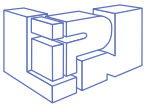
Solving strategic and tactical optimization problems in city logistics

Paolo Gianessi



UNIVERSITÉ PARIS 13

UNIVERSITÉ PARIS 13

THÈSE DE DOCTORAT

Solving Strategic and Tactical Optimization Problems in City Logistics

Thèse présentée par

Paolo GIANESSI

pour l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ PARIS 13
Spécialité: *Informatique*

Laboratoire de Recherche:

LABORATOIRE D'INFORMATIQUE DE PARIS NORD

soutenue le 26 novembre 2014

Jury:

Rapporteurs:		
Teodor Gabriel Crainic	Professeur	Université du Québec à Montréal
Frédéric Semet	Professeur	École Centrale de Lille
Examinateurs:		
Alberto Ceselli	Assistant Professor	Università degli Studi di Milano
Dominique Feillet	Professeur	École des Mines de Saint-Étienne
Christian Prins	Professeur	Université de Technologie de Troyes
Directeur de Thèse:		
Roberto Wolfler Calvo	Professeur	Université Paris 13, SPC
Co-encadrants de Thèse:		
Laurent Alfandari	Professeur	ESSEC Business School
Lucas Létocart	Maître de conférences HDR	Université Paris 13, SPC

Abstract

Solving Strategic and Tactical Optimization Problems in City Logistics

Urban freight transport is a matter of increasing concern in the economic, commercial, social and environmental operations of our cities, due to the constantly increasing growth and urbanization of the civilization. An improved management of the traffic related to the freight transport can have a positive impact in many respects: security, congestion of the road network, noise and air pollution, costs. *City Logistics* studies the dynamic management of urban freight transport in order to deliver distribution systems solutions that may be suitable for both the community and freight carriers.

This thesis originates from the *ANR* Project *MODUM*, which proposes a freight distribution system based on a ring of *Urban Distribution Centers (UDCs)* located in the outskirts of a city. In the first part, this system is studied from both a strategic and a tactical point of view. The *Multicommodity-Ring Location Routing Problem (MRLRP)* considers long-term decisions, i.e. the installation of the UDCs and the ring connection, without disregarding more tactical aspects. The MRLRP has been tackled by three solution methods, which proved effective on a large set of test instances.

In the second part of the thesis, the *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)* is studied. The VRPIRF is a more tactical problem that arises in City Logistics each time both the *multi-trip* and the *multi-depot* features, i.e. the possibility for a vehicle to be reloaded at one of a set of facilities, are present. Several exact algorithms, namely two of type Branch&Cut and two of type Branch& Price, have been developed for this problem. Computational experiments on benchmark instances taken from the literature have been conducted to assess their performance, leading to very promising results.

Résumé

Optimisation Stratégique et Tactique en Logistique Urbaine

L'efficacité du transport des marchandises en ville est un sujet complexe, préoccupant les autorités locales depuis de nombreuses années, et sur lequel de nombreuses travaux de recherche ont vu le jour dans la dernière décennie. Les enjeux sont immenses, une meilleure organisation du trafic devant permettre de limiter la congestion du réseau urbain et la pollution atmosphérique ou sonore, ainsi que de d'augmenter la sécurité routière, et minimiser les coûts liés au transport. La *Logistique Urbaine* vise à concevoir des systèmes de distribution de marchandises en ville permettant d'acheminer les flux dans les meilleurs conditions (économique et de gestion urbaine) à la fois pour la communauté et les transporteurs.

Le contexte de cette thèse est offert par le projet *ANR MODUM*, qui propose un système de distribution basé sur un anneau de Centres de Distribution Urbains (CDU) situés aux alentours d'une ville. La première partie étude ce système d'un point de vue strategique et tactique. Le *Multicommodity-Ring Location Routing Problem (MRLRP)* aborde les decisions concernant l'installation et connexion en anneau des CDU, en considerant d'une façon simplifiée les détails plus tactiques. Trois méthodes pour le MRLRP ont été developpées et testées sur un jeu d'instances exhaustif, se révélant très efficaces.

La deuxième partie porte sur le *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)*, un problème plus tactique qui se produit dans un système logistique lors que les véhicules peuvent se recharger auprès de l'un parmi un ensemble de dépôts et ainsi effectuer plusieurs tournées au long d'une journée. Deux algorithm de type Branch & Cut et deux de type Branch & Price ont été developpés et testées. Les résultats obtenus sur des jeux d'instances tirés de la literature sont prometteurs.

Contents

Introduction

Introduction

Without doubt, freight transport is a key element of modern societies, as it ensures the flow of goods from production to distribution points so as to make them available to consumers. This is even more true for what concerns cities, where the gap between production and demand of consumption goods is the highest.

The efficiency of freight transport relies on *Logistics* (more specifically *Freight Transport Logistics*), defined as *…the planning, organisation, management, execution and control of freight transport operations…* that *…can increase the efficiency of individual modes of transport and their combinations…* and *…can help disconnect transport growth […] from the harmful external effects that it produces (emissions, accidents and congestion)* [Commission of the European Communities, 2006]. The concept of Logistics is specialized by that of *City Logistics*. This latter dates back to the first years 2000 and is defined as *…the process for totally optimizing the logistics and transport activities by private companies with support of advanced information systems in urban areas while considering the traffic environment, the traffic congestion and energy consumption, the traffic safety and the energy savings within the framework of a market economy* [Taniguchi et al., 2001].

As to cities, we are witnessing to an increasing urbanization of the civilization. If we consider the simultaneous growth of the global population, it is easy to understand how the transportation of goods in urban areas is a delicate matter. It is not by chance that in the last decade this concern has become more and more the object of the attention of authorities, the investments of private companies, and the research of academia.

The achievement of an improved traffic management calls for the design of *new models of freight distribution network*, i.e.:

- the conception of new forms of freight distribution;
- the analysis of the organizational, economic and environmental contexts in which such forms should take place – and therefore their viability;

- the investigation of how to obtain high-performing transportation systems while maximizing the benefits in the various concerned respects and for all the implied actors (practitioners, community);
- the assessment of the system impact in economic, environmental and social terms.

Indeed, the stakes are considerable, as an improved management of the traffic related to the freight transport can have a positive impact in terms of security, congestion of the road network, noise and air pollution, costs – which are all sensitive subjects.
One of the most promising logistic strategies is represented by *mutualized distribution systems*, in which shared stocking spaces are made available to the different private carriers, and the shipment operations to final customers are performed by a single -often public- operator, in order to rationalize the usage of spaces, achieve higher load rates and therefore reduce the travelled distance, the costs and the polluction factors. In this sense, the usage of *Urban Distribution Centers (UDCs)* appears to be encouraging, as one of their main purposes is precisely the *mutualization* of flows of merchandise. A UDC is a bulk-breaking point similar to *hubs* in air transport, where goods can be received, processed, consolidated and then forwarded to their destinations. UDCs are a suitable solution both for the community, as they help rationalize the flows of merchandise on the territory, but also for those carriers that do not have a benefit from solving last kilometer issues with their own fleet.
To this day, the tentatives that have been conducted to verify the effectiveness of UDCs are few, as installing UDCs requires a series of conditions, notably the presence of regulation constraints to forbid the access to the city center. Moreover, these trials have had controversial results, mainly due to economic reasons: big logistic operators normally prefer to control their supply chain; further, the additional bulk-break stage introduced by UDCs is not always compensated by the achieved transportation economies.
Nevertheless, the growing concern of the public opinion for the environment and the question of sustainable development, along with the latest progresses made by the communication technologies and the consequent dropping of their costs, make think that the time is ready to deploy such a solution on an important scale.

Motivation of this PhD Thesis. The MODUM Project

This is the challenge taken up by the *ANR (Agence Nationale de la Recherche)* Project *MODUM (Mutualisation et Optimisation de la distribution Urbaine de Marchandises)* [Agence Nationale de la Recherche, 2010]. The purpose of MODUM is the study of a freight distribution system for urban areas based on a *ring of UDCs* located in the outskirts of a city, and the design and implementation of a *Decision Support System*

(DSS) that would allow potentially interested subjects to consider the adoption of such a system. This DSS is composed by a series of tools:

- an *optimization module* for the design and sizing of the proposed system from both a strategic and a tactical point of view;
- another optimization module to perform the operational planning of the freight transport on a daily basis;
- a *simulation tool*, based on the system configuration delivered by the optimization stages, to stress it and assess its impact in the aforementioned respects.

The three stages are sequenced and all fed by real-life data collected by means of field surveys. Moreover, in order to evaluate the impact of the proposed system, a programme of enquiry activities (polls, interviews to experts of the involved domains) is also taken into consideration. Finally, MODUM aims at yielding an analysis of functional, structural and organizational constraints that prevent a similar distribution system from being deployed, and a guide of good practices.
The Project relies on a group of Academic partners with considerable expertise on transports either from a socio-economic point of view or for what concerns the design of optimization algorithms:

- the *LIPN (Laboratoire d'Informatique Paris Nord)* of the *Université Paris 13* studies the medium- and long-term problem of designing the logistic network;
- the *EMSE (École Nationale Supérieure des Mines de Saint-Etienne)* tackles the short-term optimization problem of deploying the vehicles to ship the customers in a generic workday;
- the *Laboratoire Ville, Mobilité, Transport (LTMV)* of the *École des Ponts et Chaussées* undertakes the design and implementation of the simulation tool;
- the *Laboratoire d'Economie des Transports (LET)* of the *Université Lyon 2* provides the real-data coming from recent studies on the flows of goods in French cities to represent the demand profile of consumption goods of the citizens and shops of a city and allow to depict the scenarios for the previous optimization steps. Moreover, the LET takes in charge the analysis and the socio-economic assessment of the optimization and simulation tools.

This thesis originates from the MODUM Project and the strategic and tactical study of the freight transport system it proposes.

Plan of the Thesis

The thesis is essentially structured in two parts.
Part I revolves around the *Multicommodity-Ring Location Routing Problem (MRLRP)*, i.e. the problem inspired by the strategic and tactical study of the City Logistics system proposed by MODUM. After an introductory chapter on City Logistics, the MRLRP is presented, along with three solution approaches and some computational results.
In Part II we concentrate on another problem arising in Freight Transport Logistics, namely the *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)*. The problem is formulated and then located in the literature of Combinatorial Optimization, before being tackled by a series of exact approaches of type Branch & Cut and Branch & Price.
Finally, we draw the conclusions of this work, and trace future perspectives.

Chapter 1: City Logistics

A brief introduction of City Logistics and how it is dealt with in the field of Combinatorial Optimization is proposed. This will provide an introduction to the following chapters.

Chapter 2: The Multicommodity-Ring Location Routing Problem

The MRLRP is introduced, along with a formulation in the form of a *Mixed-Integer Linear Program (MILP)*, which implicitly gives rise to an exact approach that consists in solving the MILP by Branch & Bound with a commercial solver. In order to find solutions to large-sized instances of the problem, a *matheuristic* decomposition algorithm, named *GALW*, is proposed. Moreover, a *hybrid* algorithm is defined to evaluate the quality of GALW's solutions when the size of the instances makes it impossible for the exact algorithm to even produce one. Extensive computational sessions are conducted, and the results discussed, to assess the performance of the three approaches.

Chapter 3: The Vehicle Routing Problem with Intermediate Replenishment Facilities

The Vehicle Routing Problem with Intermediate Replenishment Facilities is first described. Then, a brief review is given of the problems it descends from. Lastly, some of the application papers that tackle real-life variants of the VRPIRF are presented.

Chapter 4: Branch & Cut algorithms for the Vehicle Routing Problem with Intermediate Replenishment Facilities

Since we are aware of only few works about exact methods for the VRPIRF (as highlighted by the survey of chapter 3), we try to define some. In this chapter, we focus on compact MILP formulations for the problem and outline two Branch & Cut algorithm. Computational results on the most used benchmark instances are presented, along with a comparison with the solution obtained by some state-of-the-art heuristic algorithms that are found in the literature.

Chapter 5: Branch & Price algorithms for the Vehicle Routing Problem with Intermediate Replenishment Facilities

In the same spirit of trying to define effective exact approaches to tackle the VRPIRF, in this chapter we concentrate on extended MILP formulations aiming at the design of two Branch & Price algorithms, which are fully described. Computational sessions are conducted on benchmark instances; the results are presented and compared with those of the methods of chapter 4.
The work presented in this chapter is the result of the collaboration with Alberto Ceselli, Dipartimento di Informatica, Università degli Studi di Milano.

