solutions. On the other hand, in a less restrictive scenario, DP may be unable to eliminate many possible solutions early, leading to a poor performance, whereas a BC scheme may find an optimal solution faster, without the need of adding many cuts to the LPRs.

Like for time windows, truck capacities also give rise to a similar parallel between BC and DP: if a truck is able to service a relatively large number of customer without needing to transfer load, DP has to deal with a larger set of feasible configurations, while BC may need few capacity cuts, leading to a better performance of the latter method. On the other hand, smaller truck capacities tend to favor DP, since a large number of cuts will probably be generated by BC.

These strategies also differ on that DP usually solves a routing problem for a fixed vehicle. Therefore, the solution of a pricing subproblem may require multiple calls to a DP routine, for different vehicles. In contrast, in BC, all vehicles are considered at once, and the algorithm searches for the best choice of vehicle and route. A BC approach is able to benefit from this property, by cutting off routes for multiple vehicles at once, like in the lifting strategies discussed for capacity and time window cuts, (3.23) and (3.24), respectively.

Chapter 4

A Branch-and-Cut Algorithm for the PSTTRPTW

This chapter describes Branch-and-Cut [Padberg and Rinaldi, 1991] algorithm based on the formulation introduced in Chapter 3 to solve the PSTTRPTW. Since the formulation includes an exponential number of constraints, we cannot hope to explicitly include all of them into a LPR to provide bounds for a Branch-and-Bound procedure. Instead, we discuss how each exponentially sized set of constraints can be efficiently separated after solving a LPR.

Let $\mathcal{P}^* \subset \mathbb{R}^\sigma$ be the polyhedron defined by the constraints that define $\mathcal{P}$, (3.2)-(3.16), plus the valid inequalities (3.17)-(3.21), discussed in Subsection 3.5.1. The BC algorithm is based on this reinforced formulation. There are three sets of constraints in $\mathcal{P}^*$ that are initially not entirely included in the model by the algorithm: DCUTs (3.4), 2-matching inequalities (3.17), and constraints (3.13)-(3.15), which we refer to as *infeasible walk elimination constraints* (IWECs). The remaining constraints are included in the model right from the start. At each node of the BC procedure, a LPR of the associated subproblem is solved. If a feasible solution is found, no matter if fractional or not, a separation routine is called to look for violated constraints that are not in the current relaxation. If a violated constraint is found, it is then added to the model, and the linear relaxation is re-solved. Branching takes place when the solution to the LPR is fractional and no violated inequalities are found.

4.1 Separation Algorithms

In this section, we describe the separation algorithms in charge of identifying whether or not the solution to the LPRs violate one or more inequalities (3.4), (3.17), and (3.13)-

(3.15). Separation routines are called whenever a solution for the relaxed problem is found, and it is possible to set different procedures depending on the integrality of the solution. For the DCUTs, the same procedure applies for both fractional and integral solutions; the routine for 2-matching inequalities is called only when the solution is fractional, and the IWECs have a different separation algorithm for each case.

4.1.1 Directed Cutset Constraints

In our algorithm, the DCUTs (3.4) are initially included in the model for $|W| = 1$. Remaining ones, i.e., those defined by sets $W : |W| \geq 2$, are separated by well-known separation routine based on maximum flow networks, introduced in [Padberg and Rinaldi, 1991], is used. Given an optimal solution $(\bar{x}, \bar{y}, \bar{z}, \bar{r}, \bar{w})$ for the LPR being considered at a given separation round, a network is built from $\bar{G}$, in which each arc (i, j) is given a weight corresponding to the value $\bar{x}_{ij}$ it takes in the given solution. Then, we use Dinic's algorithm [1970] to find the minimum cut W that separates a customer $i \in C$ from the depot vertex, 0. If the capacity $\bar{x}(\delta^+(W))$ of the cut $\delta^+(W)$ is less than $\bar{y}_i$, a violated DCUT, separating i from 0, was found. The procedure is called for each $i \in C : \bar{y}_i > 0$. For each violated cut set satisfying these criteria, the corresponding DCUT is added to the current LPR.

4.1.2 2-Matching Inequalities

We use the separation heuristic introduced in [Grötschel and Holland, 1988] for 2-matching inequalities (3.17). The procedure is only called when the relaxed solution is fractional. Given a fractional solution $(\bar{x}, \bar{y}, \bar{z}, \bar{r}, \bar{w})$, the heuristic procedure builds an undirected graph $\tilde{G} = (\{0\} \cup C, E)$ from $\bar{G}$, in which satellite vertices are excluded, and an edge $\{i, j\} \in E$ is created if and only if $\bar{x}_{ij} + \bar{x}_{ji} > 0$ holds. The heuristic is based on a parameter $0 < \varepsilon < \frac{1}{2}$, and considers the subgraph $\tilde{G}_\varepsilon = (\{0\} \cup C, E_\varepsilon)$, where $E_\varepsilon = \{\{i, j\} \in E : \varepsilon \leq \bar{x}_{ij} + \bar{x}_{ji} \leq 1 - \varepsilon\}$. Then, the procedure looks for the connected components of $\tilde{G}_\varepsilon$. For each connected component H with $|H| \geq 3$, it tries to find a set of arcs T in $\tilde{G}_\varepsilon$, with $T \subseteq \delta(H)$, such that $|T|$ is odd and minimum, $|T| \geq 3$, and $\bar{x}(E(H)) + \bar{x}(T) > |H| + \frac{|T|}{2}$. If such a set exists, we add the corresponding violated 2-matching inequality to the current LPR.

This heuristic can be implemented to run in $O(n + m)$ time complexity. In our implementation, we set the parameter ε to 0.1. In the separation procedure, the 2-matching routine is called after the DCUTs separation, only if the current solution violates no DCUT.

4.1.3 Infeasible Walk Elimination Inequalities

Constraints (3.13), (3.14) and (3.15) are separated by the procedures described in this section. We implemented two different separation procedures for them: one for fractional solutions and another for integral ones, and therefore they are described separately. In both cases, the routine looks for walks in the subgraph of $\bar{G}$ induced by the current solution $(\bar{x}, \bar{y}, \bar{z}, \bar{r}, \bar{w})$ that have a capacity, a time window or a satellite inconsistency for a vehicle selected in the solution, and generates a corresponding violated inequality. For the fractional separation routine, multiple violated inequalities can be added for the same separation point. For the integer separation cases, we always add a single cut for the first inconsistency found. Whenever possible, before being added to the LPR, the inequality is lifted, as described in Subsection 3.5.2.

Since these separation algorithms assume that the input solution satisfies all connectivity constraints, they are only called after the DCUT separation, and only if no violated DCUT is found at that separation round.

4.1.3.1 Fractional Separation

In the fractional point separation for IWECs, we consider the support graph $\tilde{G} = (\bar{V}, \tilde{A})$ of $\bar{G}$, in which arc $(i, j) \in \tilde{A}$ exists if and only if $(i, j) \in \bar{A}$ and $\bar{x}_{ij} > 0$. Then, using the values of $\bar{x}$ as arc weights, we compute shortest paths between all pairs of vertices. For each of these shortest paths Γ in $\tilde{G}$, we check if Γ matches certain criteria for characterization of the families (3.13)-(3.15) of inequalities, their lifted versions, and additional valid inequalities introduced in Section 3.5. In the following, we describe the criteria for each type of cut that the fractional separation procedure can generate. They follow directly from the constraints introduced in Chapter 3, as well as their corresponding lifted inequalities.

1. **Subtour capacity violation:** If

$$\begin{cases} \gamma_1 \in S_d \cup S_t \\ \gamma_2, \ldots, \gamma_{|\Gamma|} \in C \\ q(\Gamma) > Q_k \text{ for } k \in \mathcal{K}' \subseteq \mathcal{K} \\ \bar{x}(\Gamma) > |\Gamma| - 1 - \sum\limits_{k \in \mathcal{K}'} \bar{z}_{k0} \end{cases}$$

   then, a violated capacity cut was identified and can be added to the current LPR. If $\mathcal{K}' = \mathcal{K}$ we use the lifted form (3.23b) for the cut; otherwise, the inequality as formulated as (3.23a).

2. **Coupling violation:** If

$$\begin{cases} \gamma_1 = i \in S_d \cup S_t \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C \\ \gamma_{|\Gamma|} \in \bar{S} \setminus \{i_d, i_t\} \\ \bar{x}(\Gamma) > |\Gamma| - 2 \end{cases}$$

then, a violated lifted scheduling cut (3.26) can be added to the current LPR.

3. **First satellite scheduling violation:** If

$$\begin{cases} \gamma_1 = 0 \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C \\ \gamma_{|\Gamma|} \in \bar{S} \\ \bar{x}(\Gamma|C) > \sum\limits_{(\gamma_{|\Gamma|}, i) \in \delta^+(\gamma_{|\Gamma|}): i \notin C(\Gamma)} \bar{r}^1_{\gamma_{|\Gamma|} i} + |\Gamma| - 2 \end{cases}$$

then, a violated lifted scheduling cut (3.27a) can be added to the current LPR.

4. **Intermediate satellite scheduling violation:** If

$$\begin{cases} \gamma_1 = i \in \bar{S} \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C \\ \gamma_{|\Gamma|} \in \bar{S} \\ \bar{r}^{\kappa-1}_{\gamma_1 \gamma_2} + \bar{x}(\Gamma|C) > \sum\limits_{(\gamma_{|\Gamma|}, i) \in \delta^+(\gamma_{|\Gamma|}): i \notin C(\Gamma)} \bar{r}^{\kappa}_{\gamma_{|\Gamma|} i} + |\Gamma| - 2 \text{ for some } \kappa = 2, \ldots, 2\bar{c} \end{cases}$$

then, a violated lifted scheduling cut (3.27b) can be added to the current LPR.

5. **Last satellite scheduling violation:** If

$$\begin{cases} \gamma_1 = i \in S_c \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C \\ \gamma_{|\Gamma|} = 0 \\ \bar{r}^{\kappa}_{\gamma_1 \gamma_2} + \bar{x}(\Gamma|C) + \bar{r}^{\kappa+1}(\delta^+(\bar{S})) > |\Gamma| - 1 \text{ for some } \kappa = 1, \ldots, 2\bar{c} - 1 \end{cases}$$

then, a violated scheduling cut (3.15c) can be added to the current LPR.

6. **Time window violation without satellite vertices:** If

$$\begin{cases} \gamma_1 \in \{0\} \cup C \\ \gamma_2, \ldots, \gamma_{|\Gamma|} \in C \\ t_{k0}(\Gamma) > f_{\gamma_{|\Gamma|}} - e_{\gamma_1} \text{ for } k \in \mathcal{K}' \subseteq \mathcal{K} \\ t_{kl}(\Gamma) > f_{\gamma_{|\Gamma|}} - e_{\gamma_1} \text{ for } k \in \mathcal{K} \setminus \mathcal{K}' \text{ and } \mathcal{L}'_k \subseteq \mathcal{L} \\ \bar{x}(\Gamma|\{0\} \cup C) > |\Gamma| - 1 - \sum\limits_{k \in \mathcal{K}'} \bar{z}_{k0} - \sum\limits_{k \in \mathcal{K} \setminus \mathcal{K}'} \sum\limits_{l \in \mathcal{L}'_k} \bar{z}_{kl} \end{cases}$$

then, a violated time window cut can be added to the current LPR. If $\mathcal{K}' = \mathcal{K}$, then no vehicle can traverse Γ in time to service its last customer, and we use the lifted form (3.23b) for the cut. Otherwise, $\mathcal{K}'$ is the set of trucks that cannot traverse Γ in time, even decoupled, and, for trucks $k \notin \mathcal{K}'$, $\mathcal{L}_k$ is the set of trailers such that the complete vehicle (k, l) cannot traverse Γ in time. In this case, we use the lifted inequality (3.24a). Since Γ does not include satellite vertices, this inequality is simplified to

$$x(\Gamma|\{0\} \cup C) \leq |\Gamma| - 1 - \sum_{k \in \mathcal{K}'} z_{k0} - \sum_{k \in \mathcal{K} \setminus \mathcal{K}'} \sum_{l \in \mathcal{L}'_k} z_{kl}$$

7. **Time window violation with satellite vertices:** If

$$\begin{cases} \gamma_1 \in \{0\} \cup \bar{S} \\ \gamma_2, \ldots, \gamma_{|\Gamma|-1} \in C \cup \bar{S} \\ \gamma_{|\Gamma|} \in C \\ t_{k0}(\Gamma) > f_{\gamma_{|\Gamma|}} - e_{\gamma_1} \text{ for } k \in \mathcal{K}' \subseteq \mathcal{K} \\ t_{kl}(\Gamma) > f_{\gamma_{|\Gamma|}} - e_{\gamma_1} \text{ for } k \in \mathcal{K} \setminus \mathcal{K}' \text{ and } \mathcal{L}'_k \subseteq \mathcal{L} \\ \bar{x}(\Gamma|\{0\} \cup C) + \sum\limits_{i=1}^{s(\Gamma)} \bar{r}^i_{\gamma_{s_i}\gamma_{s_i+1}} > |\Gamma| - 1 - \sum\limits_{k \in \mathcal{K}'} \bar{z}_{k0} - \sum\limits_{k \in \mathcal{K} \setminus \mathcal{K}'} \sum\limits_{l \in \mathcal{L}'_k} \bar{z}_{kl} \end{cases}$$

then, a time window cut can be added to the current model. Like for the previous case, if $\mathcal{K}' = \mathcal{K}$, then no vehicle can traverse Γ in time to service its last customer, and we use the lifted form (3.24b) for the cut. Otherwise, we use the lifted inequality (3.24a).

Shortest paths in $\tilde{G}$ are found by means of calling the one-to-all version of Dijsktra's algorithm. Let Γ be the shortest path between vertices i and j. Note that, if $\bar{x}(\Gamma) \leq |\Gamma| - 3$, it is not possible for Γ to violate any of the IWECs in the current

solution. Therefore, for each shortest path, we check if it satisfies this minimum length condition, before verifying if it matches one of the criteria above. Algorithm 1 describes the fractional separation procedure for IWECs.

```
Given a fractional solution (x̄, ȳ, z̄, r̄, w̄), build the support graph G̃ from Ḡ;
for i ∈ V̄ do
    Find all shortest paths from i in G̃;
    for j ∈ V̄ do
        Let Γ_ij be the shortest path from i to j in G̃
        if x̄(Γ_ij) > |Γ_ij| − 3 then
            if Γ_ij satisfies criterion 1 then
                Add the corresponding capacity cut;
            end
            else if Γ_ij satisfies criterion 6 then
                Add the corresponding time window cut;
            end
            else if Γ_ij satisfies criterion 2, 3, 4 or 5 then
                Add the corresponding scheduling cut;
            end
        end
    end
end
if no scheduling cut was found then
    for i ∈ {0} ∪ S̄ do
        for j ∈ C do
            if Γ_ij satisfies criterion 7 then
                Add the corresponding time window cut;
            end
        end
    end
end
```

Algorithm 1: Fractional separation for infeasible walk elimination constraints

Time window violation criteria (items 6 and 7 above) consider "contracted" walks, that start at a vertex different from 0, as described Subsection 3.5.2, with inequalities (3.25). This allows the procedure to find stronger inequalities. However, these criteria can generate redundant inequalities. To verify this, assume that a shortest path between vertices i and j, $i \neq 0$, violate a time window criterion. Then, the shortest path from 0 to j also violates the same criterion, as well as other vertices that may appear in a shortest path from 0 to i. In order to avoid adding redundant cuts, after finding violated constraints, the algorithm considers the normalized vectors corresponding to each cut. Then, we calculate the inner product between all pairs of such vectors and,

whenever the result is sufficiently close to 1, we discard the corresponding cut that contains more nonzero variable coefficients. This way, we ensure that paths that are contained in another path are discarded, and only stronger cuts are added. In our implementation, we used a tolerance of 0.1 when checking if an inner product is close to 1.

Since it is based on shortest paths, the procedure we just described is a heuristic, and not all inconsistencies are necessarily found, for fractional points. For example, if the solution contains a time window inconsistency for a walk from vertex i to vertex j that visits a satellite vertex more than once, this violation certainly will not be detected, since this walk is not a shortest path from i to j. Such inconsistencies are only handled by the integer separation routine, described in the following.

4.1.3.2 Integer Separation

Differently from the fractional separation, the integer separation procedure for the IWECs is an exact algorithm. In other words, the algorithm always finds a cut violated by the integer point to be separated, provided that one such cut does exist. The structure of this algorithm is similar to the procedure used in the proof of the formulation correctness, in Lemma 1. First, we identify the selected vehicle given by $\bar{z}$. Then, starting from the depot vertex, we iterate through each arc included in the solution, in the order they are indicated by the solution route. For each visited arc, we check for possible inconsistencies and, if any is found, a corresponding cut is added to the model and the procedure ends.