Appendices

Appendices are devoted to other research works derived from those presented in the above-cited chapters, to the presentation of some minor works which have been studied in parallel to the two aforementioned main research streams, or to further details on these latter.
If we suppose that the long-term decisions in the MRLRP concerning the structure of the ring (UDC, links) have been taken, we obtain a tactical subproblem, called the *Multicommodity-Ring Vehicle Routing Problem (MRVRP)*, which is introduced in Appendix A. Since the MRVRP is simpler than the MRLRP, more sophisticated exact methods can be designed to solve it to optimality. A MILP formulation for the MRVRP and an exact solution strategy, namely a Branch & Price algorithm, are proposed.
Appendix B presents a work on the *Machine Reassignment Problem* which has been the object of the *Google ROADEF/EURO challenge 2011–2012*. This work has been done together with Daniel Chemla and Bernat Gacias.
Appendix C presents a work on some problems arising in Grid and Volunteer Cloud

Computing about the optimization of Energy. This work is the result of the collaboration with Christophe Cérin and Yanik Ngoko, Laboratoire d'Informatique de Paris Nord, Université de Paris 13, and Congfeng Jiang, Hangzhou Dianzi University.
Finally, Appendix D goes into further details of the instance generation process for the computational experience of the methods for the MRLRP proposed in Chapter 2.

Part I

The Multicommodity-Ring Location Routing Problem

Chapter 1

City Logistics

1.1 Introduction

The growth and urbanization of the civilization are constantly increasing phenomena. World population passed from 5 to 6 billion persons between 1987 and 1999 and stood at 6.5 billion in 2005, with a recorded average growth rate of 1.2% per year since the late 90's [United Nations, 2005]. The trend is not expected to change, and it is projected that world population will peak at 9.1 million individuals by 2050. Figures are even more dramatic for what concerns urbanization. For instance, 2007 is estimated to have been the first time in recorded history with a larger world-wide urban population than the rural population; yet, the European population living in cities, which was by that time the 72% [Commission of the European Communities, 2009b], is expected to reach 84% in 2050 [Commission of the European Communities, 2009a].

These are among the main reasons that make urban freight transport and goods distribution a matter of increasing interest and concern in the economic, commercial, social and environmental operations of our cities. This area of transport activity is growing at a much higher rate than other ones, such as private vehicle travel or long distance freight transport [Canberra Bureau of Transport Economics, 2001]: passenger traffic is expected to stabilize in the next years, whereas there is no sign of the same trend for urban freight activity [Gargett and Cosgrove, 2004]. At the same time, patterns and intensity of freight movements themselves are having significant changes, as a consequence of technological and societal change, the environmental impact of road-based transport systems, the urban land use, the transport systems management policies, the distribution practices based on low inventories and timely deliveries, the growth of specific commerce segments, like e-business and home deliveries by mail order, that generate significant volumes of personal deliveries. More and more trucks will be on urban roads in

the next years, making the impact of urban freight traffic an even more delicate matter. Indeed, the stakes are considerable, as an improved management of the traffic related to the freight transport can have a positive impact in terms of security, congestion of the road network, noise and air pollution [Patier, 2002] and -last but not least- costs. These are all sensitive subjects.

The costs related to freight transport grow regularly. This is due to a number of factors: cities expand with a strong growth rate from both the demographic and the territorial point of views, and so do the distances travelled by means of transport [Aguilera, 2005]; the volumes of consumed goods, too, increase year by year; finally, energy and fuels are often subject to cost increases. Costs, however, are not the only economic aspect that would be impacted by a rationalization of freight transport: increased flexibility and fluidity of traffic, reduction of delays, road network reliability are all betterment for practitioners that could be achieved by suitable transport policies.

As for the environment, transportation has been the sector with the biggest growth rate of *greenhouse gas (GHG)* emissions compared to 1990 [Commission of the European Communities, 2009a]; 60% of the global oil consumption and 25% of energy consumption are due to transportation [Rodrigue, 2013], and nowadays 25% of the CO_2 emission of the whole transport sector comes from urban transport [Commission of the European Communities, 2011]. Noise pollution in cities is also important: 8.9% of cities of China had a serious level of acoustic nuisance level in 2000 [Minister of the State Environmental Protection Administration of China, 2007], while in Europe, about 65% of the population is exposed to ambient sound at levels above 55 dBA, and about 17% to levels above 65 dBA [Chepesiuk, 2005]. Some recent analysis conducted in France are sufficient to show that the improvement of freight distribution is more urgent than ever [Delaître, 2008, Gerardin et al., 2000]. Trucks represent 10% of urban transport but produce 40% of noise and air pollution, while freight transport is estimated to represent the 45% of the consumption of fossil fuel in a medium-term horizon [Commission of the European Communities, 2007]. CO_2 emissions have increased by +28% between 1990 and 2003 [Commission of the European Communities, 2003].

There are two other main aspects related to urban traffic and therefore to freight transport in cities. The first one is road network security: for instance, road transport caused 39,000 deaths in EU in 2008 [Commission of the European Communities, 2009a]. Moreover, in 2013 it has been estimated [World Health Organization, 2005] that more than 1.2 million people die on the world's road every year, and 27% of these are pedestrians or cyclists.

The second is road network congestion: it is reported [Commission of the European Communities, 2011] that more than 50% of the weight of goods in road transport are moved over distances inferior to 50 km, and more than 75% over distances below 150 km. In the region of Paris, freight transport causes between 15 and 25% of the road

surface occupation, 50% of consumed diesel, 60% of particles emission, 25% of GHG emissions, 30.5 million tonnes per year of transported goods.
One of the essential difficulties in improving the management of freight transport is represented by the conflicting objectives of the community, on one side, and urban freight operators, on the other. The former aim at collective utility objectives, e.g. controlling traffic congestion, pursue sustainable development policies and enhancing the quality of life, whereas the latter want to increase their economic benefit and the quality of the services they offer to their customers. The *City Logistics* paradigm is one possible approach to resolving this problem. City Logistics is the study of the dynamic management and operations of urban freight transport and distribution systems. The aim is to deliver win-win solution for both business and the community by ensuring optimal productivity, reliability and customer service while reducing environmental impacts, air pollution emissions, energy consumption and traffic congestion. Current research is being directed at building City Logistics systems models that aim at optimizing logistics efficiency under congested urban traffic conditions.
Combinatorial Optimization is largely concerned, as the logistic problems that can arise in the urban context are numerous, and the criteria that the different involved actors may want to optimize are several. In these last decades, this has brought to the development of a well-known Combinatorial Optimization stream, namely that of *Vehicle Routing Problems (VRP)*, from the definition of general theoretical problems to their enrichment into more and more specific case-studies inspired by real-life situations.
In the remainder of this chapter, we will first go more in depth for what concerns City Logistics (section 1.2). Then, section 1.3 will propose to the reader a few references of decision-making problems related to City Logistics. Section 1.4 will introduce the City Logistics system that represent the core of the MODUM Project, with a focus on the parts of it that make up the core of Part I of this thesis. Finally, section 1.5 concludes this chapter.

1.2 Urban Freight Transport: Types, Stakes, Trends

1.2.1 Types of Urban Freight Movements

Urban freight activities can have many variegated forms, such as waste collection, transport of building materials, retail deliveries, courier services. All of these tasks are common to urban areas, but have different purposes and characteristics, require different vehicles, concern different times of day and involve different patterns w.r.t. frequency and spatial coverage. Hence, they need to be considered separately. A classification

based on freight type and characteristics can therefore be adopted.
The study presented in [D'Este, 2000] divides urban freight movements into five main market sectors, and can be summarized by table 1.1. These market sectors capture a wide range of different cases for what concerns both vehicles and purposes:

market sector	truck type	commodity	load type	route	trip type
courier	LCV (light commercial vehicle)	mixed	PTL (partial truckload)	variable	very complex linked trips
specialist commodities (e.g. container, bulk liquid)	large (rigid/articulated)	specific	FTL (full truckload)	regular	simple trips (mostly)
general carrier	medium (rigid)	mixed	PTL or FTL	variable	variable (simple/linked trips)
over-sized, hazardous	large (articulated)	specific	FTL	fixed	simple trips
external transport	large (mostly articulated)	mixed or specific	FTL	regular	simple trips

TABLE 1.1: General characteristics of market sectors in urban freight.

The courier and general carrier sectors are those covering most urban freight distribution tasks. The same study also highlights the importance of the adoption of *light commercial vehicles (LCVs)* for urban freight distribution, as they may represent the best response to changing logistics, life styles and consumption patterns, e.g. to the increasing level of home deliveries (mail order, e-commerce). Switching from larger vehicles to the less polluting LCVs could also help reducing emissions, even though the lower emissions per vehicle could be negatively compensated by the higher overall emissions due to the increased number and usage of vehicles [Kockelman, 2000].
Another interesting classification is proposed in [Cattaruzza et al., 2013], where the name *Urban Goods Movements (UGMs)* is used to denote freight transport movements. This time, the focus is on the type of resulting delivery problems. Apart from non-commercial minor flows, the authors identify two main classes of UGMs:

- *inter-establishment movements (IEM)*, i.e. pick-up and delivery trips which are functional to the economic activities of the urban district. They can be further divided into:
  - third party transport: haulage is performed by third-party professionals. As to the load type, we distinguish *full truckload (FTL)* and *partial truckload (PTL)*. FTL strategies mainly concern hypermarket distribution, agriculture and urban industry, whereas PTL occurs with the service of retailers and tertiary activities distribution. The cases where FTL is used are in general modeled by the well-known *Transportation Problem (TP)*, introduced in [Hitchcock, 1941], while PTL schemes are in general modeled as VRPs;
  - sender's own account: transport is performed directly by producers, craftsmen or distribution companies without involving a transport carrier. Transportation is

performed with a single vehicle in the case of small companies and can therefore be modeled as a *Traveling Salesman Problem (TSP)*, whereas in the case of large companies, a fleet of vehicles is available and thus the modeling of transportation gives rise to VRPs;

- receiver's own account: haulage is accomplished directly by the receiver, as it is e.g. for small retailers going to wholesalers. They can be modeled as pick-up TSPs.

IEM cause the 40 to 45% of traffic road occupancy (expressed in kilometers per equivalent vehicle) and the 23% of park road occupancy (in hours per equivalent vehicle) in urban extended areas, which rises to an important 62% in city centres. Among the subcategories, third party transport and sender's own account cover almost the entirety of deliveries (76 and 23%, respectively), routes (58 and 40%) and total delivered weight (64 and 33%).

- *end-consumer movements (ECM)*, i.e. those that allow the direct service of end consumers. They are subdivided into two subcategories, according to the movement direction:
  - shopping trips: this concerns movement of private individuals from their houses towards shopping points. In spite of forming most of the ECMs, shopping trips are hard to optimize as they are related to end consumers behavior;
  - home and proximity deliveries: they include all movements from retailers or distribution points towards end consumers, like e.g. business-to-customer flows, parcel delivery, proximity delivery and grocery home deliveries. LTL third-party transport is often used, and VRP routes have to be optimized.

  ECM have a traffic road occupancy rate of 45 to 55%, while park road occupancy is 67% in urban extended areas, but only 31% in city centres.

1.2.2 The Involved Stakeholders and the City Logistics Approach

The following four stakeholders can be identified as the main actors of urban freight transport [Ehmke, 2012]. Each group has specific objectives and needs and tends to behave in a different manner:

- *freight carriers* deliver goods to customers. They are expected to:
  - provide high quality services (commercial and tertiary activities) to the customers;
  - reduce the economic costs related to the last mile management;
- *shippers* send or receive goods to or from other companies or persons. They aim at offering a service which is optimized in terms of costs and reliability of transport;

- *residents* are the people who live, work, and shop in the city. They benefit from efficient and reliable delivery, but suffer from nuisances resulting from urban freight movements near their residential and retail areas. Residents and shippers are in turn customers of city logistics service providers. They expect an economic and reliable delivery service. Online retail applications are an example of direct interaction of residents and city logistics providers.
- *city administrators* attempt to enhance the economic development of the city. They aim at
  - reducing congestion in the most dense urban areas,
  - decreasing both noise levels and environmental nuisances due to GHG and other atmospheric pollutants,
  - increasing safety of road traffic, and
  - revitalize the economic activity of the urban areas, particularly the town centers.

  To this end, they consider urban transportation systems as a whole to resolve conflicts between the other stakeholders.