Algorithm 2 summarizes the integer separation routine. After defining the selected vehicle $(\hat{k}, \hat{l})$, we start to build the solution route Γ. Variable i represents the last vertex added to Γ at any iteration. The procedure iterates until the depot vertex is reached again, which means the route is complete. To discover the next vertex in the route at each iteration, we use variable x or r, depending on whether i is a customer or a satellite vertex, like in the procedure from the proof of Lemma 1. If i is a satellite, it is possible that no next vertex is found (which corresponds to Statement 1 not holding). If this happens, we have a satellite scheduling violation, and we generate a cut of type (3.15a) or (3.15b), depending on whether i is the first satellite vertex in Γ or not. If $\kappa = 1$, then i is the first satellite vertex, and we add a cut (3.15a) for walk Γ. Otherwise, we add a cut (3.15b), considering the portion of Γ starting at the last satellite vertex visited before i and ending at i.

During the iterations, we keep a flag variable to control the current coupling status of the vehicle in the walk. This status starts as *true* and, whenever a decoupling satellite

Let $(\bar{x}, \bar{y}, \bar{z}, \bar{r}, \bar{w})$ be the input integer solution
if $\bar{z}_{k0} = 1$ *for some* $k \in \mathcal{K}$ **then**
  $\hat{k} \leftarrow k$;
else
  return
if $\bar{z}_{\hat{k}l} = 1$ *for some* $l \in \mathcal{L}$ **then**
  $\hat{l} \leftarrow l$;
else
  $\hat{l} \leftarrow 0$;
$i \leftarrow 0$;
$\Gamma \leftarrow (i)$;
$\kappa \leftarrow 1$;
repeat
  if $i \in \bar{S}$ **then**
    if $\bar{r}^{\kappa}_{ij} = 1$ *for some* $(i,j) \in \delta^+(i)$ **then**
      $i \leftarrow j$;
      $\kappa \leftarrow \kappa + 1$;
    end
    else
      if $\kappa = 1$ **then**
        Generate a scheduling cut (3.15a);
      else
        Generate a scheduling cut (3.15b);
      return;
    end
  end
  else
    Let $(i,j) \in \delta^+(i)$ be the arc for which $\bar{x}_{ij} = 1$
    $i \leftarrow j$;
  end
  Add i to Γ;
  if $i \in C$ **then**
    if $q(\hat{\Gamma}(i)) > Q_{\hat{k}}$ **then**
      Generate a capacity cut (3.23);
      return;
    end
    if $t_{\hat{k}\hat{l}}(\Gamma) > f_i$ **then**
      Generate a time window cut (3.24);
      return;
    end
  end
until $i = 0$;
if $\bar{r}^{\kappa}_{ij} = 1$ *for some* $(i,j) \in \delta^+(\bar{S})$ **then**
  Generate a scheduling cut (3.15c);

Algorithm 2: Integer separation for infeasible walk elimination constraints

is reached, it is set to *false*. If a coupling satellite is reached, it is set to *true* again. While a vehicle is decoupled, we also store the total supply $q(\hat{\Gamma}(i))$ collected in the current subtour (i.e., since the last time a decoupling or transfer satellite was visited). If this value exceeds the capacity of the selected truck, a capacity inconsistency was found. We determine which trucks would be overloaded by the total subtour supply, and we generate a lifted capacity cut (3.23).

We also calculate, at each iteration, the time taken by vehicle $(\hat{k}, \hat{l})$ to traverse the current walk. If, at any customer i, this value exceeds its time window closing f_i, a time window inconsistency was found. At this point, we loop backwards in Γ in order to find the last customer j in Γ for which the selected vehicle cannot reach i in time, even if it departs at instant e_j. The portion of Γ between j and i is used to generate a lifted time window cut, as defined by (3.25). If this subwalk does not contain satellite vertices, we use the form (3.25a). Otherwise, we need to extend the subwalk until the last satellite vertex before j, and use the form (3.25b). We also determine which other vehicles cannot traverse this subwalk in time, and add the corresponding z variables to the right side of the cut, as in the lifted inequalities (3.24).

After reaching the depot vertex again, it is still necessary to check for scheduling inconsistencies of type (3.15c), which would correspond to Statement 3 not holding. If $\bar{r}^{\kappa}_{ij} = 1$ for some $(i, j) \in \delta^+(\bar{S})$ and the current value of variable κ, then the sequence of satellite vertices given by the values of $\bar{r}$ are not compatible with the route given by the solution. We consider the portion of Γ starting at the last satellite in the route and ending at the depot vertex, and we generate a scheduling cut (3.15c).

4.2 Implementation Details

Before starting the BC algorithm, we perform a preprocessing and variable fixing routine, as described in Section 3.4, in order to try to delete arcs from $\bar{G}$ and fix variables x and w at zero. The model initially contains the constraints that define $\mathcal{P}^*$, except for the exponentially sized sets: (3.4) for $|W| > 1$, (3.13), (3.14), (3.15) and (3.17). We provide a valid lower bound for the algorithm, given by the cost of the best solution that uses at most three customer vertices, for any vehicle (single truck or complete), without visiting satellite vertices.

We configure the solver to use different branching priorities for the variable sets. After solving a LPR and calling all separation routines without finding any violated cuts to add, if the solution is still fractional, the BC procedure chooses the variable to branch on based on the given priorities: variables with highest branching priority are

branched first. We used the following priority values, from the largest to the smallest priority:

- y: 4
- x and r: 3
- z_{k0}: 2
- z_{kl}: 1
- w: 0

This choice of priority values aims to guide the BC algorithm into determining the topology of the solutions, given by variables y, x and r, before the choice of vehicle, given by variables z. This decision is based on the fact that many walks in $\bar{G}$ will likely be infeasible for every solution that includes them, regardless of the choice of vehicle; therefore, this setting allows for such forbidden structures to be discarded earlier. Variables w, that are uniquely determined by variables x and z, have the lowest branching priority. In our experiments, using variable priorities allowed to obtain overall better results in average, in comparison to running the same algorithm without informing these priorities to CPLEX.

Chapter 5

Computational Experiments

In this chapter, we present computational results obtained with the BC algorithm introduced in Chapter 4. We also investigate the quality of the LPR bounds of our formulation and the strengthening capability of the families of valid inequalities we introduced earlier. First, we describe the set of instances used in the experiments, as well as the computational environment that was used. Then, we discuss the results obtained with different variations of the algorithm and suggest the one that seems to provide the best results.

5.1 Test Instances and Preprocessing Results

The test instances we used for the computational experiments were based on those introduced in [Solomon, 1987] for the VRPTW, and later adapted by Lin et al. [2011] for the TTRPTW [2011]. This benchmark instance set has been used in recent works regarding the TTRPTW [Parragh and Cordeau, 2017; Rothenbächer et al., 2018]. In this section, we describe how we adapted it for the PSTTRPTW.

The original Solomon instances were designed for the VRPTW with homogeneous fleet, i.e., in that problem, all vehicles have the same capacity for a given instance. This test bed is divided in six sets, that vary in spatial distribution of customers, time windows width and vehicle capacity. Instance sets R1 and R2 contain uniformly distributed customers; sets C1 and C2 have clustered customers, and sets RC1 and RC2 are a mix or random and clustered structures. Instance sets R1, C1 and RC1 are characterized by a short scheduling horizon, that is, vehicles have small capacity and are able to service relatively few customers in a feasible route. In contrast, instances R2, C2 and RC2 have large scheduling horizon, which permits many customers to be serviced by the same vehicle. Geographic coordinates for each vertex of these instances

are provided, and the vertices other than the depot are also given a supply, a time window and a service time.

Lin et al. [2011] used these instances to create a test set for the TTRPTW. For each instance, the authors identified truck-only customers in the following way: customers were sorted by distance to the nearest vertex, and the first p customers became truck customers, for $p \in \{25\%, 50\%, 75\%\}$. The remaining customers became vehicle customers. Therefore, three TTRPTW instances originated from each Solomon instance, one for each percentage value p of truck customers. Also, the authors set the truck and trailer capacities as half of the capacity of the vehicle in the original instance. For example, for a C1 instance, for which the vehicle capacity in the corresponding Solomon instance is 200, the truck capacity is 100 in the TTRPTW instance, and the trailer capacity is also 100.

In the problem solved by Lin et al. [2011], subtours start at vehicle customers, that also service as depots for trailers. There are, then, no dedicated satellite vertices. In order to adapt these instances to also include this type of vertex, we modified the TTRPTW instances as follows: for each truck customer i, we compute the five nearest vertices that are not truck customers. If any of these vertices is already a satellite, we move on to the next customer. Otherwise, let d be the distance from the customer to the farthest of the five vertices. We check if there is a truck customer already examined before i whose distance to i is smaller than d. If so, we skip customer i. Otherwise, we randomly select one of the five vehicle customers to become a satellite depot. For customers transformed into satellites, we only consider the positioning information provided by the instance; their time windows and supply values are ignored.

The TTRPTW instances do not provide all the parameters needed to define a PSTTRPTW instance (Table 2.1), like customer profits, vehicle travel costs, decoupling times and product transfer rates. We now describe how each of these quantities were calculated from the data outlined above. We set customer profits based on the provided supply values, in a way that customers with larger supplies would reward bigger profits. For each customer $i \in C$, we used the following formula:

$$p_i = \beta_i q_i,$$

where β_i is a random number uniformly distributed in $[0.8, 1.2]$, generated for each i.

We defined a *standard truck*, with capacity Q_0 given by the truck capacity in the original instance, and speed $s_{00} = 1$, also provided by the original instance. In order

to calculate the travel cost for this truck, we used the following expression:

$$\phi_{00} = \frac{1}{2}\frac{n_c \bar{q}}{s_{00} f_{\max}},$$

where $\bar{q}$ is the average supply of all customers in the instance, $f_{\max}$ is the largest time window closing value of all customers in the instance, and n_c is an estimation on the expected number of customers in a route in an optimal solution of the corresponding Solomon instance, an information provided in the instance set description [Solomon, 1987]. For instance sets R1, C1 and RC1, we used $n_c = 10$. For sets R2 and RC2, $n_c = 50$, and for instances C2, $n_c = 35$. This formula states that, if the standard truck travels during all the available time horizon, servicing the expected number of customers in that instance, its monetized travel cost will be approximately half of the profit it will collect.

Since the PSTTRPTW is defined in terms of a heterogeneous fleet, we create, for each instance, six additional trucks from the standard one, varying the following three parameters: the capacity, the speed and the travel cost. Each of these derived trucks have a 20% increment or decrement in two of those three conflicting parameters, like summarized by Table 5.1.

We also define a standard trailer, whose capacity is also Q_0, like in the original TTRPTW instances. Two additional trailers are created, one with a 20% capacity increase, and another with a 20% capacity decrease, like shown in Table 5.1.

Table 5.1: Calculation of truck and trailer parameters in the test instances.

Truck	Capacity	Speed	Fuel consumption
0	Q_0	s_{00}	ϕ_{00}
1	$1.2 \times Q_0$	$0.8 \times s_{00}$	ϕ_{00}
2	$0.8 \times Q_0$	$1.2 \times s_{00}$	ϕ_{00}
3	$1.2 \times Q_0$	s_{00}	$1.2 \times \phi_{00}$
4	$0.8 \times Q_0$	s_{00}	$0.8 \times \phi_{00}$
5	Q_0	$1.2 \times s_{00}$	$1.2 \times \phi_{00}$
6	Q_0	$0.8 \times s_{00}$	$0.8 \times \phi_{00}$

Trailer	Capacity
0	Q_0
1	$1.2 \times Q_0$
2	$0.8 \times Q_0$

Every instance has, therefore, 7 trucks and 3 trailers available, totaling 21 possible vehicle combinations. Differently from previous works, our model allows for different speed values when a vehicle is coupled or decoupled. Therefore, we also adapted

our instance set for that. For each vehicle combination $(k, l) \in \mathcal{V}$, we defined a factor $\alpha_{kl} \in (0.5, 1)$ that is a measure of the difference between the capacities of the decoupled truck $(k, 0)$ and the complete vehicle (k, l), when coupled:

$$\alpha_{kl} = \frac{1}{2}\left(\frac{Q_k}{Q_k + Q_l} + 1\right).$$

The bigger the trailer capacity is relatively to the truck's, the closest to 0.5 α_{kl} is. This factor is used calculate the speed and the travel cost of the coupled vehicle, in function of the speed and travel cost of the decoupled one, respectively:

$$s_{kl} = \alpha_{kl} s_{k0},$$

and

$$\phi_{kl} = \frac{\phi_{k0}}{\alpha_{kl}}.$$

From these expressions, the vehicle speed is decreased when coupled, and this decrease is influenced by the ratio between the capacities of the single truck and the complete vehicle, through factor α_{kl}. In contrast, travel cost is increased when a vehicle is coupled, by the same factor. For simplicity, we set the rent costs a_{kl} to zero for each vehicle. This setting nullifies the corresponding term in the objective function (3.1).

For transfer rates and coupling times, we opted for a unique value for all vehicles. We calculated these parameters based on the customer service time $\bar{t}$, that is a constant value for all customers, provided by the Solomon instances. The product transfer rate, applicable when the product is either collected at customers or transferred at satellite vertices, is given by

$$o = \frac{\bar{q}}{\bar{t}}.$$

Coupling/decoupling time τ, in turn, is calculated in a way that these operations take about one fifth of the time required to service an average customer:

$$\tau = \frac{\bar{t}}{5}.$$

Table 5.2 summarizes the instances used in all our experiments. The first column contains a name to identify the instances later on, in the tables of computational results. The next two columns show, respectively, the number of vertices and customers in the instance. The fourth and fifth columns display the number of truck and vehicle customers, respectively. In the sixth column, the number of satellite depots is shown. The following column represents the value of $\bar{c}$ for each instance, that bounds the

number of customers that can be serviced by a vehicle, explained in Section 3.2, and that provides the number of indexes κ of the r variables in the IP model.

Table 5.2: Test instances data.

Name	$\|V\|$	$\|C\|$	$\|C_{\mathcal{K}}\|$	$\|C_{\mathcal{V}}\|$	$\|S\|$	$\bar{c}$	Name	$\|V\|$	$\|C\|$	$\|C_{\mathcal{K}}\|$	$\|C_{\mathcal{V}}\|$	$\|S\|$	$\bar{c}$
C101-010_1	11	9	6	3	1	9	C101-025_1	26	24	6	18	1	17
C101-010_2	11	9	8	1	1	9	C101-025_2	26	23	12	11	2	17
C101-010_3	11	9	9	0	1	9	C101-025_3	26	23	18	5	2	17
C201-010_1	11	9	1	8	1	9	C201-025_1	26	23	6	17	2	23
C201-010_2	11	8	3	5	2	8	C201-025_2	26	22	12	10	3	22
C201-010_3	11	8	5	3	2	8	C201-025_3	26	23	18	5	2	23
R101-010_1	11	9	3	6	1	9	R101-025_1	26	22	6	16	3	20
R101-010_2	11	9	5	4	1	9	R101-025_2	26	21	12	9	4	19
R101-010_3	11	9	8	1	1	9	R101-025_3	26	23	18	5	2	20
R201-010_1	11	9	3	6	1	9	R201-025_1	26	23	6	17	2	23
R201-010_2	11	9	5	4	1	9	R201-025_2	26	22	12	10	3	22
R201-010_3	11	9	8	1	1	9	R201-025_3	26	23	18	5	2	23
RC101-010_1	11	8	3	5	2	8	RC101-025_1	26	23	6	17	2	15
RC101-010_2	11	8	4	4	2	8	RC101-025_2	26	22	12	10	3	15
RC101-010_3	11	9	6	3	1	9	RC101-025_3	26	24	18	6	1	15
RC201-010_1	11	8	3	5	2	8	RC201-025_1	26	23	6	17	2	23
RC201-010_2	11	9	4	5	1	9	RC201-025_2	26	22	12	10	3	22
RC201-010_3	11	9	7	2	1	9	RC201-025_3	26	23	18	5	2	23
C101-015_1	16	14	6	8	1	13	C101-050_1	51	47	12	35	3	24
C101-015_2	16	13	9	4	2	13	C101-050_2	51	44	25	19	6	23
C101-015_3	16	14	12	2	1	14	C101-050_3	51	47	37	10	3	24
C201-015_1	16	14	3	11	1	14	C201-050_1	51	44	12	32	6	44
C201-015_2	16	12	6	6	3	12	C201-050_2	51	43	25	18	7	43
C201-015_3	16	13	10	3	2	13	C201-050_3	51	47	37	10	3	47
R101-015_1	16	14	5	9	1	14	R101-050_1	51	43	12	31	7	26
R101-015_2	16	13	7	6	2	13	R101-050_2	51	44	25	19	6	26
R101-015_3	16	14	12	2	1	14	R101-050_3	51	46	37	9	4	27
R201-015_1	16	14	5	9	1	14	R201-050_1	51	43	12	31	7	43
R201-015_2	16	13	7	6	2	13	R201-050_2	51	44	25	19	6	44
R201-015_3	16	14	12	2	1	14	R201-050_3	51	47	37	10	3	47
RC101-015_1	16	13	5	8	2	12	RC101-050_1	51	45	12	33	5	21
RC101-015_2	16	12	6	6	3	12	RC101-050_2	51	44	25	19	6	22
RC101-015_3	16	14	10	4	1	12	RC101-050_3	51	47	37	10	3	22
RC201-015_1	16	13	5	8	2	13	RC201-050_1	51	46	12	34	4	46
RC201-015_2	16	12	6	6	3	12	RC201-050_2	51	46	25	21	4	46
RC201-015_3	16	14	11	3	1	14	RC201-050_3	51	47	37	10	3	47

In Table 5.3, we display the instance dimensions after setting the auxiliary graph $\bar{G}$ and the execution of the preprocessing procedure described in Section 3.4. The second column shows the number of vertices in $\bar{G}$, including the depot and the three satellite vertices created in $\bar{S}$ for each original satellite in $\bar{S}$. The next column shows the number of arcs in $\bar{A}$. The column m^* shows the number of arcs in the instance after preprocessing, and Δm is the percentage of arcs removed by preprocessing. Columns $|w|$ and $|w|^*$ represent the total of arcs that can be traversed by each vehicle combina-

tion, summed for every vehicle, before and after preprocessing, respectively. The value $|w|^*$ is the number of variables w that exist in the IP formulation after that procedure. Finally, $\Delta|w|$ is the reduction percentage in the w variable size due to preprocessing.