A very delicate overall frame results from the interlaced relationships among these actors and different conflicts within the freight transport system. The generation of freight demand is from the shippers to the consumers, while city administrators set the overall framework under which delivery tasks take place, e.g. they affect planning procedures by setting complex restrictions for the realization of delivery tours, for example, certain time slots that permit or forbid the entrance of freight vehicles in pedestrian areas. Freight carriers then operate within that framework to satisfy consumer demands. It is clear how public authorities want to pursue collective utility objective which can be in conflict with the individual performance and the goals of private stakeholders. Even the slightest change in one part may strongly affect such a fragile balance. For instance, a freight carrier with poor efficiency in terms of load rate can impact on the service quality of the overall system (augmented kilometers per equivalent vehicle, with negative effects on noise, emissions and congestion) and hence increase the difficulties of management for planners and regulators. In addition, this would reduce the satisfaction level of consumers and also the reliability of firms and increase their operating costs. The authors of a 1999 study of the environmental impacts of freight transport operations in London [Browne and Allen, 1999] conclude that the economic and environmental performance of urban freight transport is strongly influenced by the following dilemma: *policies aimed at improving community outcomes from freight transport operations may often run counter to the improved operations of freight companies and their customers.* This statement also recaps the difficulties in finding adequate solutions. Given that some combination of company initiatives and government policies is almost certainly required

for the optimal development of urban freight systems, some form of reconciliation of this dilemma is essential. One possible way forward is the *City Logistics* approach.

1.2.3 Main City Logistics Trends: Mutualization and Urban Distribution Centers

City Logistics has three main objectives:

1. reduce congestion and increase mobility of freight transportation services in urban areas. More in detail, one would aim at reducing the number of freight vehicles, limit their dimensions, improve the efficiency of freight movements and reduce the number of empty trips;
2. contribute positively to the environment and to the sustainable development, mainly by fulfilling the Kyoto objectives in terms of GHG emissions, reducing pollution and noise, improving living conditions of city inhabitants;
3. preserve city center activities, mainly commercial, tourist and tertiary.

City Logistics arises from the awareness that to pursue such objectives, traffic and parking restrictions to some given main road or time slots are no longer sufficient, as they either have side effects on road congestion, or cause the displacement of shops in outer zones of the city, with negative effects on the economy of the city center.
Starting from the 70's to these days, many traffic surveys and data collection activities, conducted mostly among public authorities and freight practitioners, have highlighted that a common trend is that average load factors are typically very low, with a high number of empty trips. This suggests that a change of perspective is necessary, i.e. to consider the aforementioned stakeholders as the actors of an *integrated, mutualized logistics system*, where shippers and carriers, instead of acting and optimizing their urban freight movements separately, are coordinated by a unique operator (which in most of the cases is, or is related to, some public institution). In this model, shared stocking spaces are made available to the former by the latter, which performs the shipment operations to final customers with its own fleet in order to rationalize the usage of spaces, achieve higher load rates and therefore reduce the travelled distance, the costs and the polluction factors [Gonzalez-Feliu and Morana, 2010]. City Logistics solutions follow this model and rely on the key concepts of *consolidation* (i.e. loading goods originating from different carriers within the same vehicles) and *coordination* to achieve the desired rationalization of freight movements. The report [Capgemini, 2008], the result of a study conducted by a group of multinational companies, is a prominent example of proposal of a mutualized system based on *urban hubs*. Reductions of 40% of the transport cost per pallet, and of 25% of both the number of travelled kilometers and the emissions of

CO_2, are foreseen as effects of the adoption of the proposed system.
Other typical elements of City Logistics initiatives are the usage of low-level environmental impact vehicles that allow to accomplish environmental targets, and the adoption of *Intelligent Transport Systems (ITS)*, based e.g. on information technologies, cargo tracking systems, traffic management centers.
Most of the City Logistics initiatives to date revolve around *Urban Distribution Centers (UDCs)*. Appeared for the first time in the years 1970 in Northern Europe countries and rapidly adopted all over Western Europe, they are in most cases managed by the public institutions. A UDC is a bulk-breaking point similar to *hubs* in air transport and is the site where consolidation activities take place. Moreover, they often offer more advanced functionality towards efficiency and coordination of the freight transport activity within the urban zone, like e.g. intermodality. Long-haul trucks dock at a UDC to undergo bulk-breaking operations; goods can then be processed, sorted, consolidated into smaller vehicles that better fit last-mile shipments and then forwarded to their destinations, both coming from, or going to, the city center [Agence de l'Environnement et de la Maîtrise de l'Energie, 2004]. UDCs are a suitable solution both for the community, as they help rationalize the flows of merchandise on the territory, but also for those carriers that do not have a benefit from solving last kilometer issues with their own fleet either because they do not have sufficient volumes of goods to make it convenient, or because of strict urban regulations.

1.2.4 Single- and Two-tiered City Logistics Systems

City Logistics projects are often based on a *single-tier distribution system*, in which shipments are performed starting directly from the UDC after city vehicles have been consolidated. Single-tier system projects have been adopted in Europe and Japan since the early 90's. Most of them concerned small- or medium-sized cities and involved one or few UDCs and a restricted number of shippers and carriers. The *City Logistik* project launched in Germany and Switzerland [Dietrich, 2001] was a private initiative based on the idea of encouraging the formation of spontaneous groupings of private carriers. The project was supposed to be profitable in the short-term due to mutualization, and the public institutions acted mostly as a *facilitator*, i.e. by not imposing any particular traffic rule, hence a very marginal role. However, the City Logistik projects have yielded mild results, mostly due to the fact that extra-cost and delays caused by consolidation operations are difficult to deal with when a short-term profitability is expected. A more successful approach was adopted in the Netherlands [Visser et al., 1999], where the involvement of public authorities was stronger. Shippers were given a limited number of licenses and a series of operating rules (e.g. limits on load and number of vehicles)

to enter the urban zone; on the other hand, longer delivery periods were permitted, and the use of electric vehicles encouraged. This resulted in carriers initiating collaboration activities to consolidate shipments, with a considerable reduction of the number of trips. Other two single-tier, single-UDC distribution system examples are found in France since the early 2000's [Agence de l'Environnement et de la Maîtrise de l'Energie, 2004]. In the case of La Rochelle, big trucks are forbidden in the city center. The UDC is positioned nearby the city and exploited by a private city operator which is delegated by public authorities to perform deliveries of consolidated freight of other carriers. These latter can either be big transporters that serve the outer La Rochelle area and unload goods for the city at the UDC, or smaller operators that can enter the city center but prefer to exploit the public service to perform part of their shipments. The results of the experience in La Rochelle are controversial: the chosen electric vehicles do not fit the main typologies of merchandises to deliver, whereas more suitable thermic vehicles have had a negative environmental impact. Moreover, in spite of a strong reduction of noise pollution, urban congestion and road occupation grew considerably. The case of Monaco has similar characteristics. The city is forbidden to heavy trucks which are compelled to unload at the single UDC and let the public operator perform their last mile shipments. The restrictions are mainly motivated by town planning reasons but have indirect environmental effects. The Monaco system also has a pre-consolidation platform, located about 30km away from the UDC, that allows to intercept beforehand a part of the freight transported by big carriers: the link between the platform and the UDC is assured by the public operator with high load rates. The success of the Monaco case resides mainly in this link, and accounts for an average 38% reduction of main polluants, and a decrease of noise pollution and congestion of, respectively, 40% and 46%.

Such approaches are particularly suitable for small- to medium-sized cities, but fails to fit large ones, particularly in cases when the city center exhibits a high density of population and activities [Dablanc, 2007]. Two-tier systems have been proposed for such cities to reduce the distance from the UDC, usually located in the city outskirts, and the city center where the delivery tours begin. In a Two-tier City Logistics system, UDCs form the first level of the system and are located on the outskirts of the urban zone, while the second level of the system is constituted of *satellites*, typically existing parking spaces, where goods coming from the UDCs are consolidated into vehicles adapted for utilization in dense city zones. Two types of vehicles are involved in a two-tier City Logistics system and both are supposed to be low-level environmental impact ones. *Urban-trucks* are used to move freight from UDC to satellites. They normally have a relatively small capacity. During a route, they may visit more than one satellite. *City-freighters* are vehicles of even smaller capacity that can travel in narrow city street in order to perform the required distribution activities. Of course, the two-level structure

imposes more intense coordination (and often synchronization) efforts of the involved facilities and vehicles without compromising timely delivery of loads to customers and economically and environmentally efficient operations. After consolidation operations at UDCs, each urban-truck receives a departure time and route and travels to one or several satellites, and the same steps apply at satellites to city-freighters w.r.t. final customers. Each of these latter performs a route to serve the designated customers, and then travels to a satellite (or a depot) for its next cycle of operations. This also contributes to diminish the number of empty vehicle-km. We recall here the two-tier system cases of Amsterdam (project CityCargo [Metrolinx, 2012]) and Rome [Crainic et al., 2004].

1.3 Some References in the Literature of Decisional Problems in City Logistics

The planning of a City Logistics system implies decision-making activities from a strategic, a tactical, and an operational point of view.

- Decisions within the *strategic* level concern a large part of the organization. They have a major financial impact and typically include the design of the transportation system, the size and mix of freight vehicles, the type and mix of transportation services. Consequent decision problems are weakly structured, but complex and of high risk and uncertainty, and strongly affect decisional problems at subsequent levels;
- *tactical* decision-making copes with short or medium-term activities, for which the planning is done once and is kept for a certain period. For example, postmen perform each day the same trips, regardless of the quantities of mail to deliver, and therefore the planning of a workday can be done once and be run for several days, i.e. for the whole week long. Therefore, tactical decisions in City Logistics concern the efficient and effective use of transportation infrastructure and the alignment of operations to fulfill strategic guidelines. At this level, logistics service providers deal with the acquisition and replacement of their equipment, medium-term driver-to-vehicle assignments, and cost and performance analysis. Decisions made at the tactical level constrain the activities of operational management level, and the quality of the achieved solutions play a major role;
- decisions of *operational* management concern short term, day-to-day operations. Operational planning is characterized by a short planning horizon and decision problems of detailed problem structure. Time dependence, synchronization, time windows, and in general detailed real-life constraints, which are usually neglected at strategic and

tactical level, are fully taken into account. Here, logistics service providers plan current and next day activities and need to be able to anticipate future developments such as congestion or expected transportation requests.

In the last four decades, Combinatorial Optimization has been hugely concerned by the theoretical study of problems related to the three above-cited aspects of City Logistics, and the development of algorithmic methods to tackle them. We now give a very short insight to a small number of works, as we will delve into many families of such problems in the following chapters of the thesis.

After identifying the main issues in the planning of freight transport systems, the authors of [Crainic and Laporte, 1997] propose and survey a set of Operations Research models accordingly. Since strategic decisions commonly revolve around the design of the system, two families of problems are proposed as general tools: *location models* and *network design models*, thus addressing the decision issues concerned with both the location of facilities and the creation of links between them, before enlarging the scope of their analysis to the strategic planning on wider geographical areas. When dealing with tactical planning aspects, a distinction is made between long-haul multimodal transport and city freight transport to retailers and final customers. These two types give rise respectively to *service network design problems* and *vehicle routing problems*. For the former case, a general network optimization model is yielded, which may represent a large variety of real situations, provided that it undergoes a further definition depending on the specific case, as the authors have done in other works concerning the French and Canadian railway system. For the latter case, the general *Capacitated Vehicle Routing Problem (CVRP)* is outlined, before yielding a two-index formulation and some general guidelines on savings and tabu search heuristic algorithms. Finally, typical issues of operational level are dealt with, most notably service scheduling, empty vehicle distribution or repositioning, crew scheduling, before treating uncertainty and the *Stochastic VRP*. For all of these subcases, a description is given, along with a review of existing methods.