Table 5.3: Test instances size after setting the auxiliary graph and after preprocessing.

Instance	n	m	m^*	Δm	$\|w\|$	$\|w\|^*$	$\Delta\|w\|$
C101-010_1	13	134	97	27.61%	1302	980	24.73%
C101-010_2	13	130	93	28.46%	1008	749	25.69%
C101-010_3	13	128	91	28.91%	924	665	28.03%
C201-010_1	13	144	111	22.92%	2772	2009	27.53%
C201-010_2	15	162	137	15.43%	2128	1757	17.43%
C201-010_3	15	154	128	16.88%	1582	1337	15.49%
R101-010_1	13	140	97	30.71%	2058	1255	39.02%
R101-010_2	13	136	92	32.35%	1512	951	37.10%
R101-010_3	13	130	86	33.85%	1008	667	33.83%
R201-010_1	13	140	109	22.14%	2058	1541	25.12%
R201-010_2	13	136	103	24.26%	1512	1155	23.61%
R201-010_3	13	130	99	23.85%	1008	791	21.53%
RC101-010_1	15	162	139	14.20%	2128	1791	15.84%
RC101-010_2	15	158	134	15.19%	1834	1584	13.63%
RC101-010_3	13	134	100	25.37%	1302	967	25.73%
RC201-010_1	15	162	139	14.20%	2128	1797	15.55%
RC201-010_2	13	138	109	21.01%	1764	1386	21.43%
RC201-010_3	13	132	103	21.97%	1134	903	20.37%
C101-015_1	18	284	189	33.45%	3752	2431	35.21%
C101-015_2	20	308	227	26.30%	2884	2145	25.62%
C101-015_3	18	272	178	34.56%	2114	1431	32.31%
C201-015_1	18	290	204	29.66%	5138	3437	33.11%
C201-015_2	22	348	286	17.82%	3990	3255	18.42%
C201-015_3	20	304	230	24.34%	2632	2051	22.07%
R101-015_1	18	286	171	40.21%	4172	2087	49.98%
R101-015_2	20	316	221	30.06%	3514	2215	36.97%
R101-015_3	18	272	157	42.28%	2114	1207	42.90%
R201-015_1	18	286	207	27.62%	4172	2899	30.51%
R201-015_2	20	316	248	21.52%	3514	2723	22.51%
R201-015_3	18	272	195	28.31%	2114	1549	26.73%
RC101-015_1	20	324	249	23.15%	4312	3167	26.55%
RC101-015_2	22	348	281	19.25%	3990	3213	19.47%
RC101-015_3	18	276	182	34.06%	2492	1618	35.07%

Table 5.3: Test instances size after setting the auxiliary graph and after preprocessing.

Instance	n	m	m^*	Δm	$\|w\|$	$\|w\|^*$	$\Delta\|w\|$
RC201-015_1	20	324	261	19.44%	4312	3367	21.92%
RC201-015_2	22	348	293	15.80%	3990	3325	16.67%
RC201-015_3	18	274	200	27.01%	2282	1711	25.02%
C101-025_1	28	734	439	40.19%	12852	7227	43.77%
C101-025_2	30	786	516	34.35%	8974	5777	35.63%
C101-025_3	30	762	492	35.43%	6328	4214	33.41%
C201-025_1	30	810	569	29.75%	13132	8661	34.05%
C201-025_2	32	842	622	26.13%	9212	6736	26.88%
C201-025_3	30	762	518	32.02%	6328	4431	29.98%
R101-025_1	32	878	575	34.51%	13370	7264	45.67%
R101-025_2	34	890	616	30.79%	9408	6133	34.81%
R101-025_3	30	762	422	44.62%	6328	3445	45.56%
R201-025_1	30	810	599	26.05%	13132	9041	31.15%
R201-025_2	32	842	645	23.40%	9212	6930	24.77%
R201-025_3	30	762	551	27.69%	6328	4643	26.63%
RC101-025_1	30	810	513	36.67%	13132	6922	47.29%
RC101-025_2	32	842	566	32.78%	9212	5914	35.80%
RC101-025_3	28	710	380	46.48%	6048	3070	49.24%
RC201-025_1	30	810	603	25.56%	13132	9178	30.11%
RC201-025_2	32	842	649	22.92%	9212	6978	24.25%
RC201-025_3	30	762	555	27.17%	6328	4693	25.84%
C101-050_1	57	3042	1844	39.38%	50862	28053	44.84%
C101-050_2	63	3306	2264	31.52%	34902	23573	32.46%
C101-050_3	57	2892	1698	41.29%	23562	14092	40.19%
C201-050_1	63	3462	2540	26.63%	52374	35691	31.85%
C201-050_2	65	3404	2524	25.85%	35322	26002	26.39%
C201-050_3	57	2892	1840	36.38%	23562	15266	35.21%
R101-050_1	65	3586	2354	34.36%	52794	28141	46.70%
R101-050_2	63	3306	1984	39.99%	34902	18994	45.58%
R101-050_3	59	2990	1584	47.02%	24108	12449	48.36%
R201-050_1	65	3586	2835	20.94%	52794	38797	26.51%
R201-050_2	63	3306	2534	23.35%	34902	26169	25.02%
R201-050_3	57	2892	1989	31.22%	23562	16241	31.07%
RC101-050_1	61	3330	2001	39.91%	51912	23514	54.70%
RC101-050_2	63	3306	2047	38.08%	34902	19206	44.97%
RC101-050_3	57	2892	1463	49.41%	23562	11111	52.84%

Table 5.3: Test instances size after setting the auxiliary graph and after preprocessing.

Instance	n	m	m^*	Δm	$\|w\|$	$\|w\|^*$	$\Delta\|w\|$
RC201-050_1	59	3190	2328	27.02%	51408	34531	32.83%
RC201-050_2	59	3086	2216	28.19%	33936	23721	30.10%
RC201-050_3	57	2892	1990	31.19%	23562	16228	31.13%

5.2 Computational Environment

The BC algorithm described in this thesis was implemented in C++, and compiled with g++, version 9.2.1, with the optimization flag `-O3` enabled. We used IBM ILOG CPLEX [IBM ILOG CPLEX Optimizer, 2020], version 12.10, through calls to the Concert Technology Library, for solving the LPRs and managing the BC tree. The experiments ran in a machine equipped with a Intel Core i7-980 6 core, 1333MHz, with 24GB of RAM, and Ubuntu operating system.

5.3 Linear Programming Bounds

In this section, we discuss results related to LPR bounds provided by our approach. For that purpose, we consider two formulations: $\mathcal{P}^0$ and $\mathcal{P}^*$. Formulation $\mathcal{P}^0$ consists of inequalities (3.2)-(3.16), plus 2-matching inequalities (3.17). Formulation $\mathcal{P}^*$, as previously defined, includes all inequalities that define $\mathcal{P}^0$, plus valid inequalities (3.18)-(3.21), replacing constraints by their lifted versions (3.22)-(3.27) whenever possible. For all experiments described in this section, we used CPLEX for solving the LPRs, with all cuts and heuristics disabled. In order to obtain LPR bounds, we instructed CPLEX to solve the given model by Branch-and-Cut, limiting the BC tree to the root node only. This configuration results in CPLEX solving LPRs and executing the separation routines for the LPR solutions to look for violated cuts. Such cuts are iteratively added to the current LPR and, when no further violated cuts can be found, the execution finishes, without branching, no matter if the optimal solution if fractional or not. We then register the dual bound and execution time obtained.

For each formulation, we experimented two different separation approaches. In the first one, we do not use the fractional separation heuristic outlined by Algorithm 1. Therefore, only DCUTs (3.4) and 2-matching inequalities are separated for fractional points. IWECs (3.13)-(3.15) are only separated when the LPR solution is integer, and no DCUT violated by that solution is found (which did not happen for any of the test instances, for none of the formulations). In the second approach, we use the separation

heuristic for fractional points, whenever a DCUT cannot be found. In this case, for formulation $\mathcal{P}^0$, whenever an IWEC is found, we add the "original" version of the violated cut (3.13)-(3.15). For $\mathcal{P}^*$, we always try to lift the violated cut, as in equations (3.23)-(3.27). Note that, since Algorithm 1 is not exact, the LPR solutions obtained by the second approach may still violate some IWECs, and the obtained bounds are possibly weaker than the actual LPR bounds for the corresponding formulation, $\mathcal{P}^0$ or $\mathcal{P}^*$, if all constraints were considered in the LPRs.

Therefore, we experimented four different executions for each instance: two separation approaches for each formulation. We use the notation $\mathcal{P}_i^0$ and $\mathcal{P}_i^*$ when referring to results for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, respectively, without calling the fractional separation for IWECs, and we use $\mathcal{P}_f^0$ and $\mathcal{P}_f^*$ to represent results for these formulations using that heuristic. By considering a "basic" and a "strengthened" formulations, $\mathcal{P}^0$ and $\mathcal{P}^*$ respectively, we are interested in assessing the impact of the valid inequalities and lifting strategies proposed in this study in the quality of the solutions and execution times. The conception of two different separation strategies, using or not the fractional separation heuristic for IWECs, comes from the fact that, even though this routine finds valid cuts that can help strengthening the LPR bounds, it is possible that the number of inequalities added slows down the solution of the LPRs, resulting in a worse performance. Hence, we are interested in investigating if one of the choices is superior, or if it is possible to characterize situations in which some of them is best recommended.

We now present compiled results for each instance set, each one of which containing three instances, with different truck customer percentages, as described in Section 5.1. Detailed computational results regarding LPR bounds for each instance can be found in Section A.1, in the Appendix. Table 5.4 shows the time necessary for solving the root node of the BC tree, and the LPR gap obtained for each version of the algorithm, given by each of the four combinations of formulation and separation approach. If UB is the LPR bound for a given instance, and LB is the best known lower bound for that instance, then the LPR gap is given by $(UB - LB)/UB$. The values displayed in this table are summarized for each instance set. In the first column of that table, we identify the instance set. The following columns display, for each version of the algorithm, the average root time and average gap obtained for the three instances in each set.

In Table 5.5, we summarize the number of cuts generated for the second approach, in which the fractional separation for IWECs is used, for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$. As mentioned earlier, the first approach was not able to find violated IWECs for none of the formulations, hence it is not shown here. Like for Table 5.4, the values here are

Table 5.4: Average root times and LP gaps for each instance set obtained for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs.

Instance	$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$	
set	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)
C101-010	0.0	9.92	0.0	9.92	0.0	5.96	0.1	5.87
C201-010	0.1	2.43	0.1	2.39	0.1	2.11	0.2	2.11
R101-010	0.1	23.01	0.1	23.01	0.1	8.73	0.1	8.73
R201-010	0.1	0.62	0.1	0.62	0.0	0.07	0.0	0.07
RC101-010	0.1	28.86	0.1	28.71	0.1	25.15	0.1	25.01
RC201-010	0.1	5.87	0.1	5.87	0.1	4.99	0.1	4.99
C101-015	0.1	40.18	0.2	40.16	0.1	28.18	0.1	28.17
C201-015	0.3	1.28	0.3	1.28	0.3	1.11	0.8	1.11
R101-015	0.1	44.86	0.1	44.74	0.1	8.93	0.1	8.93
R201-015	0.1	2.67	0.2	2.67	0.1	1.13	0.1	1.13
RC101-015	0.3	51.86	0.3	51.86	0.3	38.77	0.3	38.42
RC201-015	0.3	12.02	0.3	12.02	0.3	11.22	0.4	11.21
C101-025	0.9	43.29	0.9	43.28	0.4	34.62	0.5	34.58
C201-025	0.8	14.87	1.2	14.85	0.3	14.86	0.5	14.78
R101-025	1.2	57.66	1.4	57.58	0.7	5.95	0.8	5.95
R201-025	1.0	15.73	1.2	15.72	0.9	13.83	1.7	13.74
RC101-025	1.1	51.16	1.2	51.14	1.0	42.08	1.1	41.99
RC201-025	0.9	35.18	1.1	35.18	1.2	34.66	1.4	34.66
C101-050	15.0	52.68	17.6	52.68	19.1	48.97	20.7	48.96
C201-050	29.4	58.87	38.5	58.82	43.5	55.81	73.5	55.77
R101-050	19.0	57.43	21.7	57.35	30.1	20.95	39.3	20.48
R201-050	13.1	81.49	16.9	81.46	37.3	80.78	42.3	80.78
RC101-050	11.2	55.35	12.0	55.31	18.3	49.56	22.0	49.50
RC201-050	11.9	86.74	13.8	86.73	14.8	85.71	15.7	85.71

also taken in average for each instance set. The first column identifies the instance set. In the next two columns, we show the following values for formulation $\mathcal{P}^0$: the average number of DCUTs generated for each instance set and the average number of IWECs for each instance set. The last two columns display the same values for formulation $\mathcal{P}^*$. Almost no 2-matching cuts were generated in this experiment, and therefore their numbers are not displayed.

Comparing results for $\mathcal{P}_i^0$ and $\mathcal{P}_f^0$ with $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$ in Table 5.4, we note that, as expected, adding valid inequalities to the formulation and lifting existing inequalities resulted in stronger LPR gaps, although increasing root times for bigger instances. This LPR gap reduction is more expressive for instances C1, R1 and RC1, which, as described earlier, have short scheduling horizon, and therefore, shorter solution routes. The overall LPR gap, however, is considerably lower for the "opposite" instances, with large scheduling horizon: C2, R2 and RC2, for both formulations and both separation

Table 5.5: Average number of cuts generated in the solution of LPRs in the root node of the BC procedure for each instance set, for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, when using the fractional separation heuristic for IWECs.

Instance set	$\mathcal{P}_f^0$		$\mathcal{P}_f^*$	
	DCUTs	IWECs	DCUTs	IWECs
C101-010	103.33	4.00	0.00	0.67
C201-010	258.00	0.33	52.33	8.00
R101-010	35.67	0.00	9.67	0.00
R201-010	40.33	0.00	14.00	0.00
RC101-010	169.00	1.33	76.67	1.67
RC201-010	148.67	0.00	117.00	0.00
C101-015	340.67	3.67	48.67	1.67
C201-015	683.00	0.33	142.00	28.67
R101-015	213.67	0.67	11.33	0.00
R201-015	105.33	0.00	20.33	0.00
RC101-015	191.33	1.33	148.00	3.00
RC201-015	276.33	0.33	186.00	1.00
C101-025	1479.67	2.00	432.00	2.00
C201-025	1367.00	38.00	197.33	26.33
R101-025	1102.00	1.33	27.00	0.33
R201-025	544.67	6.33	173.00	22.00
RC101-025	591.67	1.33	224.67	3.00
RC201-025	1020.67	1.00	452.00	1.00
C101-050	3136.67	2.67	636.33	3.67
C201-050	11433.67	109.67	5778.00	111.33
R101-050	5930.67	1.67	94.00	15.33
R201-050	3488.67	13.00	1095.00	3.33
RC101-050	2337.67	1.67	1077.67	4.00
RC201-050	3082.00	12.67	2007.00	0.33

approaches. It is also noteworthy that the use of the strengthened formulation considerably reduced the number of DCUTs, as shown in Table 5.5. The use of the fractional separation heuristic from Algorithm 1, as in $\mathcal{P}_f^0$ and $\mathcal{P}_f^*$, did not provide significant gap reduction for any instance, despite increasing root times, with many instances displaying no reduction at all when the heuristic is adopted. This observation is in accordance with the fact that not many IWECs are found in the root node of the BC algorithm, as seen in Table 5.5. Instances C2, that have clustered vertices and large scheduling horizon, are the ones that generate the majority of these cuts.

5.4 Results for the Branch-and-Cut Algorithm

We now analyze the results for the Branch-and-Cut procedure introduced in this work. We implemented the same versions of the algorithm described in the previous section: formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with two separation approaches, with and without the fractional separation heuristic from Algorithm 1. For all experiments described in this section, we used CPLEX for the solution of the LPRs and management of the BC search tree. Differently from the experiment conducted to estimate the LPR bounds in the preceding section, for this one, we enabled all CPLEX cuts and heuristics, under default settings. For each execution, we provided a time limit of 7200 seconds (2 hours) and, at the end, we registered the best globally valid lower and upper bounds and the corresponding duality gap implied by them.

As done for the LP results in the previous section, here we present compiled results for each instance set. In Section A.2, in the Appendix, detailed computational results regarding the BC algorithm for each instance can be found. In Tables 5.6 and 5.7, we show, for each instance set, the average execution times and average optimality gaps obtained, given by $(UB - LB)/UB$, as defined earlier. For instance sets with up to 26 vertices, we experimented the four versions of the algorithm, as displayed in Table 5.6. For larger instances, with 51 vertices, we only tried the two separation strategies for formulation $\mathcal{P}^*$, that produced better results, shown in Table 5.7. The first column of Table 5.6 identifies the instance set. The following two columns display the execution time and the percentage optimality gap for formulation $\mathcal{P}^0$ without using the separation heuristic from Algorithm 1. The subsequent six columns show the same pairs of values for formulation $\mathcal{P}^0$ with the use of the heuristic, $\mathcal{P}^*$ without the heuristic, and $\mathcal{P}^*$ with the heuristic, in that order. The columns of Table 5.7 follow the same pattern, but only for $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$, in that order.

Table 5.8 displays comparative information for both separation strategies applied to formulation $\mathcal{P}^*$: using the separation heuristic from Algorithm 1 ($\mathcal{P}_f^*$) and not using it ($\mathcal{P}_i^*$). The first column identifies the instance set. The next five columns contain the following values for $\mathcal{P}_i^*$: BC root time, that is, the time taken until the BC algorithm branches for the first time; BC root gap, which is the optimality gap available right before the algorithm branches on for the first time; total execution time; percentage optimality gap, and number of BC nodes solved. All values are taken in average for the three instances in each set. The last five columns of Table 5.8 display the same values for $\mathcal{P}_f^*$.

Table 5.9 compares the same two versions of the algorithm, in terms of number of IWECs generated by each approach, and the time spent in separation routines for

Table 5.6: Average solution times and optimality gaps for each instance set obtained for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs, for instance sets with up to 26 vertices.

Instance	$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$	
set	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)
C101-010	5390.6	7.73	5082.2	8.10	932.9	0.00	1453.5	0.00
C201-010	20.9	0.00	33.3	0.00	15.2	0.00	31.8	0.00
R101-010	0.7	0.00	0.5	0.00	0.3	0.00	0.3	0.00
R201-010	0.1	0.00	0.1	0.00	0.0	0.00	0.0	0.00
RC101-010	429.8	0.00	1991.4	0.00	175.8	0.00	225.7	0.00
RC201-010	0.6	0.00	0.7	0.00	0.5	0.00	0.5	0.00
C101-015	7200.0	43.96	7200.0	44.43	7200.0	31.12	7200.0	30.79
C201-015	119.0	0.00	198.9	0.00	122.7	0.00	303.0	0.00
R101-015	4800.2	19.58	4800.2	9.56	0.4	0.00	0.4	0.00
R201-015	0.7	0.00	0.8	0.00	0.7	0.00	0.7	0.00
RC101-015	7200.0	39.62	7200.0	41.30	2937.3	13.25	3134.2	12.97
RC201-015	7200.0	8.29	7200.0	6.89	7200.0	5.80	7200.0	7.24
C101-025	7200.0	42.91	7200.0	43.38	7200.0	37.80	7200.0	39.43
C201-025	7200.0	20.57	7200.0	26.13	7200.0	27.09	7200.0	39.13
R101-025	7200.0	48.58	7200.0	49.76	1.9	0.00	1.9	0.00
R201-025	7200.0	18.26	7200.0	11.61	7200.0	16.57	5782.1	12.27
RC101-025	7200.0	44.28	7200.0	44.39	5108.7	19.77	4938.3	20.10
RC201-025	7200.0	58.96	7200.0	59.92	7200.0	44.99	7200.0	53.11

Table 5.7: Average solution times and optimality gaps for each instance set obtained for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs, for instance sets with 51 vertices.

Instance set	$\mathcal{P}_i^*$		$\mathcal{P}_f^*$	
	t (s)	gap (%)	t (s)	gap (%)
C101-050	7200.0	48.72	7200.0	51.37
C201-050	7200.0	55.74	7200.0	63.99
R101-050	229.6	0.00	307.6	0.00
R201-050	7200.0	82.99	7200.0	81.67
RC101-050	7200.0	41.18	7200.0	40.98
RC201-050	7200.0	85.38	7200.0	87.57

these cuts. The first column identifies each instance set. The next three columns refer to $\mathcal{P}_i^*$. In the second and third columns, we show the average time spent in the separation for IWECs (Algorithm 2), in seconds and in percentage values with respect to the total execution time. The fourth column contains the average number of IWECs generated by that algorithm for the instances in the corresponding set. The following columns concern $\mathcal{P}_f^*$. For this version, we also show the average time spent in the fractional separation heuristic for IWECs, in seconds and percentage, in the fifth and sixth column, respectively. The two next columns display the average time spent in the integer separation for IWECs, also in absolute and relative values respectively. The

Table 5.8: Comparison between both separation strategies for formulation $\mathcal{P}^*$: average root times and gaps, optimality gaps and number of BC nodes explored, for each instance set.

Instance	$\mathcal{P}_i^*$					$\mathcal{P}_f^*$				
set	t_r	gap_r (%)	t	gap (%)	nodes	t_r	gap_r (%)	t	gap (%)	nodes
C101-010	0.1	28.88	932.9	-	288611.0	0.1	28.88	1453.5	-	207527.0
C201-010	0.5	45.16	15.2	-	8291.7	0.6	45.16	31.8	-	7463.7
R101-010	0.2	5.50	0.3	-	3.3	0.2	5.50	0.3	-	3.7
R201-010	0.0	-	0.0	-	0.0	0.0	-	0.0	-	0.0
RC101-010	0.5	42.60	175.8	-	49220.3	0.5	42.43	225.7	-	39221.3
RC201-010	0.3	35.29	0.5	-	25.3	0.3	35.29	0.5	-	22.0
C101-015	0.5	51.78	7200.0	31.12	225254.0	0.5	51.77	7200.0	30.79	157184.0
C201-015	1.2	54.53	122.7	-	17651.7	2.0	54.53	303.0	-	19979.3
R101-015	0.4	3.33	0.4	-	4.7	0.4	3.35	0.4	-	4.7
R201-015	0.5	39.71	0.7	-	10.0	0.6	39.71	0.7	-	7.3
RC101-015	2.0	45.94	2937.3	13.25	109919.7	2.2	45.76	3134.2	12.97	69513.3
RC201-015	1.4	64.65	7200.0	5.80	393444.0	1.3	64.70	7200.0	7.24	131424.0
C101-025	2.5	50.61	7200.0	37.80	66528.0	3.3	50.60	7200.0	39.43	35976.7
C201-025	2.2	70.96	7200.0	27.09	50814.0	5.1	70.91	7200.0	39.13	22572.0
R101-025	1.0	3.30	1.9	-	1.3	1.0	3.30	1.9	-	1.3
R201-025	9.9	73.40	7200.0	16.57	57164.7	12.3	73.24	5782.1	12.27	13296.3
RC101-025	5.0	48.96	5108.7	19.77	118251.3	5.2	48.83	4938.3	20.10	55326.0
RC201-025	8.4	77.06	7200.0	44.99	69264.3	8.7	77.06	7200.0	53.11	15454.0
C101-050	38.6	52.66	7200.0	48.72	12173.7	44.0	52.66	7200.0	51.37	5764.7
C201-050	104.7	83.13	7200.0	55.74	10623.0	135.1	83.10	7200.0	63.99	2405.7
R101-050	69.1	31.44	229.6	-	471.0	70.4	31.44	307.6	-	257.3
R201-050	102.3	84.43	7200.0	82.99	7117.0	130.5	84.40	7200.0	81.67	1631.7
RC101-050	51.4	56.87	7200.0	41.18	38955.0	55.3	56.81	7200.0	40.98	11934.7
RC201-050	60.7	88.16	7200.0	85.38	8287.3	69.0	88.16	7200.0	87.57	962.0

last two columns of the table contain the average number of IWECs generated by the fractional separation algorithm and the integer one, respectively.

Table 5.10 compares the same two approaches regarding the types of IWECs generated by each one, for each instance set. Recall that, in Chapter 3, we classified IWECs in three types: capacity (3.13), time window (3.14) and satellite scheduling (3.15) constraints. We recorded the number of times a cut from each of these types (or a lifted version of them) was generated by separation Algorithms 1 and 2. The first column in that table contains the instance set name. The next six columns refer to $\mathcal{P}_i^*$. In the second and third columns, we display the number of capacity cuts found, in average for that instance set, and the average percentage value of that number with respect to the total number of IWECs generated. The fourth and fifth columns show the absolute and relative numbers of time window cuts found, respectively, also in average values in each set. The sixth and seventh columns display these numbers for scheduling cuts. The subsequent six columns contain the same values for $\mathcal{P}_f^*$.

Finally, in Table 5.11, we compare the overall performance for both separation strategies, in terms of number of times each one obtained the best results for each instance size. The first column shows the number of vertices in the instances. From the second to the sixth column, we display the following values for $\mathcal{P}_i^*$, for the corresponding instance size: number of instances solved to optimality by this algorithm, number of

Table 5.9: Comparison between both separation strategies for formulation $\mathcal{P}^*$: average time spent in the IWEC separation routines and average number of IWECs generated, for each instance set.

Instance	$\mathcal{P}_i^*$			$\mathcal{P}_f^*$					
	t_i		IWEC	t_f		t_i		IWEC	
set	s	%	int	s	%	s	%	frac	int
C101-010	0.4	0.24	1087.7	192.8	10.00	0.3	0.17	2412.7	717.0
C201-010	0.1	0.39	127.0	9.2	28.94	0.1	0.31	570.3	101.3
R101-010	0.0	0.97	1.0	0.0	0.92	0.0	0.99	0.3	1.0
R201-010	0.0	1.46	0.0	0.0	0.00	0.0	1.42	0.0	0.0
RC101-010	0.7	0.33	773.3	60.1	19.46	0.3	0.20	1587.3	382.7
RC201-010	0.0	1.00	4.3	0.0	4.38	0.0	0.85	1.0	2.7
C101-015	10.7	0.15	6679.0	822.9	11.43	5.1	0.07	14740.7	2783.7
C201-015	0.8	0.38	351.7	77.9	26.88	0.5	0.27	2152.3	229.0
R101-015	0.0	0.59	1.0	0.0	2.71	0.0	0.60	1.0	1.0
R201-015	0.0	1.15	1.3	0.0	2.56	0.0	1.08	1.3	1.0
RC101-015	2.9	0.36	1010.3	511.5	20.65	0.7	0.12	6170.0	238.0
RC201-015	8.5	0.12	3125.7	1125.8	15.64	2.8	0.04	18788.3	1401.3
C101-025	29.4	0.41	4382.7	771.0	10.71	7.5	0.10	27026.7	1002.0
C201-025	45.8	0.64	4353.7	954.9	13.26	10.0	0.14	32533.0	989.3
R101-025	0.0	3.37	0.0	0.0	1.41	0.0	3.40	0.0	0.0
R201-025	31.5	0.56	2591.0	569.6	10.04	1.9	0.03	23385.7	145.3
RC101-025	17.9	0.44	2059.7	1224.0	25.09	4.3	0.10	11548.7	445.7
RC201-025	52.5	0.73	4497.0	894.6	12.43	1.8	0.03	41113.0	131.0
C101-050	136.3	1.89	1479.7	1537.9	21.36	12.5	0.17	12858.0	162.3
C201-050	241.1	3.35	1329.0	1879.5	26.10	8.2	0.11	15350.0	90.7
R101-050	5.7	1.31	20.0	85.8	22.27	2.8	1.02	251.0	10.3
R201-050	93.3	1.30	597.0	1421.2	19.74	2.4	0.03	10324.0	27.0
RC101-050	183.9	2.55	1582.7	2675.3	37.16	37.3	0.52	13865.0	352.0
RC201-050	49.8	0.69	446.0	812.8	11.29	0.1	0.00	12130.7	0.3

times this algorithm obtained the best lower bound, number of times it obtained the best upper bound, number of times it obtained the smallest optimality gap, and average optimality gap obtained by this algorithm. The last five columns contain the same information for $\mathcal{P}_f^*$. When both algorithms reached the same lower or upper bound or the same gap for a given instance, we counted that instance for both. The last line on that table summarizes these values for all instances.

As seen in Tables 5.6 and 5.7, there are more instance sets with zero average optimality gap for versions $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$ than for $\mathcal{P}_i^0$ and $\mathcal{P}_f^0$, which indicates that the use of valid and lifted inequalities allowed to solve to proven optimality some instances that were not solved with formulation $\mathcal{P}^0$. For instances still not solved with $\mathcal{P}^*$, optimality gaps were generally smaller with this formulation. Considering only instances with up to 26 vertices that were not solved by any of the versions of the algorithm, the average

Table 5.10: Comparison between both separation strategies for formulation $\mathcal{P}^*$: average number of IWECs of each type generated by each approach, for each instance set.

Set	$\mathcal{P}_i^*$						$\mathcal{P}_f^*$					
	Capacity		Time window		Scheduling		Capacity		Time window		Scheduling	
	#	%	#	%	#	%	#	%	#	%	#	%
C101-010	67	2.1	2394	73.4	802	24.6	114	1.2	4703	50.1	4572	48.7
C201-010	0	0.0	330	86.6	51	13.4	0	0.0	524	26.0	1491	74.0
R101-010	0	0.0	3	100.0	0	0.0	0	0.0	3	75.0	1	25.0
R201-010	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
RC101-010	138	5.9	1830	78.9	352	15.2	228	3.9	2923	49.5	2759	46.7
RC201-010	0	0.0	12	92.3	1	7.7	0	0.0	8	72.7	3	27.3
C101-015	821	4.1	15123	75.5	4093	20.4	1696	3.2	27042	51.4	23835	45.3
C201-015	0	0.0	841	79.7	214	20.3	0	0.0	1533	21.5	5611	78.5
R101-015	0	0.0	3	100.0	0	0.0	0	0.0	4	66.7	2	33.3
R201-015	0	0.0	4	100.0	0	0.0	0	0.0	3	42.9	4	57.1
RC101-015	52	1.7	2371	78.2	608	20.1	277	1.4	7976	41.5	10971	57.1
RC201-015	0	0.0	6925	73.9	2452	26.1	0	0.0	31450	51.9	29119	48.1
C101-025	310	2.4	9867	75.0	2971	22.6	587	0.7	37986	45.2	45513	54.1
C201-025	26	0.2	9065	69.4	3970	30.4	47	0.0	29726	29.6	70794	70.4
R101-025	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
R201-025	0	0.0	4996	64.3	2777	35.7	0	0.0	32454	46.0	38139	54.0
RC101-025	188	3.0	4851	78.5	1140	18.4	814	2.3	14211	39.5	20958	58.2
RC201-025	0	0.0	8756	64.9	4735	35.1	0	0.0	64531	52.2	59201	47.8
C101-050	17	0.4	3069	69.1	1353	30.5	66	0.2	10207	26.1	28788	73.7
C201-050	48	1.2	2486	62.4	1453	36.4	1060	2.3	13860	29.9	31402	67.8
R101-050	9	15.0	36	60.0	15	25.0	14	1.8	73	9.3	697	88.9
R201-050	0	0.0	888	49.6	903	50.4	0	0.0	5765	18.6	25288	81.4
RC101-050	75	1.6	4079	85.9	594	12.5	544	1.3	8589	20.1	33518	78.6
RC201-050	0	0.0	584	43.6	754	56.4	0	0.0	11574	31.8	24819	68.2

Table 5.11: Comparison between both separation strategies for formulation $\mathcal{P}^*$: number of instances for which each approach obtained the optimal solution, the best bounds and optimality gaps, and average optimality gap for each strategy.

n	$\mathcal{P}_i^*$					$\mathcal{P}_f^*$				
	#opt	$\#LB$	$\#UB$	#gap	$\langle$gap$\rangle$	#opt	$\#LB$	$\#UB$	#gap	$\langle$gap$\rangle$
10	18/18	18	17	18	0.00%	18/18	18	18	18	0.00%
15	11/18	17	17	17	8.36%	11/18	15	12	16	8.50%
25	4/18	13	11	13	24.37%	5/18	10	11	10	27.34%
50	3/18	15	13	15	52.33%	3/18	11	8	11	54.26%
Total	36/72	63	58	63	21.27%	37/72	54	49	55	22.53%

optimality gap provided by versions $\mathcal{P}_i^0$ and $\mathcal{P}_f^0$ is 36.1% for both, while $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$ reached average gaps of 29.2% and 32.3%, respectively.