In [Taniguchi et al., 2003], a mathematical programming and simulation framework is defined for the evaluation of a set of City Logistics measures: advanced information systems, co-operative freight transport systems and load factor control. The authors suppose to deal with an urban area where some freight carriers have introduced *Advanced Vehicle Routing and Scheduling (AVRS)* procedures and established a co-operative freight transport system. Moreover, the governing municipality perform controls over the load factors of vehicles. Based on trends in Japan at the beginning of the 2000's, the authors incorporated designated pick-up/delivery times in the system requirements. Therefore, they modeled the problem as a *Vehicle Routing Problem with Time Windows and Pickup and Delivery (VRPTWPD)*, for which a mathematical

model is given and discussed. The AVRS couples a *Genetic Algorithm (GA)* to solve the VRPTWPD with a Dynamic Simulation Model based on the macroscopic dynamic simulation BOX model introduced by [Fujii et al., 1994], which estimates the travel time of each arc of the city network in which pick-ups and deliveries take place. In its basic configuration, the AVRS considers a competitive context. The GA block yields VRPTWPD solutions to serve the pick-up/delivery tasks of each company and feeds the BOX-based simulator, which updates the travel times every 30 minutes. The VRPTWPD is then solved again with the updated travel times. VRPTWPD can be considered to incorporate time dependent travel times. The system runs until a predefined stopping criterion is satisfied. The AVRS system is run in three configurations on the real road network of the city of Kobe, Japan, and fed with freight demand data resulting form some survey of good movements conducted in 1995. The three configurations are a) the basic competitive configuration described before, b) the same, improved with the introduction of co-operative freight transport policies, and c) still the basic configuration, enhanced with load factor controls. The tests results are compared with the real-data of Kobe in terms of total costs, total operation times and CO_2 emissions, proving that each of the considered City Logistics measures can have a favorable impact on each of these three indicators.

Finally, [Crainic et al., 2009] is an example of work that aims at defining a new rich type of VRP that incorporates a series of constraints inspired from real City Logistics situations. The proposed problem, the *Two-Echelon, Synchronized, Scheduled, Multi-Depot, Multiple-Tour, Heterogeneous Vehicle Routing Problem with Time Windows (2SS-MDMT-VRPTW)*, revolves around a two-tiered distribution system. Goods are located at *external zones* (i.e. UDCs or other similar logistic platforms) and available to serve customer demands. Each of the latter is defined by a customer, a quantity, a required unload time and a service time window. Freight is transported by urban-trucks to satellites, where it is unloaded and consolidated on city-freighters. The operational model considers cross-dock transshipment and synchronization at satellites between vehicles of the two types at specific meeting times. This, on one hand, allows for the satellites to be existing urban spaces, e.g. parkings, but on the other hand it prevents city-freighters to park and wait at satellites. In the latter case, city-freighters return to so-called *depots* until a new duty is requested. Each urban-truck starts at an external zone, visits one or more satellites to unload, then either exits the system or reaches another external zone to wait for the next duty; a city-freighter starts from a depot, visits the first satellite, performs a service trip to serve a set of customers and ends up to a (possibly different) satellite, then either leaves for a new trip or goes to a depot if no new duty is scheduled. The problem consists of determining the routing of each demand from the external zone to the final customer, the dispatching of the vehicles of both urban-truck and city-freighter fleets, the personnel deployment and the freight-delivery

schedules, while respecting customers time windows, synchronization requirements, in such a way to maximize economic and environmental efficiency and minimize the impact on road congestion. The problem therefore exhibits both tactical decision-making features and operational elements, in that a short-term planning is coped with, most of the demands do not have a regular basis, and real-time control and adjustment of operations are required. Moreover, the planning covers a set of T short time periods, and travel times are considered to depend on the considered time period and are not necessarily symmetric. A general modeling framework is yielded, along with a set of variants derived from assumptions on the management of the urban-truck fleet. It is noteworthy the modeling of a single-tier system, which can be seen as a particular case of the general framework. A hierarchical decomposition approach is defined for the 2SS-MDMT-VRPTW, that first determines the schedules for urban-trucks and first-level demand distribution strategy, then finds the planning of city-freighters that fits first-stage output, serves customers within their time windows and repositions city-freighters to satellites or depots. A model for the first subproblem is derived from the general case. As to the second stage, it gives rise to a new City Logistics problem, the *Synchronized, Scheduled, Multi-Depot, Multiple-Tour, Heterogeneous Vehicle Routing Problem with Time Windows (SS-MDMT-VRPTW)*, for which a new formulation and a new decomposition approach are proposed.

1.4 The MODUM Project and the proposed City Logistics System

The *ANR* (*Agence Nationale de la Recherche*, the French National Agency for Research) Project *MODUM (Mutualisation et Optimisation de la distribution Urbaine de Marchandises)* [Agence Nationale de la Recherche, 2010] has its main purposes in the proposal and study of a freight distribution system for urban areas based on a *ring of UDCs* situated in the outskirts of a city. The proposed logistic architecture is composed by three main elements:

- the Urban Distribution Centers to be located in the outskirts of the city;
- the links to connect the UDCs in a ring, to allow massive flows of goods among them via a dedicated fleet of fast, non-polluting means of transport (trains, shuttles…);
- a fleet of electric vans, or more generally small, low-level environmental impact vehicles, to perform the shipment (delivery or pick-up) trips from the UDC to the final customers or vice versa.

Both the ring and the electric vans fleets are managed by a single operator, either public or delegated by public authorities. The originality of a ring-of-UDC-based distribution system resides in the possibility that the ring offers to make massive flows of goods circulate around the city before being shipped starting from the most suitable UDC. Long-haul trucks of private carriers come from the main motorways outside the city, i.e. freight is considered to originate from one of a finite set of source points called *gates*. Trucks then address to the nearest UDC and leave their freight there, where bulk-break and consolidation on ring vehicles will take place. Goods are then sent to the different destination UDCs by the ring fleet, which assure regular and fast deliveries, a high load rate due to mutualization, and low emissions. In the destination UDC, a second bulk-breaking and consolidation step takes place to prepare goods for the shipment. High load rate and reduced pollution are the drivers of the service to final customers, which is then performed with green vans according to the same mutualization principle. In order to avoid empty trips and meet the environmental purposes of the system, service trips can be opened, i.e. they are not compelled to go back to the UDC they started from. Unlike many real-life City Logistics system proposed in the last decades [Crainic et al., 2009], the proposed system addresses the reverse UGM from origins (i.e. pick-up customers) within the city to destinations outside.

A *self-service hiring system* is also considered to meet the needs of small operators who want to perform their deliveries autonomously. Therefore, such self-service system addresses minor urban flows of goods. This additional service shares the same fleet of electric vans that is used for shipment operations to final customers: vans are made available to private transporters not only at UDCs, but also in *Self-Service Parking Lots (SPLs)* located inside the city. Trips performed by private operators do not have to go back to the starting SPL, either.

A rebalancing issue arises from the possibility of having open trips for both the service trips and self-service trips. The distribution system presented by the MODUM project is supposed to take charge of the repositioning activities which are necessary to the daily availability of the fleet of green vans. The proposed system can be seen as a *two-echelon, single-tier* distribution system, as goods go from gates to customers via an intermediate echelon, i.e. the UDCs, but no satellite layer inside the city is considered.

1.4.1 Decision-making issues involved by the proposed system

From a decisional point of view, the freight distribution system proposed by MODUM calls for the approach to different problems on three different levels, as it is the case for many City Logistics projects: strategic, tactical and operational. The scenario of the resulting problems is the urban area of a medium-sized city, of which we suppose to have

a detailed description in terms of flows of merchandise.
On a strategic level, the decisions concern the position and the dimensioning of the UDC and the ring links, given a demand distribution, i.e. a set of demands localized within the city. The resulting optimization problem must determine the system configuration that best fits such demand distribution.
On a tactical level, the flows to, from and among the chosen UDCs must be routed, taking into account both their capacity and that of the ring links. The latter will be expressed in terms of ring vehicles capacity per service frequency. Maximum load and autonomy of the electric vans must also be considered, as well as the rebalancing contraints - at least in a simplified form. Note that both the first two levels, i.e. the strategic and the tactical problems, disregard the time dependency: the static case, with fixed demands, is considered. Moreover, neither of the two takes into account the usage of the self-service hiring system by private operators, which is estimated to represent a small percentage of the overall freight movements and is hence neglected at this level of detail.
Finally, at the operational level, the problem is the optimal deployment of the fleet of electric vans to ship the final customers, considering nonaggregated demands, time dependency, full operational constraints (including the rebalancing policy) and the self-service hiring system.

1.4.2 A Focus on Strategic-Tactical Planning

The work presented in this thesis originates from the study of the problem arising when considering the strategic and the tactical decision layers concerned by the requirements of the MODUM project. This problem has been given the name of *Multicommodity-Ring Location Routing Problem (MRLRP)*, as it can be considered to belong to the family of *Location-Routing Problems (LRPs)*, one of the most known strategic problems in Combinatorial Optimization. In addition to the strategic elements that are common to most of the LRPs, the MRLRP features the requirement to connect the opened depots via a ring, a network design aspect that considerably impacts the decision-making. The *multicommodity* attribute refers to the fact that goods to deliver are supposed to originate from different sources and are therefore considered to represent as many commodities. In the MRLRP, some additional assumptions have been made w.r.t. the strategic and tactical subproblems of MODUM: here we recap the main ones. For what concerns the strategic side, the locations where UDCs can possibly be built are in finite number, and each has associated fixed capacity and cost. This holds also for each potential ring link. A bound is imposed on the maximum number of UDC that can be opened. As to the tactical aspects, the size of fleet associated with each UDC is

not given. Accordingly, the fleet repositioning policy is introduced in an implicit (and somehow milder w.r.t. the operational problem) way, i.e. by characterizing each UDC and SPL with fixed *rebalancing bounds*: the difference between the number of outgoing and incoming open trips must be comprised between them.

1.5 Conclusions

In this chapter, the main concepts of City Logistics have been briefly reviewed. Stakes, types of freight, and finally the main trends in building urban freight distribution systems, have been presented, with a focus on Urban Distribution Centers, the facility that represents the core of the most promising City Logistics initiatives to date. Single- and Two-tier UDC-based systems have been treated, along with some examples of project that have been proposed in the last two decades. Moreover, a short examination of the main decision-making issues has been presented, as well as some reference papers among those that apply Combinatorial Optimization concepts to City Logistics. This has offered us the possibility to exhaustively introduce the proposal of the ANR Project MODUM, and in particular the MRLRP, a Location-Routing Problem inspired from it. The following Chapter 2 presents the MRLRP in full detail, from the problem definition to some proposed solution approaches.

Chapter 2

The Multicommodity-Ring Location Routing Problem

2.1 Introduction

The last mile problem in freight transportation and distribution is a complex issue. Each day, large amounts of merchandise are transported to and from the outside of the city via long-haul vehicles, capable of carrying large quantities of freight, but often inappropriate or forbidden for retail pick-up and delivery. In that case, transshipment is necessary to reach the final customers. We address in this chapter a new city logistics problem called the Multicommodity-Ring Location Routing Problem (MRLRP), based on publicly held depots called *Urban Distribution Centers* (UDCs) that collect inbound and outbound freight. The underlying logistics system combines cross-docking operations on the *first level*, i.e. the long-haul transport side, and retail trips on the *second level*, i.e. the city side. Each potential site on which to install a UDC has an associated installation cost and capacity – which must be thought of as a short-term storage capacity,as it is the case with cross-docking stations. Once chosen, the installed UDCs will act as starting and ending points of service paths to final customers. *Gates* on the outskirts of the city represent the sources of goods that must be delivered to city customers, and the destinations of goods that must be collected from them. Therefore, they can be thought of as located on the terminal points of the main roads via which freight can come or go. Potential sites can be reached from gates via the existing road network, thus no connection needs to be constructed between them. However, an upper bound is given on the number of UDCs that a gate can make use of. In the city, there can be both pick-up and delivery *demands*, each one being characterized by the quantity to pick up or to deliver and the concerned gate, since goods must come from, or go to, a specific

one. However, the pick-up and delivery duties must be routed separately. Furthermore, a *ring* must be designed to connect the selected UDCs, so that the goods to deliver, after being transported from their gate to a first UDC, can either leave it directly for delivery or transit to a second UDC in the ring before being shipped (the former case is called *direct delivery* or *direct pick-up*, or more generally *direct shipment*). In order to minimize the empty trips and thus fulfill environmental purposes, retail trips are allowed to be open, i.e. they are not compelled to end at the same UDC from which they started. For the same reasons, the system makes use of low-level environmental impact vehicles (i.e. electric vans), which therefore have a maximum trip length constraint, along with a canonical load limit. In addition, a self-service van hiring system is available and its stations, a set of *Self-service Parking Lots* (SPLs), can serve both as additional ending points of open delivery paths and as additional starting points of open pick-up paths. UDCs and SPLs consequently share the same fleet of vehicles. Finally, to simplify the fleet repositioning at the end of the workday, *fleet rebalancing constraints* are imposed: the balance of open paths concerning an SPL or a UDC (i.e. those ending at it minus those leaving it) is requested to respect given upper and lower bounds. The aim is then to determine:

1. a subset of UDCs to open and a ring (Hamiltonian circuit) to connect them;
2. the flows between gates and UDCs and the flows in the constructed ring;
3. the assignment of demands to UDCs, via delivery and pick-up service paths.