In accordance to the LPR results discussed in Section 5.3, the improvement provided by the formulation strengthening is more notable for instances C1, R1 and RC1, that have short scheduling horizon. For example, for those instances, version $\mathcal{P}_i^*$ was able to solve 10 additional instances when compared to $\mathcal{P}_i^0$, and the average optimality gap for unsolved instances in these sets was reduced from 45.0% to 34.0%. Instances sets R1 and RC1 are, in fact, the ones for which most of the instances were solved to optimality: all instances from set R1 and all but five from set RC1 were solved to optimality by at least one version of the algorithm. In contrast, for instances sets with

large scheduling horizon, C2, R2 and RC2, the algorithm has more difficulty in lowering the dual bounds. For these sets, no additional instances were solved with version $\mathcal{P}_i^*$ in comparison to $\mathcal{P}_i^0$, and the average gap reduction was significantly smaller: from 28.8% to 25.3%. This may be explained by the fact that an IWEC involving a large number of variables is weaker and less informative. Instances with large scheduling horizon possibly require more of such cuts, and therefore are harder to solve by this approach.

Comparing both separation strategies, from versions $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$, the latter was able to solve one additional instance, as shown in Table 5.11. Aside from that instance, both were able to solve the same ones to optimality. Table 5.8 shows that the use of the fractional separation heuristic from Algorithm 1 does not significantly reduce root duality gaps, as also observed in the previous section, despite spending more time in the root node of the BC tree. In turn, the version $\mathcal{P}_i^*$ explores more BC nodes, which could be expected, since IWECs are not generated for fractional LPR solutions. For instances that were solved to optimality, Table 5.8 shows that version $\mathcal{P}_i^*$ took less time overall. In fact, the average solution time for instances solved to optimality by both versions was 193.8 seconds for $\mathcal{P}_i^*$, and 266.5 seconds for $\mathcal{P}_f^*$. Although the use of the fractional separation heuristic from Algorithm 1 has allowed to solve one more instance, for instances left unsolved, $\mathcal{P}_i^*$ was able to obtain better optimality gaps more often, as displayed by Table 5.11. This version of the algorithm also provided a slightly smaller average optimality gap overall.

The version $\mathcal{P}_f^*$ of the algorithm generates considerably more IWECs than $\mathcal{P}_i^*$, as exposed by Table 5.9. For larger instances, $\mathcal{P}_f^*$ expends a substantial portion of the execution time in the fractional separation for those cuts. As already observed, this computational effort is generally not compensated by better dual bounds. Because of the amount of cuts added, the LPRs are certainly harder to solve with this approach, which results in fewer BC nodes being explored. Version $\mathcal{P}_i^*$, on the other hand, is able to generate less IWECs and spends more computational effort in the BC tree management and in the solution of LPRs. Table 5.10 shows another distinction between the two versions: the majority of IWECs found by approach $\mathcal{P}_f^*$ are, in general, satellite scheduling cuts (3.15). For $\mathcal{P}_i^*$, time window cuts are the most frequently found.

5.5 Comments

In this chapter, we presented computational experiments for four versions of the BC algorithm introduced in this work. The results show that the valid and lifted inequal-

ities resulted in better algorithms. The fractional separation heuristic from Algorithm 1, on the other hand, allowed to solve only one further instance to optimality, and did not significantly help lowering optimality gaps. We can conclude that the version $\mathcal{P}_i^*$, that consists in the strengthened formulation in which IWECs are only separated for integer LPR solutions, performed best for this test bed.

Some aspects revealed by the results point at how the fractional heuristic could be best employed: the amount of time spent in that routine and the number of IWECs generated by it suggest that the heuristic could be used in a more moderate way. For instance, that procedure could only be called at the first BC nodes, or only every N nodes, where N is an adjustable parameter. Another possibility is to modify it in such a way that a cut is only added to the current LPR if it is sufficiently "sparse", that is, if it has few nonzero variable coefficients. Also, the fact that much more scheduling cuts are generated when that procedure is used might suggest an adaptation in which it only tries to find capacity and time windows cuts. Overall, we believe that the fractional separation needs to be further investigated, before it can be ruled out.

We also observed that, specially for larger instances, the algorithm has difficulty in finding feasible solutions. For six of those instances, it was not able to find any, and the final lower bound obtained was the one given by the starting solution we provided, with up to three customers. This strongly suggests our approach could benefit from a heuristic, both for providing better starting solutions, and for searching for improving feasible solutions during the Branch-and-Cut.

Chapter 6

Conclusion

In this thesis, we studied the Profitable Single Truck and Trailer Routing Problem with Time Windows (PSTTRPTW). We introduced an Integer Programming formulation for the problem, considering customer time windows, dedicated satellite depots, coupling/decoupling times and load-dependent transfer times. The model deals with a heterogeneous fleet of trucks and trailers, in which not only capacities and arc costs can be different for each vehicle, but also speed values can vary. In other words, we take into account the fact that one vehicle may take less to travel a given distance than another, due to the first being smaller, or more powerful. When this aspect is considered in the modeling, it is possible to exist routes for which some vehicles are able to reach a certain customer in time, while other vehicles are not. This particularity has not been tackled in previous works regarding TTRPs. The model we proposed has a compact number of variables, and a exponentially-sized constraint set. To the best of our knowledge, among exact approaches with these properties in the literature for TTRPTWs, this is the first one to allow for a solution to include multiple subtours from the same satellite depot, for a given vehicle.

In addition to the IP formulation, we described a preprocessing procedure for problem instances and introduced valid inequalities for the formulation, along with new inequalities that lift some of the original formulation constraints. Then, a Branch-and-Cut algorithm was introduced, with two separation procedures for infeasible walk elimination constraints: a heuristic for fractional LPR solutions, and an exact one, for integer solutions. Our computational experiments show an improvement in the algorithm with the strengthening of the formulation we described. However, the use of the separation heuristic did not result in better solution times or optimality gaps.

We believe that adding a heuristic method to our approach has a potential for solving larger and harder instances. Also, our algorithm can be incorporated into a

Column Generation scheme for solving the TTRPTW, in which the subproblem consists of finding an improving choice of vehicle and route, as described in Section 3.6. The resulting method can be contrasted with existing Column Generation approaches for the same problem, like the one described in [Rothenbächer et al., 2018].

In another direction for future works, there are still modeling aspects of TTRPs that can be considered, besides the ones handled here. In our model, the truck load collected in a subtour is always transferred to the trailer before coupling. A more elaborated model could consider the choice of performing or not a load transfer, allowing for a scenario in which that load is left in the truck, saving the load transfer time. Another challenging improvement would be to enable trailers to be shared by different trucks along their routes, as in the problem solved by Drexl [2014], but without restricting the number of subtours departing from a satellite. Both these modeling refinements would allow for more, potentially better, feasible solutions.

Bibliography

Ascheuer, N., Fischetti, M., and Grötschel, M. (2001). Solving the Asymmetric Travelling Salesman Problem with Time Windows by branch-and-cut. *Mathematical Programming*, 90:475–506.

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6):621–636.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An Exact Algorithm for the Two-Echelon Capacitated Vehicle Routing Problem. *Operations Research*, 61(2):298–314.

Bard, J., Huang, L., Dror, M., and Jaillet, P. (1998). A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30:821–834.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329.

Bartolini, E. and Schneider, M. (2020). A two-commodity flow formulation for the capacitated truck-and-trailer routing problem. *Discrete Applied Mathematics*, 275:3–18. ISSN 0166-218X. Advances in Optimization.

Batsyn, M. and Ponomarenko, A. (2014). Heuristic for a Real-life Truck and Trailer Routing Problem. *Procedia Computer Science*, 31:778–792.

Belenguer, J. M., Benavent, E., Martínez, A., Prins, C., Prodhon, C., and Villegas, J. G. (2016). A Branch-and-Cut Algorithm for the Single Truck and Trailer Routing Problem with Satellite Depots. *Transportation Science*, 50(2):735–749.

Bodin, L. and Levy, L. (2000). *Scheduling of Local Delivery Carrier Routes for the United States Postal Service*, pages 419–442. Springer US, Boston, MA.

Caramia, M. and Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, 61(7):1168–1180. ISSN 1476-9360.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51. ISSN 0305-0548.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489. ISSN 0377-2217.

Chopra, S., Gorres, E. R., and Rao, M. R. (1992). Solving the Steiner Tree Problem on a Graph Using Branch and Cut. *ORSA Journal on Computing*, 4(3):320–335.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research*, 54(4):756–766.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). The Vehicle Routing Problem. In Toth, P. and Vigo, D., editors, *VRP with Time Windows*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199. ISSN 0305-0548.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 2(4):393–410. ISSN 00963984.

Dantzig, G. B. (1957). Discrete-Variable Extremum Problems. *Operations Research*, 5(2):266–288.

Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing - Problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546. ISSN 0305-0548.

Dinitz, Y. (1970). Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Math. Dokl.*, 11:1277–1280.

Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Metodos Cuantitativos para la Economia y la Empresa*, 12:5–38.

Drexl, M. (2014). Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, 63.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, page 125.

Feillet, D., Dejax, P., and Gendreau, M. (2005). Traveling Salesman Problems with Profits. *Transportation Science*, 39(2):188–205. ISSN 00411655.

Ferland, J. A. and Michelon, P. (1988). The Vehicle Scheduling Problem with Multiple Vehicle Types. *The Journal of the Operational Research Society*, 39(6):577–583. ISSN 01605682, 14769360.

Gerdessen, J. C. (1996). Vehicle routing problem with trailers. *European Journal of Operational Research*, 93(1):135–147. ISSN 0377-2217.

Grötschel, M. and Holland, O. (1988). A Cutting Plane Algorithm for Minimum Perfect 2-Matchings. *Computing*, 39(4):327–344. ISSN 0010-485X.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228. ISSN 0305-0548.

Hoff, A. (2006). *Heuristics for rich vehicle routing problems*. PhD dissertation, Molde University College.

IBM ILOG CPLEX Optimizer (2020). `https://www.ibm.com/analytics/cplex-optimizer`. Visited in June 22, 2020.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330. ISSN 0305-0548. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

Kolen, A. W. J., Rinnooy Kan, A. H. G., and Trienekens, H. W. J. M. (1987). Vehicle Routing with Time Windows. *Operations Research*, 35(2):266–273.

Koskosidis, Y. A., Powell, W. B., and Solomon, M. M. (1992). An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints. *Transportation Science*, 26(2):69–85.

Lin, S.-W., Yu, V. F., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692. ISSN 0305-0548. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Lin, S.-W., Yu, V. F., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252. ISSN 0957-4174.

Mirmohammadsadeghi, S. and Ahmed, S. (2015). Memetic Heuristic Approach for Solving Truck and Trailer Routing Problems with Stochastic Demands and Time Windows. *Networks and Spatial Economics*, 15.

Padberg, M. and Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33(1):60–100.

Parragh, S. N. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44. ISSN 0305-0548.

Pasha, U., Hoff, A., and Løkketangen, A. (2014). A Hybrid Approach for Milk Collection Using Trucks and Trailers. *Annals of Management Science*, 3:85–108.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-Price-and-Cut for the Truck-and-Trailer Routing Problem with Time Windows. *Transportation Science*, 52(5):1174–1190.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2015). A Branch-and-Cut-and-Price Algorithm for the Two-Echelon Capacitated Vehicle Routing Problem. *Transportation Science*, 49(2):355–368.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909. ISSN 0305-0548. Part Special Issue: Optimization Days 2003.

Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41(4):469–488. ISSN 1572-9338.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265.

Tan, K., Chew, Y., and Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172(3):855–885. ISSN 0377-2217.

Torres, I., Cruz, C., and Verdegay, J. (2015). Solving the Truck and Trailer Routing Problem with Fuzzy Constraints. *International Journal of Computational Intelligence Systems*, 8:713–724.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2010). GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, 23(5):780–794. ISSN 0952-1976. Advances in metaheuristics for hard optimization: new trends and case studies.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334. ISSN 0305-0548.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244. ISSN 0377-2217.

Wang, C., Zhou, S., Gao, Y., and Liu, C. (2018). A self-adaptive bat algorithm for the truck and trailer routing problem. *Engineering Computations*, 35:00–00.

Appendix A

Detailed Computational Results

In this appendix, we present detailed computational results obtained with the experiments described in Chapter 5, complementing the results shown in that chapter. We recall that we experimented four versions of the algorithm, as described in Section 5.3: formulation $\mathcal{P}^0$ without the fractional separation heuristic ($\mathcal{P}^0_i$), $\mathcal{P}^0$ with fractional separation ($\mathcal{P}^0_f$), $\mathcal{P}^*$ without fractional separation ($\mathcal{P}^*_i$), and $\mathcal{P}^*$ with fractional separation ($\mathcal{P}^*_f$). Results are displayed for all instances in the test bed developed for this work, with the computational environment detailed in Section 5.2.

A.1 Detailed Linear Programming Results

This section shows computational results regarding LPR bounds obtained by the four versions of the algorithm, detailing the results displayed in Section 5.3.

In Table A.1, we display root node LPR bounds, for each instance in the test set. In the first two columns, we identify the test instances, characterized by different truck customer percentages, as described in Section 5.1. The first column refers to the instance set, and the second, to its identification inside that set. The following eight columns show the BC root execution time in seconds and the upper bound obtained for each version of the algorithm, in this order: $\mathcal{P}^0_i$, $\mathcal{P}^0_f$, $\mathcal{P}^*_i$, and $\mathcal{P}^*_f$. The last column, LB, shows the best lower bound known for each instance, provided by our computational experiments.

Table A.1: Linear Programming relaxation execution times and bounds obtained for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs.