The objective function consists of minimizing the sum of the installation costs for both selected UDCs and selected arcs on the ring, flow transportation costs and routing costs (in the following we will distinguish between the expressions *routing cost*, which we will use to refer to the fixed cost involved in moving from one point to another at second level, and *flow transportation cost*, which will be used to refer to the per-unit cost of carrying one unit of goods from one point to another at first level).
The MRLRP is NP-hard since it generalizes the classical CVRP (Capacitated Vehicle Routing Problem), which we obtain in the special case with a single UDC of unlimited capacity, no SPL, no pick-up demands, and in which the distance constraint on service paths is relaxed. It belongs to the well-known family of *Location-Routing Problems* (LRP), since it is guided by strategic location and network-design decisions (where to install facilities and how to connect them), but without disregarding more operational vehicle routing and fleet management aspects; the latter prevent suboptimization that would result from the separation of the two aspects and avoid long-term decisions that would penalize subsequent services on a daily basis. The nature of the problem impacts on some of the requirements, which are weaker than they would be in a more operational context. We consider a time-independent scenario: the time horizon is assumed to be a

generic workday with neither decomposition of the day into time steps nor time-specific constraints (e.g. time windows for service duties). Moreover, goods are characterized only in terms of quantities; the demand units may be tons, m^3, containers or pallets. Finally, notice that even the fleet rebalancing constraints are more tactical than operational. Figure 2.1 displays the elements of an MRLRP instance along with a feasible solution: gates, potential sites for UDCs, SPLs and customers, flows between gates and UDCs, arcs connecting the opened UDCs and routes picking or delivering the goods to customers.

In this chapter, we propose a new problem and a Mixed-Integer Linear Programming formulation that makes use of route variables for the second-level routing part, edge variables for the first-level flows, and binary variables for the location and network design decisions. To solve this problem, an exact approach is proposed that is able to solve instances with up to 40 pick-up/delivery demands, and some simple larger instances, i.e. with up to 100 demands. To solve more complex larger instances, we propose *GALW*, a matheuristic based on the decomposition of the original problem into four subproblems, three of which are solved to optimality. Furthermore, a hybrid method is proposed and tested, both to solve medium-sized instances and to prove the quality of the matheuristic for the instances in which the exact algorithm fails. The last contribution, which in our opinion is worth mentioning, is the generation of a benchmark set of MRLRP instances, the parameters of which vary according to different criteria, thus offering a considerable testbed.

The rest of this chapter is structured as follows. Section 2.2 gives an insight into the existing literature on problems that are close to the MRLRP. Section 2.3 describes the data and graphs that define an instance of the MRLRP. A MILP formulation is given in section 2.4, while section 2.5 introduces different classes of valid inequalities for improving the formulation. Section 2.6 describes the four-stage heuristic algorithm GALW. In section 2.7, a set of instances is introduced, starting from LRP instances taken from the recent literature and integrated with MRLRP specific features. Numerical results are displayed and analysed for the three methods for small-sized instances, and for the matheuristic and the hybrid approach for larger instances. Finally, section 2.8 concludes the chapter.

2.2 Positioning in the Literature

As we mentioned previously, the MRLRP is a deterministic, static problem that belongs to the class of *Location-Routing Problems (LRPs)*. Reference surveys on Location-Routing problems can be found in [Laporte, 1988], [Laporte, 1989], [Min et al., 1998],

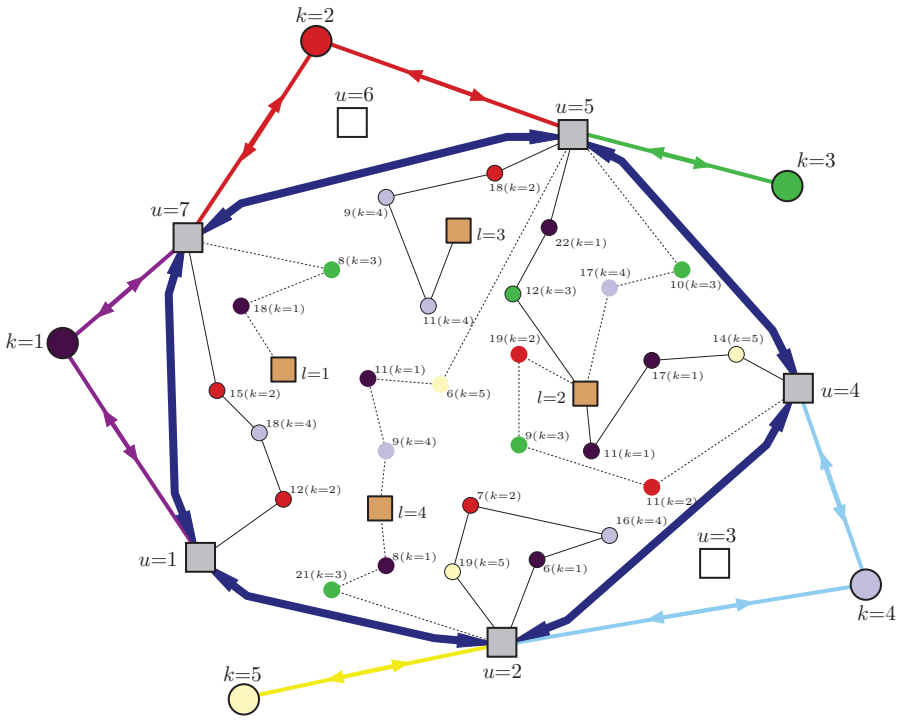


FIGURE 2.1: An example of an MRLRP instance and a feasible solution. There are 5 gates, each associated with a commodity k (represented by a color); we have 7 candidate sites u to install UDCs; 4 SPLs l; 15 delivery demands, shown as circled spots; and 12 pick-up demands. In the solution, the service paths used to satisfy the demands are represented by a continuous stroke for delivery and a dashed stroke for pick-up. Note that there are UDC–UDC closed paths, UDC–UDC open paths and finally UDC–SPL paths: none of them exceeds the given length and load limits. All of the UDCs and SPLs have a fleet balance between -1 and +1, with the exception of SPL l=4 with -2. Five UDCs are chosen and connected in the ring; 3 gates make use of as many UDCs as the maximum allowed number, which in this case is 2.

[Nagy and Salhi, 2007] and, more recently, [Prodhon and Prins, 2014].

The most investigated problem of this wide class is the *Capacitated Location-Routing Problem (CLRP)*. The CLRP is modelled with an undirected graph, in which the nodes are a set of clients, each one with a demand, and a set of potential sites on which to install facilities, each one with a depot cost and a capacity. Clients must be served by an unlimited fleet of homogeneous vehicles of fixed capacity. A service route is asked to return to the same depot from which it started, and its cost is given by the sum of the costs of the visited edges. The objective is to minimize the sum of depot, vehicle and routing costs.

One can find LRP variants with, for instance, bounds on the number of open depots [Laporte and Nobert, 1981, Laporte et al., 1986], uncapacitated depots [Burke and Tuzun, 1999], mobile depots [Amaya et al., 2007, Del Pia and Filippi, 2006]; with a heterogeneous fleet [Ambrosino and Scutellà, 2001]; with bounds on the route length [Berger et al., 2007, Laporte and Nobert, 1981] or constraints on both route length and customer deadline [Aksen and Altinkemer, 2008]; variants arising from the study of emergency situations

[Rath and Gutjahr, 2014]; variants with possibility of round-trip and multi-trip routes [Lin et al., 2002], open routes [Berger, 1997] or delivery and pick-up routes [Karaoglan et al., 2012, Nagy and Salhi, 1998]; more complex variants with multi-period planning [Prodhon and Prins, 2008] or inventory management [Guerrero et al., 2013, Liu and Lee, 2003].

Let us now introduce some extensions of the LRP that display some features in common with MRLRP. In [Min, 1996], the author proposed the problem of installing consolidation terminals and both suppliers and customers must be allocated. Similarly, in [Perl and Daskin, 1985], the authors introduced a warehouse level to define the *Warehouse Location-Routing Problem (WLRP)*. In both [Min, 1996] and [Perl and Daskin, 1985], the allocation of customers' demands from the originating source to an intermediate consolidation depot makes the interaction between strategic and operational levels even tighter.

In [Singh, 1998], a problem called the *LRP-HTSP* was introduced: we only have depots and clients as in the classical CLRP, but location decisions are also affected by the construction of a ring to connect the chosen depots. Both the warehouse (or supplier) level and the connection of chosen depots via a ring are integrated into the MRLRP, which furthermore introduces flows in the selected ring in order to represent completely the routing of goods from suppliers to final customers. This is made possible by the integration of the two aspects.

The closest problem to the MRLRP may be the *Many-to-Many LRP (MMLRP)*, introduced in [Nagy and Salhi, 1998]. A list of n customers, the nodes of a complete undirected graph, is given. Moreover, a list of *pairs* of customers (i, j), $i, j \in 1 \dots n$, is known, each one associated with a quantity q_{ij} of a specific commodity that i wishes to send to j. There are no special candidate sites for terminals; instead, each customer can become a *terminal* (or *hub*) with a fixed cost. Once chosen, terminals can be connected by means of direct routes in order to exchange goods, while service tours to other clients result in mixed delivery/pick-up trips. We have two classes of vehicles, *inter-hub* and service vehicles, both of which are homogeneous and have fixed usage costs and variable transportation costs; the former are cheaper in terms of variable cost and the latter have a capacity and a maximum trip length. Solving the MMLRP consists of choosing a set of service routes and terminals so as to minimize the sum of the terminal fixed costs and transportation costs. The authors outlined a heuristic two-level nested method, in which both the terminal location and the service-level routing are approached heuristically, with the former playing the master role and the latter the subproblem role, in a hierarchical feedback framework. The MMLRP generalizes other problems than the *VRP with pick-up and delivery (VRP-PD)* and LRP, like the *Hub Location Problem* and the *Many-to-Many Transportation Problem*. Despite the strong similarities between the MRLRP and the MMLRP, there is a strong difference concerning the network design

aspect: in the MRLRP we must trace a Hamiltonian circuit between the selected facilities, which has to be minimal w.r.t. the arc installation costs, while in the MMLRP the chosen terminals are explicitly completely connected among them and no fixed cost is present at the hub level. Moreover, there are substantial differences regarding the service routes' structure.

To conclude our brief analysis of the literature, note that the *Two-Echelon LRP (2E-LRP)* also presents similarities to the MRLRP. It aims to create a three-level distribution system in which the number and location of two levels of facilities have to be determined. The name comes from the *Two-Echelon VRP (2E-VRP)*, which is a two-level extension of the common VRP in which deliveries are made from a central depot to customers via intermediate depots called satellites and a set of both first- and second-level tours. The 2E-VRP will be briefly reviewed in section A.2. [Boccia et al., 2011] offered a complete statement of 2E-LRP, along with a three-index, a two-index and a Set-Partitioning-based MILP formulations, even though the first article concerning the 2E-LRP ever to appear was, to the best of our knowledge, the one by [Boccia et al., 2010], which proposes a metaheuristic approach. GRASP heuristics [Nguyen et al., 2012a] and multi-start heuristics [Nguyen et al., 2012b] with path relinking, as well as genetic algorithms [Lin and Lei, 2009] and Variable Neighborhood Search (VNS) algorithms [Schwengerer et al., 2012], have been applied to the 2E-LRP, while [Contardo et al., 2012] presented valid lower and upper bounds.

As for solving methods for Location-Routing Problems, let us cite first some previous papers before more recent ones. In [Berger, 1997], Berger proposed a Branch&Price algorithm for the *Distance-Constrained LRP*, using a Set-Partitioning-like formulation in which binary variables are associated with every possible service route. [Berger et al., 2007] developed Berger's work to obtain an improved Branch&Price algorithm (see [Nemhauser and Savelsbergh, 1996]) capable of solving to optimality instances with up to 100 customer nodes and 10 candidate facilities. Both [Berger, 1997] and later [Berger et al., 2007] introduced valid inequalities to reduce the number of constraints dramatically and further strengthen the Linear Programming relaxation and the lower bound. Improved column generation techniques based on Dantzig-Wolfe decomposition were proposed for the LRP in [Akça et al., 2009]. An efficient Branch&Cut algorithm was also proposed in [Belenguer et al., 2011]. [Baldacci et al., 2011b] provided a new exact method with tight bounds for both the capacitated and the uncapacitated versions of the LRP, again starting from a Set-Partitioning-based formulation of the problem. A set of bounding procedures, based on dynamic programming and dual ascent methods, was presented, in order to decompose the LRP into a small set of Multi-Capacitated Depot VRPs (MCDVRPs). In a more recent work, [Contardo et al., 2014a] proposes a Branch &Cut&Price algorithm. Both [Baldacci et al., 2011b] and [Contardo et al., 2014a] can solve some instances with up to 200 customers and 10 to 14 depots.