Instance		$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$		LB
Set	id	t	UB	t	UB	t	UB	t	UB	
	1	0.0	122.24	0.0	122.24	0.0	114.96	0.0	114.96	106.81
C101-010	2	0.0	145.02	0.1	145.02	0.0	136.80	0.0	136.71	128.69
	3	0.0	136.28	0.0	136.28	0.1	134.82	0.1	134.53	128.26
	1	0.1	121.46	0.1	121.30	0.2	121.02	0.2	121.02	118.60
C201-010	2	0.1	119.86	0.1	119.86	0.2	119.55	0.2	119.55	117.35
	3	0.1	100.96	0.1	100.96	0.1	100.60	0.4	100.60	98.09
	1	0.1	44.22	0.1	44.22	0.0	34.75	0.1	34.75	33.50
R101-010	2	0.1	44.64	0.1	44.64	0.2	33.23	0.2	33.23	28.84
	3	0.0	31.80	0.0	31.80	0.0	31.80	0.0	31.80	28.80
	1	0.1	89.39	0.1	89.39	0.1	88.44	0.1	88.44	88.25
R201-010	2	0.1	89.75	0.1	89.75	0.0	89.23	0.0	89.23	89.23
	3	0.0	70.42	0.0	70.42	0.0	70.42	0.0	70.42	70.42
	1	0.1	93.52	0.1	93.52	0.1	88.28	0.1	88.23	68.09
RC101-010	2	0.1	73.50	0.1	73.30	0.1	70.27	0.1	69.96	48.91
	3	0.1	93.40	0.1	93.04	0.0	88.88	0.0	88.82	69.18
	1	0.1	159.02	0.1	159.02	0.1	157.60	0.1	157.60	148.56
RC201-010	2	0.1	191.37	0.1	191.37	0.1	190.13	0.1	190.13	181.34
	3	0.1	177.62	0.1	177.62	0.0	175.40	0.0	175.40	167.32
	1	0.1	209.99	0.1	209.99	0.1	176.44	0.1	176.44	127.46
C101-015	2	0.1	216.76	0.2	216.74	0.1	180.05	0.2	179.99	128.69
	3	0.1	215.91	0.2	215.80	0.1	178.78	0.1	178.78	128.26
	1	0.1	237.68	0.2	237.62	0.2	237.23	0.2	237.23	234.93
C201-015	2	0.4	199.42	0.4	199.42	0.4	199.10	1.5	199.10	196.63
	3	0.3	211.67	0.3	211.67	0.3	211.31	0.8	211.31	208.92
	1	0.1	79.64	0.1	79.06	0.1	41.23	0.1	41.23	36.07
R101-015	2	0.2	80.70	0.2	80.70	0.2	48.75	0.2	48.75	41.80
	3	0.1	65.56	0.1	65.56	0.0	44.81	0.0	44.81	44.81
	1	0.2	164.92	0.2	164.92	0.1	163.62	0.1	163.62	159.76
R201-015	2	0.2	156.81	0.2	156.81	0.2	153.74	0.2	153.74	152.17
	3	0.1	142.76	0.1	142.76	0.1	140.01	0.1	140.01	140.01
	1	0.4	151.32	0.4	151.32	0.3	141.33	0.4	138.41	70.44
RC101-015	2	0.4	136.42	0.4	136.42	0.4	90.37	0.4	90.37	65.47
	3	0.1	139.52	0.1	139.52	0.1	113.34	0.1	113.34	69.59
	1	0.4	252.57	0.4	252.57	0.2	250.81	0.4	250.75	216.70
RC201-015	2	0.3	261.49	0.3	261.49	0.5	259.86	0.7	259.86	235.87
	3	0.2	271.79	0.2	271.79	0.1	268.01	0.1	268.01	239.02
	1	1.2	233.11	1.2	233.11	0.4	207.39	0.5	207.08	156.77
C101-025	2	0.9	248.96	0.9	248.96	0.4	214.98	0.5	214.98	128.69

Instance		$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$		LB
Set	id	t	UB	t	UB	t	UB	t	UB	
	3	0.5	250.54	0.6	250.43	0.4	211.38	0.7	211.38	128.26
	1	0.7	421.50	1.2	421.46	0.4	421.36	0.7	421.03	346.68
C201-025	2	0.8	388.75	1.3	388.70	0.3	388.70	0.4	388.61	343.41
	3	0.8	404.06	1.0	403.85	0.3	404.06	0.4	403.39	342.63
	1	1.6	131.88	1.9	131.21	0.9	47.08	0.9	47.08	42.77
R101-025	2	1.2	119.62	1.4	119.57	1.3	56.46	1.3	56.46	51.55
	3	0.7	117.07	0.9	116.91	0.1	60.27	0.1	60.27	60.27
	1	1.0	247.67	1.2	247.66	1.0	243.98	1.2	243.68	205.82
R201-025	2	1.3	267.35	1.6	267.21	1.1	263.70	1.1	263.70	213.47
	3	0.7	232.69	0.8	232.68	0.8	224.35	2.8	223.93	209.09
	1	1.2	154.28	1.5	154.10	1.0	144.70	1.3	144.10	70.44
RC101-025	2	1.7	148.88	1.9	148.88	1.5	111.86	1.6	111.79	73.86
	3	0.3	141.27	0.3	141.27	0.4	122.65	0.5	122.56	72.41
	1	0.8	446.90	0.8	446.90	1.4	443.88	1.4	443.88	301.30
RC201-025	2	1.2	445.64	1.5	445.59	1.5	442.32	1.5	442.32	271.05
	3	0.8	449.42	0.9	449.42	0.8	445.01	1.3	445.01	297.56
	1	11.9	253.38	17.0	253.31	16.7	243.10	18.3	242.99	115.68
C101-050	2	28.5	256.65	31.1	256.64	36.4	233.70	39.3	233.70	124.29
	3	4.7	250.88	4.9	250.88	4.2	229.54	4.4	229.54	120.07
	1	36.6	742.48	37.9	742.46	61.2	695.34	75.7	695.01	354.08
C201-050	2	43.5	722.37	65.6	721.50	64.1	669.77	136.8	668.91	284.78
	3	8.0	747.68	11.9	745.75	5.2	693.48	8.0	692.73	271.26
	1	22.9	174.47	23.6	174.47	55.9	86.56	56.7	86.56	65.56
R101-050	2	22.2	158.73	28.6	157.95	26.9	87.92	49.4	86.42	70.68
	3	12.1	151.32	13.0	151.32	7.3	85.19	11.8	85.18	69.02
	1	17.1	521.60	20.2	521.13	67.5	501.65	71.9	501.65	94.33
R201-050	2	15.3	495.85	20.2	494.52	38.0	478.79	46.7	478.79	77.63
	3	6.8	473.39	10.3	472.42	6.5	455.63	8.2	455.63	103.10
	1	9.4	165.40	10.1	165.22	17.7	143.52	25.9	143.05	74.92
RC101-050	2	19.5	168.11	20.9	167.84	29.4	148.69	30.6	148.69	67.72
	3	4.7	161.78	5.0	161.78	7.9	146.09	9.5	146.08	78.26
	1	15.2	719.94	17.4	719.74	17.6	681.56	18.0	681.56	78.76
RC201-050	2	14.2	706.53	17.3	706.48	19.5	659.16	21.4	659.16	79.74
	3	6.2	716.57	6.6	716.13	7.4	655.17	7.7	655.17	125.89

Table A.2 shows the LP gap obtained for each instance. Recall that the LP gap for a given instance is given by $(UB - LB)/UB$, where UB is the LP bound obtained, and LB is the best known lower bound. Similarly to Table A.1, the first two columns of Table A.2 identify the instances. The following columns display, for each of the four combinations of formulation and separation approach, the gap obtained for each instance, and the average gap for the three instances in each set.

Table A.2: Linear Programming gaps for each instance obtained for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs, along with the average LP gap for each instance set.

Instance		$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$	
Set	id	%	avg(%)	%	avg(%)	%	avg(%)	%	avg(%)
	1	12.62		12.62		7.09		7.09	
C101-010	2	11.26	9.92	11.26	9.92	5.92	5.96	5.86	5.87
	3	5.89		5.89		4.86		4.66	
	1	2.36		2.23		2.00		2.00	
C201-010	2	2.09	2.43	2.09	2.39	1.84	2.11	1.84	2.11
	3	2.84		2.84		2.50		2.50	
	1	24.23		24.23		3.58		3.58	
R101-010	2	35.39	23.01	35.39	23.01	13.20	8.73	13.20	8.73
	3	9.42		9.42		9.42		9.42	
	1	1.28		1.28		0.22		0.22	
R201-010	2	0.58	0.62	0.58	0.62	0.00	0.07	0.00	0.07
	3	0.00		0.00		0.00		0.00	
	1	27.19		27.19		22.87		22.83	
RC101-010	2	33.46	28.86	33.28	28.71	30.40	25.15	30.10	25.01
	3	25.94		25.65		22.17		22.12	
	1	6.57		6.57		5.73		5.73	
RC201-010	2	5.24	5.87	5.24	5.87	4.62	4.99	4.62	4.99
	3	5.80		5.80		4.61		4.61	
	1	39.30		39.30		27.76		27.76	
C101-015	2	40.63	40.18	40.62	40.16	28.52	28.18	28.50	28.17
	3	40.60		40.56		28.26		28.26	
	1	1.16		1.13		0.97		0.97	
C201-015	2	1.40	1.28	1.40	1.28	1.24	1.11	1.24	1.11
	3	1.30		1.30		1.13		1.13	
	1	54.71		54.38		12.53		12.53	
R101-015	2	48.21	44.86	48.21	44.74	14.27	8.93	14.27	8.93
	3	31.65		31.65		0.00		0.00	
	1	3.13		3.13		2.36		2.36	
R201-015	2	2.96	2.67	2.96	2.67	1.02	1.13	1.02	1.13
	3	1.93		1.93		0.00		0.00	
	1	53.45		53.45		50.16		49.11	
RC101-015	2	52.01	51.86	52.01	51.86	27.55	38.77	27.55	38.42
	3	50.13		50.13		38.60		38.60	
	1	14.20		14.20		13.60		13.58	
RC201-015	2	9.80	12.02	9.80	12.02	9.23	11.22	9.23	11.21
	3	12.05		12.05		10.82		10.82	
	1	32.75		32.75		24.41		24.29	
C101-025	2	48.31	43.29	48.31	43.28	40.14	34.62	40.14	34.58

Instance		$\mathcal{P}_i^0$		$\mathcal{P}_f^0$		$\mathcal{P}_i^*$		$\mathcal{P}_f^*$	
Set	id	%	avg(%)	%	avg(%)	%	avg(%)	%	avg(%)
	3	48.81		48.78		39.32		39.32	
	1	17.75		17.74		17.72		17.66	
C201-025	2	11.66	14.87	11.65	14.85	11.65	14.86	11.63	14.78
	3	15.20		15.16		15.20		15.06	
	1	67.57		67.40		9.14		9.14	
R101-025	2	56.91	57.66	56.89	57.58	8.70	5.95	8.70	5.95
	3	48.52		48.45		0.00		0.00	
	1	16.90		16.89		15.64		15.54	
R201-025	2	20.15	15.73	20.11	15.72	19.05	13.83	19.05	13.74
	3	10.15		10.14		6.80		6.63	
	1	54.34		54.29		51.32		51.12	
RC101-025	2	50.39	51.16	50.39	51.14	33.97	42.08	33.93	41.99
	3	48.75		48.75		40.96		40.92	
	1	32.58		32.58		32.12		32.12	
RC201-025	2	39.18	35.18	39.17	35.18	38.72	34.66	38.72	34.66
	3	33.79		33.79		33.14		33.14	
	1	54.34		54.33		52.41		52.39	
C101-050	2	51.57	52.68	51.57	52.68	46.81	48.97	46.81	48.96
	3	52.14		52.14		47.69		47.69	
	1	52.31		52.31		49.08		49.05	
C201-050	2	60.58	58.87	60.53	58.82	57.48	55.81	57.43	55.77
	3	63.72		63.63		60.88		60.84	
	1	62.42		62.42		24.26		24.26	
R101-050	2	55.47	57.43	55.25	57.35	19.61	20.95	18.21	20.48
	3	54.39		54.39		18.98		18.97	
	1	81.92		81.90		81.20		81.20	
R201-050	2	84.34	81.49	84.30	81.46	83.79	80.78	83.79	80.78
	3	78.22		78.18		77.37		77.37	
	1	54.70		54.65		47.80		47.62	
RC101-050	2	59.72	55.35	59.65	55.31	54.46	49.56	54.46	49.50
	3	51.63		51.63		46.43		46.43	
	1	89.06		89.06		88.44		88.44	
RC201-050	2	88.71	86.74	88.71	86.73	87.90	85.71	87.90	85.71
	3	82.43		82.42		80.78		80.78	

In Table A.3, we summarize the number of cuts generated for the second approach, in which the fractional separation for IWECs is used, for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$. The first two columns identify the instances. In the next three columns, we show the following values for formulation $\mathcal{P}^0$: the number of DCUTs generated for each instance, the number of IWECs for each instance, and the average number of IWECs for each instance set. The last three columns display the same values for formulation $\mathcal{P}^*$.

Table A.3: Number of cuts generated in the solution of LPRs in the root node of the BC procedure, for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, when using the fractional separation heuristic for IWECs, along with the average number of IWECs for each instance set.

Instance		$\mathcal{P}_f^0$			$\mathcal{P}_f^*$		
Set	id	DCUTs	IWECs	⟨IWECs⟩	DCUTs	IWECs	⟨IWECs⟩
	1	285	0		0	0	
C101-010	2	25	12	4.00	0	1	0.67
	3	0	0		0	1	
	1	280	1		9	0	
C201-010	2	206	0	0.33	9	0	8.00
	3	288	0		139	24	
	1	58	0		16	0	
R101-010	2	49	0	0.00	13	0	0.00
	3	0	0		0	0	
	1	77	0		42	0	
R201-010	2	44	0	0.00	0	0	0.00
	3	0	0		0	0	
	1	197	2		119	1	
RC101-010	2	211	1	1.33	102	3	1.67
	3	99	1		9	1	
	1	73	0		144	0	
RC201-010	2	174	0	0.00	132	0	0.00
	3	199	0		75	0	
	1	490	0		121	0	
C101-015	2	215	4	3.67	25	3	1.67
	3	317	7		0	2	
	1	585	1		157	0	
C201-015	2	719	0	0.33	103	42	28.67
	3	745	0		166	44	
	1	245	2		5	0	
R101-015	2	244	0	0.67	29	0	0.00
	3	152	0		0	0	
	1	44	0		0	0	
R201-015	2	180	0	0.00	52	0	0.00
	3	92	0		9	0	
	1	231	1		243	8	
RC101-015	2	214	3	1.33	172	0	3.00
	3	129	0		29	1	

Instance		$\mathcal{P}_f^0$			$\mathcal{P}_f^*$		
Set	id	DCUTs	IWECs	⟨IWECs⟩	DCUTs	IWECs	⟨IWECs⟩
	1	349	1		180	2	
RC201-015	2	249	0	0.33	259	1	1.00
	3	231	0		119	0	
	1	2209	0		676	3	
C101-025	2	1134	0	2.00	109	1	2.00
	3	1096	6		511	2	
	1	2135	7		125	26	
C201-025	2	1513	3	38.00	9	3	26.33
	3	453	104		458	50	
	1	1191	1		26	0	
R101-025	2	738	2	1.33	55	1	0.33
	3	1377	1		0	0	
	1	472	1		70	2	
R201-025	2	405	12	6.33	299	0	22.00
	3	757	6		150	64	
	1	732	3		279	5	
RC101-025	2	877	1	1.33	331	2	3.00
	3	166	0		64	2	
	1	1109	0		612	0	
RC201-025	2	1372	2	1.00	448	0	1.00
	3	581	1		296	3	
	1	1558	5		944	9	
C101-050	2	4615	3	2.67	822	1	3.67
	3	3237	0		143	1	
	1	5593	3		10114	27	
C201-050	2	19073	101	109.67	5470	221	111.33
	3	9635	225		1750	86	
	1	2306	0		79	0	
R101-050	2	7563	4	1.67	149	43	15.33
	3	7923	1		54	3	
	1	3970	7		1564	5	
R201-050	2	3642	17	13.00	148	3	3.33
	3	2854	15		1573	2	
	1	2542	3		1324	7	
RC101-050	2	3430	1	1.67	761	2	4.00
	3	1041	1		1148	3	

Instance		$\mathcal{P}_f^0$			$\mathcal{P}_f^*$		
Set	id	DCUTs	IWECs	⟨IWECs⟩	DCUTs	IWECs	⟨IWECs⟩
	1	3432	3		1436	0	
RC201-050	2	4063	3	12.67	1867	1	0.33
	3	1751	32		2718	0	

A.2 Detailed Results for the Branch-and-Cut Algorithm

In this section presents computational results for the four versions of the BC algorithm, for each instance, complementing the compiled results shown in Section 5.3.

In Tables A.4 and A.5, we show the lower and upper bounds obtained, along with the corresponding optimality gap. Table A.4 displays results for instances with up to 26 vertices, for the four versions of the algorithm. The first two columns of Table A.4 identify the instance; the first one displays the set, and the second, the instance id inside that set. The following three columns display the lower bound, the upper bound and the percentage optimality gap for formulation $\mathcal{P}_i^0$. The subsequent nine columns show the same values for $\mathcal{P}_f^0$, $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$, in that order. When an instance was solved to optimality within the available time limit of 7200 seconds (2 hours), we display the symbol "-" for the optimality gap. Table A.5 presents those values for larger instances, with 51 vertices. The columns of this table follow the same pattern, but only for $\mathcal{P}_i^*$ and $\mathcal{P}_f^*$, in that order.

Table A.6 displays comparative information for both separation strategies applied to formulation $\mathcal{P}^*$: $\mathcal{P}_f^*$ and $\mathcal{P}_i^*$. The first two columns show the instance set and id. The next five columns contain the following values for $\mathcal{P}_i^*$: BC root time, BC root gap, total execution time, percentage optimality gap, and number of BC nodes solved. The last five columns of that table display the same values for $\mathcal{P}_f^*$.