Many heuristic and metaheuristic methods have been developed for the CLRP and several works can be found in the literature: clustering-based algorithms [Barreto et al., 2007], heuristics based on solution construction and local search [Prins et al., 2006], Tabu Search methods [Burke and Tuzun, 1999], Iterative Local Search techniques [Derbel et al., 2010], VNS algorithms [Jarboui et al., 2013], and finally matheuristic approaches, like [Prins et al., 2007], [Escobar et al., 2013] and more recently [Contardo et al., 2014b].

2.3 Graph Representation, Data and Notation

The MRLRP problem is defined on a complete mixed graph $G = (V, E \cup A)$, where $V = K \cup U \cup L \cup P \cup D$. K is the set of gates, U is the set of locations in which a UDC can be constructed and L is the set of SPLs. P and D are the sets of pick-up and delivery demands, respectively. Each pick-up demand $i \in P$ is defined by a quantity q_i to pick and a gate $k_i \in K$ to which this quantity goes. For a delivery demand $i \in D$, the same notation holds with quantity q_i delivered from gate k_i. Each UDC $u \in U$ has a given installation fixed cost F_u and capacity Q_u, which is occupied only by goods that undergo cross-docking operations on u, as we will explain later. We will assume, without loss of generality, that $|U| > 3$ and that the UDC capacities Q_u and the overall demand $\sum_{i \in P \cup D} q_i$ guarantee that at least three UDCs must be opened. On the other side, the number of UDCs to be built in the ring is bounded by a given number $N \geq 3$, for budget reasons. For conciseness purposes, we define the collection of UDC subsets $\mathcal{S}_U = \{S \subset U : 3 \leq |S| \leq |U| - 3\}$. In the following, we identify each gate $k \in K$ with a different commodity. We note $P_k = \{i \in P : k_i = k\}$ the set of pick-up demands that go to gate $k \in K$, and $D_k = \{i \in D : k_i = k\}$ the set of delivery demands that come from gate k. Therefore, sets P_k form a partition of P, and the same applies to sets D_k w.r.t. D.

The arc set A represents the first level and is defined as $A = (K \times U) \cup (U \times K) \cup A_U$, where $A_U = \{(u, v) : u, v \in U, u \neq v\}$ is the arc set defined on U. Each arc $(u, v) \in A_U$ has an associated per-flow-unit transportation cost c_{uv} and capacity q_{uv}. Per-flow-unit transportation costs c_{ku} and c_{uk} are also associated with arcs $(k, u) \in K \times U$ and $(u, k) \in U \times K$, respectively. A gate can directly exchange goods with a maximum number B of UDCs. The edge set E, the undirected part of graph G, represents the second level; it is defined as $E = E^p \cup E^d$, where $E^p = (U \cup L \cup P) \times P$ is the set of pick-up edges and $E^d = (U \cup L \cup D) \times D$ is the set of delivery edges. The cost data associated with edges $(i, j) \in E$ are routing costs c_{ij}, proportional to Euclidean distances. The part of G that represents the first level is directed (arc set A) because of the potentially asymmetric cost and capacity data and the possibility to model flows between two connected UDCs in both directions. Nevertheless, given any two $u, v \in U$,

they are supposed to be connected either in both directions or not at all; that is why we define arc subset $A'_U = \{(u,v) : u,v \in U, u < v\}$ and associate an installation cost g_{uv} with each arc $(u,v) \in A'_U$, i.e. the cost to connect u and v in both directions. In a similar way, the B UDCs that a gate can be linked to must be the same for both delivery and pick-up.

Each UDC or SPL $h \in U \cup L$ has given bounds $-\delta_h^-$ and δ_h^+ to impose fleet rebalancing constraints. The number of second-level vehicles at each UDC $u \in U$ and each SPL $l \in L$ at the end of the day should be the same as at the beginning, with an allowed deviation comprised in $[-\delta_h^-, \delta_h^+]$. An unlimited fleet of homogeneous electrical vehicles is available in each $u \in U$ and each vehicle has a limited capacity q and length M.

In the following, the name *route*, with index r, refers to a second-level route that starts at a UDC (or an SPL for pick-up), visits a set of demand nodes sequentially and ends at a UDC (or at an SPL for delivery). When the starting and ending points are the same, it is a classical back-to-depot route, otherwise we call it an *inter-depot route*. We denote by $R_r = (i_{r_1}, \ldots, i_{r_{|R_r|}})$ the sequence of demands served by route r, and by E_r the subset of edges of E^p or E^d (depending on whether $R_r \subset P$ or $R_r \subset D$) that compose route r. For each route r we can compute its load $q(r) = \sum_{i \in R_r} q_i$ and routing cost $c(r) = \sum_{(i,j) \in E_r} c_{ij}$; the former term can be further classified according to the commodity by defining $q_k(r) = \sum_{\substack{i \in R_r : \\ k_i = k}} q_i$. A *feasible route* is a route r such that $q(r) \leq q$ and $c(r) \leq M$. The set of all feasible routes is denoted by $\mathscr{R}$, partitioned into $\mathscr{R}^p$ and $\mathscr{R}^d$ according to pick-up and delivery. $\mathscr{R}_i$ denotes the set of all routes that serve demand $i \in P \cup D$. We then define $\mathscr{R}_h^+$ and $\mathscr{R}_h^-$, $h \in U \cup L$, as the sets of routes that start and end at h, respectively; and finally $\mathscr{R}_u^{+d} = \mathscr{R}_u^+ \cap \mathscr{R}^d$ and $\mathscr{R}_u^{-p} = \mathscr{R}_u^- \cap \mathscr{R}^p$, the sets of delivery and pick-up routes that start and end at $u \in U$, respectively.

2.4 A Mixed-Integer Linear Programming (MILP) Formulation for the MRLRP

We propose to model the MRLRP with a Set-Partitioning-like formulation, which uses variables associated with service paths, instead of two/three-index variables linked to arcs and vehicles. The equivalence between models of the two families has been proved for various problems of the VRP class, and notably for the CLRP in [Akça et al., 2009]; moreover, Set-Partitioning-like formulations are in general more efficient, since they provide stronger LP relaxations and lower bounds. In the case of the MRLRP, the use of path variables rather than arc variables on the second level allows us to express the relation between the first-level network and the second-level network better. Furthermore, the previous works by [Berger, 1997] and [Berger et al., 2007], which

we take advantage of, used such a formulation. This modelling choice would not be affected if we removed the assumption that second-level routing costs are proportional to Euclidean distances (i.e. symmetric and subject to triangular inequality), as the impact of such a removal would be underlying in the definition of the collections of routes.

2.4.1 Decision variables

We define four families of decision variables:

1. binary *ring variables*:
   - ▶ $y_u = 1$ if a UDC is located at site $u \in U$, 0 otherwise (location variables)
   - ▶ $z_{uv} = 1$ if sites u and v are directly connected (in both directions) in the ring, 0 otherwise, for $(u, v) \in A'_U$ (connection variables)
2. binary *service variables* $\chi_{ku} = 1$ if gate k directly exchanges goods with UDC u, 0 otherwise
3. binary second-level *routing variables* $x_r = 1$ if route $r \in \mathscr{R}$ is selected, 0 otherwise
4. first-level *flow variables*, which are all non-negative and continuous:
   - ▶ φ_{ku} = flow of goods transported from gate k to site u
   - ▶ φ_{uk} = flow of goods collected from site u to gate k
   - ▶ φ^{dk}_{uv} = flow of delivery goods of commodity k through arc $(u, v) \in A_U$ (k-outflows)
   - ▶ φ^{pk}_{uv} = flow of pick-up goods of commodity k through arc $(u, v) \in A_U$ (k-inflows)
   - ▶ ϕ_{ku} = maximum value between the quantity that comes directly from gate k to site u and the quantity coming from gate k that leaves u for a delivery route
   - ▶ ϕ_{uk} = maximum value between the quantity that goes directly from site u to gate k and the quantity going to gate k that arrives at u with a pick-up route

2.4.2 MILP formulation

The MILP model associated with the MRLRP is the following:

$$(\mathcal{M}^{MRLRP})$$

$$\min \overbrace{\sum_{u \in U} F_u y_u}^{(A)} + \overbrace{\sum_{(u,v) \in A_U} g_{uv} z_{uv}}^{(B)} + \overbrace{\sum_{r \in \mathscr{R}} c(r) x_r}^{(C)} +$$

$$\overbrace{\sum_{\substack{k\in K\\ u\in U}}(c_{ku}\varphi_{ku}+c_{uk}\varphi_{uk})}^{(D)}+\overbrace{\sum_{\substack{(u,v)\in A_U\\ k\in K}}c_{uv}(\varphi^{pk}_{uv}+\varphi^{dk}_{uv})}^{E} \tag{2.1}$$

s.t.
$$\sum_{u\in U}\varphi_{ku}=\sum_{i\in D_k}q_i \qquad \forall k\in K \tag{2.2}$$
$$\sum_{u\in U}\varphi_{uk}=\sum_{i\in P_k}q_i \qquad \forall k\in K \tag{2.3}$$
$$\varphi_{uk}+\varphi_{ku}\le\chi_{ku}\sum_{i\in P_k\cup D_k}q_i \qquad \forall k\in K,u\in U \tag{2.4}$$
$$\chi_{ku}\le y_u \qquad \forall k\in K,u\in U \tag{2.5}$$
$$\sum_{u\in U}\chi_{ku}\le B \qquad \forall k\in K \tag{2.6}$$
$$\varphi_{ku}+\sum_{\substack{v\in U\\ v\neq u}}\varphi^{dk}_{vu}=\sum_{\substack{v\in U\\ v\neq u}}\varphi^{dk}_{uv}+\sum_{r\in\mathscr{R}^{+d}_u}q_k(r)x_r \qquad \forall k\in K,u\in U \tag{2.7}$$
$$\sum_{r\in\mathscr{R}^{-p}_u}q_k(r)x_r+\sum_{\substack{v\in U\\ v\neq u}}\varphi^{pk}_{vu}=\sum_{\substack{v\in U\\ v\neq u}}\varphi^{pk}_{uv}+\varphi_{uk} \qquad \forall k\in K,u\in U \tag{2.8}$$
$$\sum_{r\in\mathscr{R}_i}x_r=1 \qquad \forall i\in P\cup D \tag{2.9}$$
$$x_r\le y_u \qquad \forall r\in\mathscr{R}^+_u\cup\mathscr{R}^-_u,u\in U \tag{2.10}$$
$$-\delta^-_h\le\sum_{r\in\mathscr{R}^-_h}x_r-\sum_{r\in\mathscr{R}^+_h}x_r\le\delta^+_h \qquad \forall h\in U\cup L \tag{2.11}$$
$$\sum_{\substack{v\in U\\ u<v}}z_{uv}+\sum_{\substack{v\in U\\ v<u}}z_{vu}=2y_u \qquad \forall u\in U \tag{2.12}$$
$$\sum_{\substack{u\in S\\ v\notin S\\ u<v}}z_{uv}+\sum_{\substack{u\notin S\\ v\in S\\ u<v}}z_{uv}\ge 2(y_w+y_{w'}-1) \qquad \forall S\in\mathcal{S}_U,w\in S,w'\in U\backslash S \tag{2.13}$$
$$\sum_{k\in K}(\varphi^{dk}_{uv}+\varphi^{pk}_{uv})\le q_{uv}\ z_{uv} \qquad \forall(u,v)\in A'_U \tag{2.14}$$
$$\sum_{k\in K}(\varphi^{dk}_{vu}+\varphi^{pk}_{vu})\le q_{vu}\ z_{uv} \qquad \forall(u,v)\in A'_U \tag{2.15}$$
$$\sum_{k\in K}(\phi_{ku}+\phi_{uk})\le Q_u y_u \qquad \forall u\in U \tag{2.16}$$
$$\phi_{ku}\ge\varphi_{ku} \qquad \forall k\in K,u\in U \tag{2.17}$$
$$\phi_{ku}\ge\sum_{r\in\mathscr{R}^{+d}_u}q_k(r)x_r \qquad \forall k\in K,u\in U \tag{2.18}$$
$$\phi_{uk}\ge\varphi_{uk} \qquad \forall k\in K,u\in U \tag{2.19}$$
$$\phi_{uk}\ge\sum_{r\in\mathscr{R}^{-p}_u}q_k(r)x_r \qquad \forall k\in K,u\in U \tag{2.20}$$
$$\sum_{u\in U}y_u\le N \tag{2.21}$$
$$y_u\in\{0,1\} \qquad \forall u\in U$$
$$z_{uv}\in\{0,1\} \qquad \forall(u,v)\in A'_U$$
$$\chi_{ku}\in\{0,1\} \qquad \forall k\in K,u\in U$$

$$x_r \in \{0,1\} \qquad \forall r \in \mathscr{R}$$
$$\varphi_{uv}^{pk}, \varphi_{uv}^{dk} \geq 0 \qquad \forall k \in K, (u,v) \in A_U$$
$$\varphi_{ku}, \varphi_{uk}, \phi_{uk}, \phi_{ku} \geq 0 \qquad \forall k \in K, u \in U$$

The objective function minimizes the sum of the installation cost of selected UDCs (A), the connection costs of the ring (B), the routing costs of the second-level delivery and pick-up paths (C) and the transportation costs between gates and UDCs (D) and between UDCs on the ring (E). Constraints (2.2) and (2.3) express that the flow that leaves or goes to gate k is equal to the total quantity of demands coming from or going to k, while constraints (2.4) force the flow of commodity k that enters or exits u to be 0 if commodity k does not directly exchange goods with site u; (2.5) express the straightforward relation between location and service variables, while (2.6) bounds to B the number of UDCs with which a gate can send and receive some demand. Flow conservation constraints (2.7) ensure that for every $u \in U$ and $k \in K$, the total quantity of commodity k that arrives at u either directly from k or through the ring (k-outflows) is equal to the total quantity that leaves u to go to the ring (again k-outflows) or leaves u for delivery service paths. The reverse holds for pick-up flows in constraints (2.8). Constraints (2.9) assign each delivery or pick-up demand to exactly one route. Constraints (2.10) assert that no route can start or end at a UDC if it is not chosen. (2.11) are the fleet rebalancing constraints at every UDC and SPL. Relations (2.12) and (2.13) concern the construction of a ring with the selected UDCs. The latter are subtour elimination constraints adapted to the fact that the nodes to be connected are not known a priori. Section 2.5.2 is devoted to this aspect. (2.14) and (2.15) are capacity constraints on the arcs of the ring. Note that, given two UDCs $u, v \in U$ and such that $u < v$, all flows on both (u,v) and (v,u) depend on z_{uv}.

Relations (2.16)–(2.20) are storage capacity constraints on UDCs, in the following sense. Suppose that a delivery demand $i \in D$ is transported from k_i to a UDC u and then shipped starting from UDC v. In order to model the cross-docking operations, i is considered to occupy capacity on both u and v (whereas possible intermediate UDCs are not concerned), unless a direct shipment takes place (i.e. $v \equiv u$), in which case a portion q_i, and not 2 q_i, of Q_u is occupied. This means that given $k \in K$, the occupation of u due to demands in D_k is not the sum of φ_{ku} and $\sum_{r \in \mathscr{R}_u^{+d}} q_k(r) x_r$ (i.e. the sum of demands of D_k shipped from u), but the maximum of the two. Terms ϕ_{ku} and ϕ_{uk} account for that, respectively, for delivery ((2.17), (2.18)) and pick-up ((2.19), (2.20)); thus, the sum on the left-hand side of (2.16) is an upper bound on the occupation of u. A consequence of this modelling choice is that the condition $\sum_{i \in D \cup P} q_i > \max_{u,v \in U, u \neq v} (Q_u + Q_v)$ is enough to guarantee that at least three UDCs will be opened, since the left-hand-side term is the minimum value of the overall occupation of UDC capacities, which occurs in the

particular case of all direct shipments. Finally, (2.21) is the budget constraint on the maximum number of UDCs that can be built.

2.5 Strengthening the model

In this paragraph, we add some constraints that are redundant for the initial formulation, but useful for speeding up the resolution. Moreover, we show how to write the subtour elimination constraints in a more effective way; finally, we add some valid inequalities derived from the literature.

2.5.1 Logical inequalities

Constraints (2.10) and the objective function are enough to relate variables y and x_r. Nevertheless we add complementary constraints (2.22):

$$y_u \leq \sum_{r \in \mathscr{R}_u^{+d}} x_r + \sum_{r \in \mathscr{R}_u^{-p}} x_r \quad \forall u \in U \tag{2.22}$$

These constraints essentially assert that there must be delivery routes starting at u, or pick-up routes ending at u, to justify the choice to open it. The reason for this is that, e.g. for the delivery, the roles of the starting point and the endpoint of a delivery path are strongly unbalanced: the latter simply acts like a parking point (i.e. if it is a UDC, then its function does not differ from that of an SPL) and is affected only in terms of rebalancing, while the former is essential for the delivery service and is affected also in terms of capacity occupation. With (2.22), we explicitly forbid the opening of a UDC only to perform parking functions. However, once we decide not to open it, then delivery paths ending at it and pick-up paths starting from it are also forbidden by (2.10).

2.5.2 Subtour elimination inequalities

As mentioned before, constraints (2.12) and (2.13) are a variant of the usual Subtour Elimination Constraints for the case when the nodes to be connected in the tour are not known a priori. This is a substantial difference w.r.t. the standard STSP (Symmetric Traveling Salesman Problem) and some other more specific variants, in which the nodes to be visited are not known except for a subset of compulsory nodes (like in the Orienteering Problem, see [Laporte, 1986] and [Fischetti et al., 1998], the Prize Collecting TSP, see [Ausiello et al., 2007] for instance, or the Steiner TSP, see [Letchford et al.,

2012]). Subtour Elimination Constraints (2.13) are taken from the literature on Generalized Cut Constraints and on the Generalized Traveling Salesman Problem (GTSP); one can refer, for example, to [Fischetti et al., 1997] to gain an insight into those subjects. However, constraints (2.13) can be substituted in two different ways, as we explain in the following.

2.5.2.1 First form

We add new variables $\zeta_u \in \{0,1\}$ for every $u \in U$, and replace (2.13) with the following families of constraints:

$$\zeta_u \leq y_u \qquad \forall u \in U \tag{2.23}$$

$$\sum_{u \in U} \zeta_u = 1 \tag{2.24}$$

$$\sum_{\substack{u \in S \\ v \in S \\ u<v}} z_{uv} \leq \sum_{u \in S} y_u + 1 - y_w - \sum_{u \in S} \zeta_u \qquad \forall S \in \mathcal{S}_U, \forall w \in U \backslash S \tag{2.25}$$

By doing so, we dramatically reduce the overall number of constraints, since if we compare (2.25) and (2.13), then for each subset S we have $\mathcal{O}(|U|)$ constraints instead of $\mathcal{O}(|U|^2)$. Both the equivalence between constraints (2.23)–(2.25) and relations (2.13) and the proof of this equivalence are inspired by the GTSP and Generalized Cut Constraints theory and the aforementioned [Fischetti et al., 1997].

Lemma 2.1. *The aggregate of constraints* (2.12) *and* (2.23)–(2.25) *allows us to avoid subtours in the MRLRP case, i.e. when the nodes to be connected by the tour are not known a priori.*

Proof. Let S be an element of $\mathcal{S}_U$. Let us also suppose, without loss of generality, that $\sum_{u \in U} y_u \geq 3$. We can distinguish four cases:

1. If $\sum_{u \in S} y_u = 0$, from (2.23)-(2.25) we obtain $\sum_{u,v \in S: u<v} z_{uv} \leq 0$, i.e. no edge with both endpoints in S is allowed.
2. If $0 < \sum_{u \in S} y_u < \sum_{u \in U} y_u$ and $\sum_{u \in S} \zeta_u = 1$, then there exists $w \in U \backslash S : y_w = 1$, and for such a w, constraint (2.25) becomes:

$$\sum_{\substack{u,v \in S \\ u<v}} z_{uv} \leq \sum_{u \in S} y_u - 1$$

so in this case subtours in S are indeed forbidden.

3. If $0 < \sum_{u\in S} y_u < \sum_{u\in U} y_u$ and $\sum_{u\in S} \zeta_u = 0$, then again there is a $w \in U\backslash S : y_w = 1$, and for such a w, constraint (2.25) becomes:

$$\sum_{\substack{u,v\in S\\ u<v}} z_{uv} \leq \sum_{u\in S} y_u$$

which do not prevent having a subtour in S, but constraint (2.25) on $U\backslash S$ (which also belongs to $\mathcal{S}_U$) and for some $w \in S : y_w = 1$ becomes:

$$\sum_{u,v\in U\backslash S} z_{uv} \leq \sum_{u\in U\backslash S} y_u + 1 - y_w - \sum_{u\in U\backslash S} \zeta_u = \sum_{u\in U\backslash S} y_u - 1$$

this prevents subtours in $U\backslash S$, and consequently in S.

4. If $\sum_{u\in S} y_u = \sum_{u\in U} y_u$ (i.e. $\sum_{u\in U\backslash S} y_u = 0$), then $y_w = 0$ for all $w \in U\backslash S$ which implies $\sum_{u\in U} \zeta_u = 1$. So for any $w \in U\backslash S$ the constraint becomes

$$\sum_{u,v\in S:u<v} z_{uv} \leq \sum_{u\in S} y_u$$

Thus, subtours are allowed in S. A single subtour should be allowed in this case in S, but not several. Indeed, it is not possible to have several subtours in S as if a subtour spans a subset $S' \subset S$, this subtour would be eliminated by constraint (2.25) for S' instead of S. This completes the proof.

□

2.5.2.2 Second form

We add a fictitious UDC $\hat{u}$. For ease of notation, let $(\forall u \in U)\ \hat{u} < u$. Then we define:

- the extended UDC set $\hat{U} = U \cup \{\hat{u}\}$ and arc subset $\hat{A}'_U = A'_U \cup \{(\hat{u}, u) : u \in U\}$. We will talk about *actual* UDCs and ring arcs, to distinguish them from $\hat{u}$ and the arcs of $\hat{A}'_U\backslash A'_U$;
- ring variables $z_{\hat{u}u}, u \in U$ for the new arcs, all associated with a cost 0;
- the cut-set $\hat{D}(S) = \{(u, v) : u \in S, v \in \hat{U}\backslash S\} \cup \{(u, v) : u \in \hat{U}\backslash S, v \in S\}$ of set $S \in \mathcal{S}_U$. Note that no $S \in \mathcal{S}_U$ includes fictitious node $\hat{u}$, hence:

$$(\forall S \in \mathcal{S}_U) \quad \{(\hat{u}, u) : u \in S\} \subset \hat{D}(S).$$

We can now replace (2.13) with the following families of constraints:

$$\sum_{u \in U} z_{\hat{u}u} = 2 \tag{2.26}$$

$$y_u \geq z_{\hat{u}u} \qquad \forall u \in U \tag{2.27}$$

$$z_{uv} \geq z_{\hat{u}u} + z_{\hat{u}v} - 1 \qquad \forall (u,v) \in A'_U \tag{2.28}$$

$$\sum_{(u,v) \in \hat{D}(S)} z_{uv} \geq 2y_w \qquad \forall S \in \mathcal{S}_U, \forall w \in S \tag{2.29}$$

Constraints (2.26) impose the connection of $\hat{u}$ with two UDC, which are then chosen (i.e. the corresponding y variables are forced to 1, due to constraints (2.27)) and connected between them (as a consequence of constraints (2.28)). Finally, constraints (2.29) are Generalized Subtour Elimination Constraints (GSECs). As one can observe, the overall number of constraints decreases significantly, as by comparing (2.29) and (2.13) we have $\mathcal{O}(|U|)$ constraints instead of $\mathcal{O}(|U|^2)$ for each $S \in \mathcal{S}_U$, as it was for relations (2.25) (see section 2.5.2.1).

This second technique to substitute constraints (2.13) is based on a simple idea. Since the main issue in preventing subtours is the lack of a subset of compulsory nodes, we add one, i.e. $\hat{u}$: this way, we can exploit the GSECs like it is done, for example, in [Fischetti et al., 1998], in order to get a unique tour that visits $\hat{u}$ and the subset of chosen actual UDCs of U. The requested actual ring, i.e. that makes use of arcs of A'_U only, is then obtained by *shortcutting* $\hat{u}$ with a *chord* that connects the two actual UDC $u, v \in U$ s.t. $z_{\hat{u}u} = z_{\hat{u}v} = 1$ (constraints (2.27), (2.28)). Note that the cost 0 associated with fictitious arcs $(\hat{u}, u), u \in U$, assures to obtain a least cost actual ring.