Table A.4: Solution bounds obtained by the BC algorithm for formulations $\mathcal{P}^0$ and $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs, for instances with up to 26 vertices.

Instance		$\mathcal{P}_i^0$			$\mathcal{P}_f^0$			$\mathcal{P}_i^*$			$\mathcal{P}_f^*$		
Set	id	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)
C101-010	1	106.81	120.74	11.54	103.80	120.94	14.17	106.81	106.81	-	106.81	106.81	-
	2	126.38	143.02	11.64	128.69	143.20	10.13	128.69	128.69	-	128.69	128.69	-
	3	128.26	128.26	-	128.26	128.26	-	128.26	128.26	-	128.26	128.26	-
C201-010	1	118.60	118.60	-	118.60	118.60	-	118.60	118.60	-	118.60	118.60	-
	2	117.35	117.35	-	117.35	117.35	-	117.35	117.35	-	117.35	117.35	-
	3	98.09	98.09	-	98.09	98.09	-	98.09	98.09	-	98.09	98.09	-
R101-010	1	33.50	33.50	-	33.50	33.50	-	33.50	33.50	-	33.50	33.50	-
	2	28.84	28.84	-	28.84	28.84	-	28.84	28.84	-	28.84	28.84	-
	3	28.80	28.80	-	28.80	28.80	-	28.80	28.80	-	28.80	28.80	-
R201-010	1	88.25	88.25	-	88.25	88.25	-	88.25	88.25	-	88.25	88.25	-
	2	89.23	89.23	-	89.23	89.23	-	89.23	89.23	-	89.23	89.23	-
	3	70.42	70.42	-	70.42	70.42	-	70.42	70.42	-	70.42	70.42	-
RC101-010	1	68.09	68.09	-	68.09	68.09	-	68.09	68.09	-	68.09	68.09	-
	2	48.91	48.91	-	48.91	48.91	-	48.91	48.91	-	48.91	48.91	-
	3	69.18	69.18	-	69.18	69.18	-	69.18	69.18	-	69.18	69.18	-
RC201-010	1	148.56	148.56	-	148.56	148.56	-	148.56	148.56	-	148.56	148.56	-
	2	181.34	181.34	-	181.34	181.34	-	181.34	181.34	-	181.34	181.34	-
	3	167.32	167.32	-	167.32	167.32	-	167.32	167.32	-	167.32	167.32	-
C101-015	1	111.77	208.77	46.46	107.61	208.77	48.46	105.22	175.14	39.92	127.46	175.16	27.23
	2	128.69	216.01	40.42	119.35	216.11	44.77	128.69	178.19	27.78	108.34	178.27	39.23
	3	117.76	214.09	45.00	128.26	214.03	40.07	128.26	172.54	25.66	128.26	173.10	25.90

Instance		$\mathcal{P}_i^0$			$\mathcal{P}_f^0$			$\mathcal{P}_i^*$			$\mathcal{P}_f^*$		
Set	id	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)
C201-015	1	234.93	234.93	-	234.93	234.93	-	234.93	234.93	-	234.93	234.93	-
	2	196.63	196.63	-	196.63	196.63	-	196.63	196.63	-	196.63	196.63	-
	3	208.92	208.92	-	208.92	208.92	-	208.92	208.92	-	208.92	208.92	-
R101-015	1	36.07	48.24	25.23	36.07	37.69	4.29	36.07	36.07	-	36.07	36.07	-
	2	41.80	62.87	33.51	41.80	55.27	24.38	41.80	41.80	-	41.80	41.80	-
	3	44.81	44.81	-	44.81	44.81	-	44.81	44.81	-	44.81	44.81	-
R201-015	1	159.76	159.76	-	159.76	159.76	-	159.76	159.76	-	159.76	159.76	-
	2	152.17	152.17	-	152.17	152.17	-	152.17	152.17	-	152.17	152.17	-
	3	140.01	140.01	-	140.01	140.01	-	140.01	140.01	-	140.01	140.01	-
RC101-015	1	70.44	130.17	45.88	70.44	130.80	46.14	70.44	116.91	39.75	70.44	115.29	38.90
	2	65.47	118.04	44.54	58.36	117.96	50.52	65.47	65.47	-	65.47	65.47	-
	3	69.18	96.67	28.44	68.09	93.56	27.22	69.59	69.59	-	69.59	69.59	-
RC201-015	1	210.81	238.40	11.57	210.81	238.69	11.68	216.70	238.16	9.01	216.60	238.63	9.23
	2	235.87	249.09	5.31	235.87	249.20	5.35	235.87	248.89	5.23	235.87	249.23	5.36
	3	230.02	249.98	7.99	239.02	248.09	3.65	239.02	246.81	3.15	231.62	249.40	7.13
C101-025	1	155.82	231.52	32.70	156.77	231.14	32.17	134.91	206.30	34.61	128.00	205.96	37.85
	2	128.69	246.34	47.76	128.69	246.08	47.70	127.45	213.21	40.22	128.12	213.08	39.87
	3	127.96	247.32	48.26	123.26	247.78	50.25	128.26	208.83	38.58	124.77	209.94	40.57
C201-025	1	284.91	420.47	32.24	346.68	420.31	17.52	286.56	420.17	31.80	279.41	420.10	33.49
	2	343.41	387.65	11.41	226.04	387.60	41.68	252.95	387.44	34.71	129.10	387.40	66.68
	3	329.51	402.07	18.05	324.95	402.09	19.19	342.63	402.02	14.77	332.81	402.08	17.23
R101-025	1	42.77	114.93	62.78	42.77	114.64	62.69	42.77	42.77	-	42.77	42.77	-
	2	51.55	103.30	50.10	51.01	102.86	50.40	51.55	51.55	-	51.55	51.55	-

Instance		$\mathcal{P}_i^0$			$\mathcal{P}_f^0$			$\mathcal{P}_i^*$			$\mathcal{P}_f^*$		
Set	id	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)	LB	UB	gap (%)
	3	60.27	89.76	32.86	58.52	91.73	36.20	60.27	60.27	-	60.27	60.27	-
	1	149.44	238.23	37.27	202.81	237.50	14.61	198.41	236.78	16.21	205.82	236.63	13.02
R201-025	2	213.47	254.16	16.01	206.90	254.25	18.62	182.28	253.23	28.02	193.08	253.33	23.78
	3	209.09	212.26	1.49	209.09	212.46	1.59	198.73	210.25	5.48	209.09	209.09	-
	1	68.12	134.52	49.36	67.31	135.37	50.28	70.44	116.70	39.64	68.09	116.93	41.77
RC101-025	2	65.47	128.86	49.19	65.04	129.35	49.72	73.86	91.94	19.66	73.86	90.64	18.51
	3	68.72	104.58	34.29	68.77	102.93	33.19	72.41	72.41	-	72.41	72.41	-
	1	196.67	436.81	54.98	119.58	436.61	72.61	301.30	435.06	30.75	168.03	435.23	61.39
RC201-025	2	165.15	429.82	61.58	103.28	430.72	76.02	203.40	428.73	52.56	271.05	429.43	36.88
	3	170.30	429.22	60.32	297.56	432.05	31.13	207.37	429.06	51.67	167.31	429.78	61.07

Table A.5: Solution bounds obtained by the BC algorithm for formulation $\mathcal{P}^*$, with and without the use of the fractional separation heuristic for IWECs, for instances with 51 vertices.

Instance		$\mathcal{P}_i^*$			$\mathcal{P}_f^*$		
Set	id	LB	UB	gap (%)	LB	UB	gap (%)
C101-050	1	115.68	241.73	52.14	109.47	241.68	54.70
	2	124.29	231.89	46.40	124.29	232.42	46.52
	3	120.07	229.22	47.62	108.01	229.28	52.89
C201-050	1	352.30	693.28	49.18	354.08	693.64	48.95
	2	284.78	666.32	57.26	270.78	667.02	59.40
	3	271.26	691.42	60.77	113.05	690.35	83.62
R101-050	1	65.56	65.56	-	65.56	65.56	-
	2	70.68	70.68	-	70.68	70.68	-
	3	69.02	69.02	-	69.02	69.02	-
R201-050	1	94.33	492.07	80.83	74.60	493.30	84.88
	2	77.63	465.30	83.32	77.63	464.72	83.30
	3	67.48	445.00	84.84	103.10	445.28	76.85
RC101-050	1	71.91	120.44	40.29	74.92	121.52	38.35
	2	67.72	131.00	48.31	62.50	128.99	51.55
	3	78.26	120.27	34.93	78.26	116.87	33.04
RC201-050	1	78.76	675.61	88.34	78.76	677.33	88.37
	2	79.74	641.42	87.57	79.74	646.75	87.67
	3	125.89	636.70	80.23	85.65	641.75	86.65

Table A.6: Comparison between both separation strategies for formulation $\mathcal{P}^*$: root times and gaps, optimality gaps and number of BC nodes explored.

Instance		$\mathcal{P}_i^*$					$\mathcal{P}_f^*$				
Set	id	t_r	gap_r (%)	t	gap (%)	nodes	t_r	gap_r (%)	t	gap (%)	nodes
C101-010	1	0.2	44.43	1916.8	-	494609	0.2	44.43	3670.2	-	470877
	2	0.2	42.23	881.8	-	371224	0.2	42.22	690.2	-	151704
	3	0.0	-	0.1	-	0	0.0	-	0.1	-	0
C201-010	1	0.4	45.06	31.1	-	19493	0.4	45.06	69.9	-	17450
	2	0.6	41.53	9.5	-	3454	0.6	41.53	16.9	-	3284
	3	0.6	48.89	5.0	-	1928	0.8	48.89	8.7	-	1657
R101-010	1	0.0	-	0.1	-	0	0.0	-	0.1	-	0
	2	0.6	16.51	0.7	-	10	0.6	16.51	0.7	-	11
	3	0.0	-	0.0	-	0	0.0	-	0.0	-	0
R201-010	1	0.0	-	0.1	-	0	0.0	-	0.1	-	0
	2	0.0	-	0.0	-	0	0.0	-	0.0	-	0
	3	0.0	-	0.0	-	0	0.0	-	0.0	-	0
RC101-010	1	0.6	43.92	325.9	-	67277	0.8	43.72	356.4	-	58761
	2	0.4	50.66	201.0	-	80362	0.4	50.62	320.2	-	58877
	3	0.4	33.22	0.5	-	22	0.4	32.96	0.5	-	26
RC201-010	1	0.4	53.02	0.8	-	48	0.4	53.02	0.8	-	38
	2	0.5	52.84	0.6	-	28	0.5	52.84	0.7	-	28
	3	0.0	-	0.1	-	0	0.0	-	0.1	-	0

Instance		$\mathcal{P}_i^*$					$\mathcal{P}_f^*$				
Set	id	t_r	gap_r (%)	t	gap (%)	nodes	t_r	gap_r (%)	t	gap (%)	nodes
	1	0.4	51.14	(tl)	39.92	177434	0.5	51.14	(tl)	27.23	157290
C101-015	2	0.5	50.12	(tl)	27.78	155600	0.7	50.06	(tl)	39.23	131300
	3	0.4	54.10	(tl)	25.66	342728	0.3	54.10	(tl)	25.90	182962
	1	0.6	56.42	76.1	-	20464	0.7	56.42	396.4	-	37633
C201-015	2	1.6	51.42	284.2	-	30325	3.5	51.42	489.0	-	20421
	3	1.5	55.75	8.0	-	2166	1.8	55.76	23.5	-	1884
	1	0.4	1.68	0.5	-	3	0.5	1.72	0.5	-	3
R101-015	2	0.7	8.32	0.7	-	11	0.7	8.32	0.8	-	11
	3	0.0	-	0.0	-	0	0.0	-	0.0	-	0
	1	0.7	59.00	0.9	-	19	0.7	59.00	1.0	-	19
R201-015	2	0.9	60.13	1.0	-	11	1.1	60.13	1.1	-	3
	3	0.0	-	0.1	-	0	0.0	-	0.1	-	0
	1	2.7	53.89	(tl)	39.75	179175	3.2	53.18	(tl)	38.90	123100
RC101-015	2	2.7	36.45	1590.3	-	143326	2.8	36.45	2188.5	-	83403
	3	0.6	47.48	21.7	-	7258	0.6	47.64	14.2	-	2037
	1	1.3	66.60	(tl)	9.01	251174	1.3	66.60	(tl)	9.23	101006
RC201-015	2	2.0	60.12	(tl)	5.23	243285	1.9	60.14	(tl)	5.36	88568
	3	0.9	67.23	(tl)	3.15	685873	0.5	67.37	(tl)	7.13	204698
	1	1.5	53.32	(tl)	34.61	58582	2.3	53.32	(tl)	37.85	41529
C101-025	2	3.4	50.27	(tl)	40.22	79072	4.0	50.27	(tl)	39.87	35012
	3	2.6	48.24	(tl)	38.58	61930	3.7	48.20	(tl)	40.57	31389
	1	2.6	73.20	(tl)	31.80	55265	4.0	73.18	(tl)	33.49	18972
C201-025	2	2.0	71.09	(tl)	34.71	50380	6.5	71.08	(tl)	66.68	13408
	3	2.1	68.58	(tl)	14.77	46797	4.8	68.47	(tl)	17.23	35336
	1	3.0	9.90	3.1	-	4	3.1	9.90	3.2	-	4
R101-025	2	0.0	-	2.4	-	0	0.0	-	2.5	-	0
	3	0.0	-	0.1	-	0	0.0	-	0.1	-	0
	1	11.7	72.46	(tl)	16.21	50546	10.5	72.29	(tl)	13.02	13190
R201-025	2	13.7	75.00	(tl)	28.02	52300	14.4	74.94	(tl)	23.78	11253
	3	4.3	72.75	(tl)	5.48	68648	12.2	72.48	2946.1	-	15446
	1	8.0	55.47	(tl)	39.64	156025	7.3	55.21	(tl)	41.77	62239
RC101-025	2	5.1	46.72	(tl)	19.66	101663	5.7	46.72	(tl)	18.51	77361
	3	2.0	44.68	926.2	-	97066	2.8	44.55	414.7	-	26378
	1	8.1	78.96	(tl)	30.75	69893	9.2	78.96	(tl)	61.39	17584
RC201-025	2	13.2	76.51	(tl)	52.56	61100	12.7	76.51	(tl)	36.88	13466
	3	4.0	75.71	(tl)	51.67	76800	4.0	75.71	(tl)	61.07	15312
	1	31.4	54.83	(tl)	52.14	11064	36.6	54.82	(tl)	54.70	6398
C101-050	2	77.6	50.23	(tl)	46.40	9237	87.9	50.23	(tl)	46.52	2437
	3	6.8	52.93	(tl)	47.62	16220	7.5	52.93	(tl)	52.89	8459
	1	176.2	83.37	(tl)	49.18	6575	194.5	83.36	(tl)	48.95	1287
C201-050	2	124.9	80.96	(tl)	57.26	9011	181.5	80.93	(tl)	59.40	1690
	3	13.0	85.05	(tl)	60.77	16283	29.5	85.01	(tl)	83.62	4240
	1	138.5	31.58	535.9	-	1276	138.1	31.58	726.2	-	610
R101-050	2	39.5	33.36	88.1	-	67	44.4	33.36	131.8	-	62
	3	29.3	29.37	64.7	-	70	28.8	29.37	64.8	-	100
	1	135.5	84.95	(tl)	80.83	4006	164.8	84.94	(tl)	84.88	927
R201-050	2	127.6	83.41	(tl)	83.32	6652	184.7	83.34	(tl)	83.30	626
	3	43.8	84.93	(tl)	84.84	10693	41.9	84.93	(tl)	76.85	3342
	1	53.0	56.06	(tl)	40.29	27387	59.4	55.88	(tl)	38.35	7763
RC101-050	2	74.8	56.80	(tl)	48.31	27078	79.5	56.78	(tl)	51.55	7264
	3	26.5	57.76	(tl)	34.93	62400	27.0	57.76	(tl)	33.04	20777
	1	62.7	88.37	(tl)	88.34	4227	81.6	88.37	(tl)	88.37	691
RC201-050	2	85.8	87.68	(tl)	87.57	7195	91.3	87.68	(tl)	87.67	703

Instance		$\mathcal{P}_i^*$					$\mathcal{P}_f^*$				
Set	id	t_r	gap_r (%)	t	gap (%)	nodes	t_r	gap_r (%)	t	gap (%)	nodes
	3	33.5	88.43	(tl)	80.23	13440	34.1	88.43	(tl)	86.65	1492

Table A.7 compares the same two versions of the algorithm, in terms of number of IWECs generated by each approach, and the time spent in separation routines for these cuts. The first two columns identify each instance. The next three columns refer to $\mathcal{P}_i^*$. In the third and fourth columns, we show the time spent in the separation for IWECs (Algorithm 2), in seconds and in percentage values with respect to the total execution time. The fifth column contains the number of IWECs generated by that algorithm for each instance. The following columns concern $\mathcal{P}_f^*$. For this version, we also show the time spent in the fractional separation heuristic for IWECs, in seconds and percentage, in the sixth and seventh column, respectively. The two next columns display the time spent in the integer separation for IWECs, also in absolute and relative values respectively. The last two columns of the table contain the number of IWECs generated by the fractional separation algorithm and the integer one, respectively.

Table A.7: Comparison between both separation strategies for formulation $\mathcal{P}^*$: time spent in the IWEC separation routines and number of IWECs generated.

Instance		$\mathcal{P}_i^*$			$\mathcal{P}_f^*$					
		t_i		IWEC	t_f		t_i		IWEC	
Set	id	s	%	int	s	%	s	%	frac	int
	1	0.7	0.02	1884	465.5	12.68	0.5	0.01	5339	1121
C101-010	2	0.4	0.06	1378	112.8	16.34	0.3	0.05	1898	1030
	3	0.0	0.63	1	0.0	0.98	0.0	0.45	1	0
	1	0.1	0.16	225	20.1	28.75	0.1	0.14	1069	182
C201-010	2	0.1	0.41	82	4.9	28.86	0.1	0.37	296	72
	3	0.1	0.61	74	2.5	29.21	0.0	0.43	346	50
	1	0.0	1.27	1	0.0	0.00	0.0	1.28	0	1
R101-010	2	0.0	0.11	1	0.0	2.03	0.0	0.12	1	1
	3	0.0	1.52	1	0.0	0.72	0.0	1.58	0	1
	1	0.0	0.62	0	0.0	0.00	0.0	0.65	0	0
R201-010	2	0.0	1.90	0	0.0	0.00	0.0	1.91	0	0
	3	0.0	1.86	0	0.0	0.00	0.0	1.71	0	0
	1	1.3	0.36	1478	89.9	25.23	0.7	0.19	2250	719
RC101-010	2	0.7	0.21	840	90.5	28.26	0.4	0.11	2510	428
	3	0.0	0.42	2	0.0	4.88	0.0	0.30	2	1
	1	0.0	1.36	7	0.1	7.92	0.0	0.88	2	2
RC201-010	2	0.0	0.67	4	0.0	3.91	0.0	0.67	0	4
	3	0.0	0.97	2	0.0	1.32	0.0	1.00	1	2
	1	7.9	0.11	5707	761.5	10.58	3.1	0.04	15414	2103
C101-015	2	17.1	0.24	7214	992.7	13.79	8.6	0.12	15799	3073

Instance		$\mathcal{P}_i^*$			$\mathcal{P}_f^*$					
		t_i		IWEC	t_f		t_i		IWEC	
Set	id	s	%	int	s	%	s	%	frac	int
	3	7.1	0.10	7116	714.6	9.92	3.6	0.05	13009	3175
	1	0.5	0.13	458	104.2	26.29	0.5	0.12	3737	395
C201-015	2	1.8	0.36	503	122.6	25.07	0.8	0.17	2171	223
	3	0.2	0.66	94	6.9	29.28	0.1	0.50	549	69
	1	0.0	0.45	2	0.0	2.03	0.0	0.47	1	2
R101-015	2	0.0	0.34	1	0.0	6.10	0.0	0.33	2	1
	3	0.0	0.98	0	0.0	0.00	0.0	0.99	0	0
	1	0.0	0.72	2	0.0	4.73	0.0	0.71	0	2
R201-015	2	0.0	0.97	2	0.0	2.95	0.0	0.75	4	1
	3	0.0	1.75	0	0.0	0.00	0.0	1.78	0	0
	1	5.7	0.08	2050	1028.3	14.28	1.5	0.02	12342	508
RC101-015	2	2.8	0.13	799	502.8	22.97	0.5	0.02	5321	145
	3	0.1	0.86	182	3.5	24.70	0.0	0.31	847	61
	1	11.6	0.16	3714	1182.4	16.42	2.5	0.03	22989	650
RC201-015	2	11.2	0.16	2737	1242.1	17.25	1.1	0.01	15201	231
	3	2.6	0.04	2926	952.9	13.23	4.9	0.07	18175	3323
	1	15.3	0.21	3414	658.7	9.15	5.0	0.07	30603	944
C101-025	2	37.4	0.52	4789	938.2	13.03	11.3	0.16	27673	1194
	3	35.4	0.49	4945	715.9	9.94	6.1	0.09	22804	868
	1	39.2	0.54	3435	934.5	12.98	6.1	0.09	34844	472
C201-025	2	53.6	0.74	3574	835.9	11.61	4.4	0.06	30532	282
	3	44.6	0.62	6052	1094.3	15.20	19.3	0.27	32223	2214
	1	0.0	0.71	0	0.1	4.23	0.0	0.74	0	0
R101-025	2	0.0	1.68	0	0.0	0.00	0.0	1.75	0	0
	3	0.0	7.70	0	0.0	0.00	0.0	7.70	0	0
	1	31.6	0.44	2525	632.7	8.79	1.2	0.02	23812	92
R201-025	2	43.7	0.61	2723	757.4	10.52	3.5	0.05	35516	201
	3	19.1	0.65	2525	318.7	10.82	1.0	0.03	10829	143
	1	21.1	0.29	2410	1480.7	20.57	3.4	0.05	14272	366
RC101-025	2	30.0	0.42	2610	2084.3	28.95	8.8	0.12	15726	715
	3	2.5	0.61	1159	106.9	25.76	0.6	0.14	4648	256
	1	52.1	0.72	4252	1012.9	14.07	2.6	0.04	45388	185
RC201-025	2	66.0	0.92	4191	992.6	13.79	1.9	0.03	31062	112
	3	39.4	0.55	5048	678.5	9.42	0.9	0.01	46889	96
	1	123.1	1.71	1322	1764.6	24.51	16.6	0.23	14958	173
C101-050	2	171.8	2.39	892	1543.4	21.44	6.0	0.08	9727	29
	3	114.0	1.58	2225	1305.9	18.14	14.8	0.21	13889	285
	1	203.1	2.82	574	1871.6	25.99	2.8	0.04	13156	5
C201-050	2	350.4	4.87	1021	2372.4	32.95	2.2	0.03	10056	5
	3	169.9	2.36	2392	1394.5	19.37	19.4	0.27	22838	262

Instance		$\mathcal{P}_i^*$			$\mathcal{P}_f^*$					
		t_i		IWEC	t_f		t_i		IWEC	
Set	id	s	%	int	s	%	s	%	frac	int
R101-050	1	15.2	2.09	51	223.5	30.77	6.5	0.89	529	20
	2	1.2	0.87	4	20.9	15.87	1.2	0.89	148	4
	3	0.6	0.95	5	13.1	20.17	0.8	1.27	76	7
R201-050	1	101.3	1.41	219	2109.0	29.29	0.8	0.01	10652	1
	2	79.8	1.11	286	930.0	12.92	0.2	0.00	7804	0
	3	98.8	1.37	1286	1224.7	17.01	6.3	0.09	12516	80
RC101-050	1	251.0	3.49	1614	2825.8	39.25	58.0	0.81	13542	368
	2	217.3	3.02	1287	2944.8	40.90	29.1	0.40	10728	175
	3	83.3	1.16	1847	2255.2	31.32	24.7	0.34	17325	513
RC201-050	1	37.5	0.52	194	822.2	11.42	0.1	0.00	11124	0
	2	47.0	0.65	315	723.7	10.05	0.1	0.00	9251	0
	3	64.9	0.90	829	892.6	12.40	0.2	0.00	16017	1

Table A.8 compares the same two approaches regarding the three types of IWECs generated by each one, for each instance: capacity, time window and satellite scheduling constraints. The first two columns in that table contain the instance set and id. The next six columns refer to $\mathcal{P}_i^*$. In the third and fourth columns, we display the number of capacity cuts found and the percentage value of that number with respect to the total number of IWECs generated. The fifth and sixth columns show the absolute and relative numbers of time window cuts found, respectively. The seventh and eighth columns display these numbers for scheduling cuts. The subsequent six columns contain the same values for $\mathcal{P}_f^*$.

Table A.8: Comparison between both separation strategies for formulation $\mathcal{P}^*$: number of IWECs of each type generated by each approach.

Instance		$\mathcal{P}_i^*$						$\mathcal{P}_f^*$					
		Capacity		Time window		Scheduling		Capacity		Time window		Scheduling	
Set	id	#	%	#	%	#	%	#	%	#	%	#	%
C101-010	1	25	1.3	1389	73.7	470	24.9	33	0.5	3246	50.2	3181	49.2
	2	42	3.0	1004	72.9	332	24.1	80	2.7	1457	49.8	1391	47.5
	3	0	0.0	1	100.0	0	0.0	1	100.0	0	0.0	0	0.0
C201-010	1	0	0.0	196	87.1	29	12.9	0	0.0	343	27.4	908	72.6
	2	0	0.0	72	87.8	10	12.2	0	0.0	116	31.5	252	68.5
	3	0	0.0	62	83.8	12	16.2	0	0.0	65	16.4	331	83.6
R101-010	1	0	0.0	1	100.0	0	0.0	0	0.0	1	100.0	0	0.0
	2	0	0.0	1	100.0	0	0.0	0	0.0	1	50.0	1	50.0
	3	0	0.0	1	100.0	0	0.0	0	0.0	1	100.0	0	0.0
R201-010	1	0	-	0	-	0	-	0	-	0	-	0	-
	2	0	-	0	-	0	-	0	-	0	-	0	-
	3	0	-	0	-	0	-	0	-	0	-	0	-
RC101-010	1	121	8.2	1126	76.2	231	15.6	186	6.3	1473	49.6	1310	44.1
	2	17	2.0	702	83.6	121	14.4	42	1.4	1448	49.3	1448	49.3
	3	0	0.0	2	100.0	0	0.0	0	0.0	2	66.7	1	33.3
RC201-010	1	0	0.0	6	85.7	1	14.3	0	0.0	2	50.0	2	50.0
	2	0	0.0	4	100.0	0	0.0	0	0.0	4	100.0	0	0.0
	3	0	0.0	2	100.0	0	0.0	0	0.0	2	66.7	1	33.3

Instance		$\mathcal{P}_i^*$						$\mathcal{P}_f^*$					
		Capacity		Time window		Scheduling		Capacity		Time window		Scheduling	
Set	id	#	%	#	%	#	%	#	%	#	%	#	%
C101-015	1	206	3.6	4303	75.4	1198	21.0	334	1.9	9707	55.4	7476	42.7
	2	304	4.2	5268	73.0	1642	22.8	667	3.5	8426	44.6	9779	51.8
	3	311	4.4	5552	78.0	1253	17.6	695	4.3	8909	55.0	6580	40.7
C201-015	1	0	0.0	418	91.3	40	8.7	0	0.0	901	21.8	3231	78.2
	2	0	0.0	341	67.8	162	32.2	0	0.0	505	21.1	1889	78.9
	3	0	0.0	82	87.2	12	12.8	0	0.0	127	20.6	491	79.4
R101-015	1	0	0.0	2	100.0	0	0.0	0	0.0	2	66.7	1	33.3
	2	0	0.0	1	100.0	0	0.0	0	0.0	2	66.7	1	33.3
	3	0	-	0	-	0	-	0	-	0	-	0	-
R201-015	1	0	0.0	2	100.0	0	0.0	0	0.0	2	100.0	0	0.0
	2	0	0.0	2	100.0	0	0.0	0	0.0	1	20.0	4	80.0
	3	0	-	0	-	0	-	0	-	0	-	0	-
RC101-015	1	19	0.9	1626	79.3	405	19.8	79	0.6	5861	45.6	6910	53.8
	2	9	1.1	593	74.2	197	24.7	23	0.4	1921	35.1	3522	64.4
	3	24	13.2	152	83.5	6	3.3	175	19.3	194	21.4	539	59.4
RC201-015	1	0	0.0	2496	67.2	1218	32.8	0	0.0	13414	56.7	10225	43.3
	2	0	0.0	1970	72.0	767	28.0	0	0.0	5902	38.2	9530	61.8
	3	0	0.0	2459	84.0	467	16.0	0	0.0	12134	56.4	9364	43.6
C101-025	1	29	0.8	2676	78.4	709	20.8	124	0.4	19906	63.1	11517	36.5
	2	75	1.6	3645	76.1	1069	22.3	222	0.8	12731	44.1	15914	55.1
	3	206	4.2	3546	71.7	1193	24.1	241	1.0	5349	22.6	18082	76.4

Instance		$\mathcal{P}_i^*$						$\mathcal{P}_f^*$					
		Capacity		Time window		Scheduling		Capacity		Time window		Scheduling	
Set	id	#	%	#	%	#	%	#	%	#	%	#	%
C201-025	1	3	0.1	2389	69.5	1043	30.4	9	0.0	9039	25.6	26268	74.4
	2	0	0.0	1734	48.5	1840	51.5	0	0.0	7664	24.9	23150	75.1
	3	23	0.4	4942	81.7	1087	18.0	38	0.1	13023	37.8	21376	62.1
R101-025	1	0	-	0	-	0	-	0	-	0	-	0	-
	2	0	-	0	-	0	-	0	-	0	-	0	-
	3	0	-	0	-	0	-	0	-	0	-	0	-
R201-025	1	0	0.0	1504	59.6	1021	40.4	0	0.0	8415	35.2	15489	64.8
	2	0	0.0	1650	60.6	1073	39.4	0	0.0	20143	56.4	15574	43.6
	3	0	0.0	1842	73.0	683	27.0	0	0.0	3896	35.5	7076	64.5
RC101-025	1	13	0.5	1940	80.5	457	19.0	138	0.9	5678	38.8	8822	60.3
	2	38	1.5	1974	75.6	598	22.9	189	1.1	6986	42.5	9266	56.4
	3	137	11.8	937	80.8	85	7.3	487	9.9	1547	31.5	2870	58.5
RC201-025	1	0	0.0	3082	72.5	1170	27.5	0	0.0	25534	56.0	20039	44.0
	2	0	0.0	2401	57.3	1790	42.7	0	0.0	12480	40.0	18694	60.0
	3	0	0.0	3273	64.8	1775	35.2	0	0.0	26517	56.4	20468	43.6
C101-050	1	4	0.3	903	68.3	415	31.4	13	0.1	4742	31.3	10376	68.6
	2	6	0.7	464	52.0	422	47.3	8	0.1	2444	25.1	7304	74.9
	3	7	0.3	1702	76.5	516	23.2	45	0.3	3021	21.3	11108	78.4
C201-050	1	10	1.7	271	47.2	293	51.0	55	0.4	7302	55.5	5804	44.1
	2	23	2.3	444	43.5	554	54.3	195	1.9	338	3.4	9528	94.7
	3	15	0.6	1771	74.0	606	25.3	810	3.5	6220	26.9	16070	69.6

Instance		$\mathcal{P}_i^*$						$\mathcal{P}_f^*$					
		Capacity		Time window		Scheduling		Capacity		Time window		Scheduling	
Set	id	#	%	#	%	#	%	#	%	#	%	#	%
R101-050	1	8	15.7	28	54.9	15	29.4	7	1.3	52	9.5	490	89.3
	2	1	25.0	3	75.0	0	0.0	4	2.6	7	4.6	141	92.8
	3	0	0.0	5	100.0	0	0.0	3	3.6	14	16.9	66	79.5
R201-050	1	0	0.0	83	37.9	136	62.1	0	0.0	3215	30.2	7438	69.8
	2	0	0.0	83	29.0	203	71.0	0	0.0	87	1.1	7717	98.9
	3	0	0.0	722	56.1	564	43.9	0	0.0	2463	19.6	10133	80.4
RC101-050	1	58	3.6	1324	82.0	232	14.4	114	0.8	4419	31.8	9377	67.4
	2	1	0.1	1044	81.1	242	18.8	211	1.9	1480	13.6	9212	84.5
	3	16	0.9	1711	92.6	120	6.5	219	1.2	2690	15.1	14929	83.7
RC201-050	1	0	0.0	105	54.1	89	45.9	0	0.0	4502	40.5	6622	59.5
	2	0	0.0	111	35.2	204	64.8	0	0.0	232	2.5	9019	97.5
	3	0	0.0	368	44.4	461	55.6	0	0.0	6840	42.7	9178	57.3