Since this technique actually relies on the Generalized Subtour Elimination Constraints, the equivalence between (2.13) and the aggregate of constraints (2.26)–(2.29) does not need to be proven.

2.5.3 Path inequalities

Following an idea of [Berger, 1997], which was later reused in [Berger et al., 2007], we replace constraints (2.10) by the family of constraints:

$$\sum_{r \in \mathscr{R}_i \cap (\mathscr{R}_u^+ \cup \mathscr{R}_u^-)} x_r \leq y_u \qquad \forall u \in U, \forall i \in P \cup D \tag{2.30}$$

These constraints are valid, since at most one route among those that start or end at u can be selected to serve demand i. Moreover, family (2.10) has an exponential number of constraints, while those of family (2.30) are in polynomial number. Most of all, the above replacement makes the continuous relaxation of the MILP model stronger, thus tightening the lower bound. In [Berger, 1997], the author applied this substitution to

a Set-Partitioning-based formulation of LRP, proving the new constraints to be rank-1 Chvátal cuts.

2.6 A Matheuristic Algorithm

This section presents GALW, a matheuristic to solve large MRLRP instances.
The exact approach often fails to solve medium- to large-sized instances, due to the huge number of possible routes; even when they can be solved, the computational times can grow considerably, due to the fact that the numerous constraints belong to different families (notably assignment, STSP and flow constraints), which are strongly interconnected. The choice of what heuristic approach should be used to address an instance of MRLRP is not immediate. The complexity, heterogeneity and interdependence of the different components of the problem (location, network design, routing, flow) make the definition of one or more neighborhoods of a MRLRP solution a difficult task; therefore, a local search based technique would be quite hard to design.
This is the main reason that led us to choose a decomposition-based matheuristic approach. GALW is based on two ideas: to generate only a subset of routes and to decompose the model into three phases. Each phase is devoted to one or more specific aspects of the problem solved to optimality. The procedure to generate routes, which is the only heuristic one, is a nearest neighbor algorithm conveniently modified to take into account the load and trip length constraints of the second-level vehicles. The other three phases consist of a MILP model for the generalized assignment problem, an STSP and a multicommodity flow problem on a ring.

2.6.1 Route generator

Let $R = (i_1, \ldots, i_{|R|})$ be a sequence of demands (as seen before in section 2.3), and $r(R) \in \mathscr{R}$ the route (either an inter-depot route or a back-to-depot route) we obtain when R is completed with a starting point that is the nearest to i_1 and an ending point that is the nearest to $i_{|R|}$. The nearest starting and ending points of demand i may differ, due to the role of SPLs, as previously discussed. We can say that R is a valid sequence of demands if $c(r(R)) \leq M$ and $q(r(R)) \leq q$, i.e. if $r(R)$ is a feasible route. Note that we can have many feasible routes, other than $r(R)$, that descend from R. We build a set $\overline{\mathscr{R}} \subset \mathscr{R}$ of routes by using a nearest neighborhood procedure. We denote by $\overline{\mathscr{R}}_i \subset \mathscr{R}_i$ the set of routes serving demand i that we progressively build. The algorithm ensures that $\overline{\mathscr{R}}$ contains routes of different maximum lengths $M^t = \beta^t M \leq M$, where $\beta < 1$, $t = 0 \ldots t^*$, since it would be hard to find solutions if all the routes had the same maximum length M. The generation process is characterized by parameters β and τ,

$0 < \tau < \beta < 1$; it begins with $t = 0$ and progressively increases t until $\beta^t < \tau$; thus, $t^* = \lfloor \log_\beta \tau \rfloor$. Each new route is initialized with a demand i, which is randomly chosen among those with a number of newly generated (t-th iteration) visiting paths which is less than the average over the set of demands. Then, the algorithm iteratively performs the following two steps, until no more demands can be inserted without exceeding M^t or q:

1. Choose a demand i to add to the current sequence R by means of a nearest neighbor rule that minimizes the function $d(R, i)$ defined below;
2. Improve the updated R with a local search step, which consists of a sequence of 2-opt moves on $r(R)$ until no further improvement is possible.

The function to minimize is $d(R,i) = p_q \frac{q(r)+q_i}{q} + p_M \frac{\min\{c(r(i,i_1..i_{|R|})), c(r(i_1..i_{|R|},i))\}}{M} + p_\omega \frac{|\overline{\mathscr{R}}_i|}{\omega}$, where the second term is the minimum between the costs to insert new demand i at the beginning or at the end of R, and $p = (p_q, p_M, p_\omega)$ is a vector of weights. The only difference from classical 2-opt techniques (see e.g. [Lin and Kernighan, 1973]) is that in our case, a 2-opt move that changes, for instance, the first-visited demand could also alter the nearest starting point of the route; similarly, this can be said for moves that change the last-visited demand. This must be taken into account in the computation of the saving of such moves. Figure 2.2 explains the difference between a move that does not alter the extreme points of the current route (b) and what happens when such a move is performed (c). The t-th iteration stage terminates when each demand i is covered by at least ω routes, to guarantee that $\overline{\mathscr{R}}_i$ contains a minimum number of routes. Since β, ω and τ are constants, the complexity of the overall route generation algorithm turns out to be $\mathcal{O}(n^4)$. We use $\overline{\mathscr{R}}$ as a good set of feasible routes in the following section.

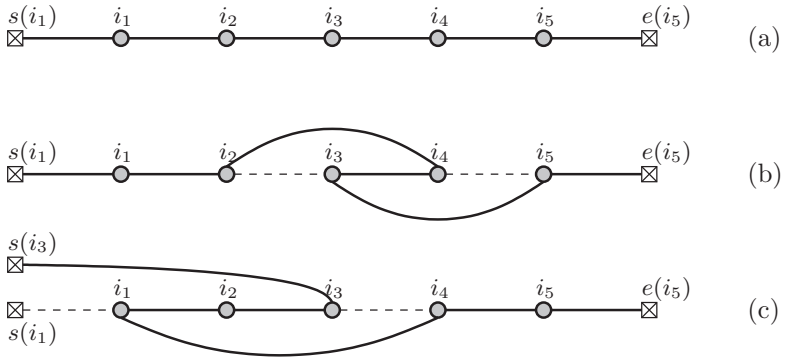


FIGURE 2.2: Illustration of the principle of a 2-opt move in the local search step of the route generation. We respectively note $s(i)$ and $e(i)$ as the nearest starting point and nearest ending point for demand i. When the move does not change the first- and last-visited demand, the saving is computed as usual, as in (b), in which it amounts to $c_{i_2,i_3} + c_{i_4,i_5} - (c_{i_2,i_4} + c_{i_3,i_5})$. However, in a case like (c) in which the first-visited demand changes, it may also be possible to alter the starting point (in the example $s(i_3) \neq s(i_1)$): the saving will be $c_{s(i_1),i_1} + c_{i_3,i_4} - (c_{s(i_3),i_3} + c_{i_1,i_4})$. Similar considerations apply when we change the last-visited demand.

2.6.2 MILP Assignment subproblem

The second stage of GALW deals with the decisions concerning routing, UDC activation, and demand assignment. The two distribution levels are disjoint in the graph representation: the problem is solved as an assignment of demands both at the first level (from gates to UDCs) and at the second level, while taking into account the UDCs' capacities. We neither have to build the ring nor have to route flows in it; however, if for instance a delivery demand $i \in D_k$ is brought from gate k to a UDC u and later delivered from another UDC v, then we have either a direct shipment if $u = v$ or an implicit use of the external network, if not, and similarly for pick-up. We use a subset of the variables of the MILP general model for the MRLRP (see section 2.4), namely: location variables y_u, $u \in U$, routing variables x_r, where $r \in \overline{\mathscr{R}}$, service variables χ_{ku}, $(k,u) \in K \times U$ and a subset of flow variables: φ_{ku}, φ_{uk}, ϕ_{ku} and ϕ_{uk}, $k \in K$, $u \in U$. All these variables have the same meaning and domain as in the general model, with the obvious exception of the routing variables x_r. The objective function is composed of terms (A), (C) and (E) of that of the MILP general model. The constraints are a subset of those of the general model, namely (2.2)–(2.6), (2.9), (2.11), (2.16)–(2.21), (2.22) and (2.30), along with the bounds on the variables inherited from the MILP general model. The variables and constraints removed are those concerning the ring and the flows in it. Since the MILP model defined so far does not have the terms (B) and (D) of the initial model, it could decide to open a subset of UDCs that later determine a ring that turns out to be too expensive in terms of the fixed cost and flow transportation cost. It is therefore necessary to estimate these costs a priori. To achieve this, first of all we find the solution to the STSP on any subset $O \subset U$ of the UDCs such that $3 \leq |O| \leq N$; then, for each $u \in U$ and any $o = 3 \ldots N$, we determine the cost $F_{u,o} = \frac{1}{2} g_{uv_o} + \frac{1}{2} g_{uw_o}$, where v_o and w_o are the nodes connected to u in the least costly ring of exactly o nodes among those that include u. Note that cost function $F_{u,o}$ can be fully determined a priori in a preprocessing phase. Then, we add to the model a new set ξ_o^u of binary variables, $u \in U$, $o = 3 \ldots N$: $\xi_o^u = 1$ means that $u \in O$ and $|O| = o$. Finally, we introduce the following constraints:

$$\sum_{v \in U \setminus u} y_v \geq \sum_{o=3}^{N} (o-1)\xi_o^u \qquad \forall u \in U \tag{2.31}$$

$$\sum_{v \in U \setminus u} y_v \leq \sum_{o=3}^{N} (o-1)\xi_o^u + N(1 - y_u) \qquad \forall u \in U \tag{2.32}$$

$$\xi_o^u \in \{0,1\} \qquad \forall u \in U, o \in \{3..N\}$$

and add to the objective function the term $\sum_{u \in U} \sum_{o=3}^{N} F_{u,o} \xi_o^u$, which represents a lower bound of the fixed installation cost of the ring. The impact of the flow transportation

costs and ring arc capacities is tackled as follows. For each commodity k and UDC u, terms $(\varphi_{ku} - \sum_{r \in \overline{\mathscr{R}}_u^{+d}} q_k(r)x_r)$ and $(\sum_{r \in \overline{\mathscr{R}}_u^{-p}} q_k(r)x_r - \varphi_{uk})$, when positive, represent the quantities of goods that must leave u; this allows us to compute the total amount of goods that u must send through the ring, for which we can both impose an upper bound and compute a transportation cost lower bound to add to the objective function. This is achieved by considering, respectively, the two arcs $(u,v),(u,w) \in A_U$ such that the total capacity $q_{uv}+q_{uw}$ is maximum and the two arcs $(u,v'),(u,w') \in A_U$ that minimize the per-flow-unit cost $\frac{1}{2}(c_{uv'} + c_{uw'})$.

2.6.3 Ring construction

Given the subset $O \subset U$ of the chosen UDCs, the third stage of GALW consists of finding the way to connect them in a ring with the minimum cost. The costs to consider for this problem are $g_{uv}, (u,v) \in A'_U, u,v \in O$, i.e. the costs to connect two UDCs u and v in both directions. Thus, even if the original subgraph that connects UDCs is oriented, the problem to solve is a Symmetric TSP (STSP), and is solved using CONCORDE, the well-known computer code for the STSP ([Applegate et al., 2006]). In spite of being NP-hard, this stage of GALW requires negligible solution times, due to the relatively small number of UDCs. The flows on the resulting ring can circulate in two directions, which in the following will be referred to as *clockwise* and *counterclockwise* for the sake of simplicity.

2.6.4 Ring flows

Once we have solved the assignment subproblem and built the ring, we have to send flows along it to accomplish indirect shipments. We have as many commodities as twice the number of original commodities, i.e. $2|K|$, since we distinguish between pick-up and delivery flows. For each commodity k, and for both pick-up and delivery, we have UDCs with available goods, i.e. the ones that received goods either from a gate (case $\varphi_{ku} > \sum_{r \in \overline{\mathscr{R}}_u^{+d}} q_k(r)x_r$) or from pick-up services (case $\sum_{r \in \overline{\mathscr{R}}_u^{-p}} q_k(r)x_r > \varphi_{uk}$); those with a request for goods, because they have to send them either to a gate (case $\sum_{r \in \overline{\mathscr{R}}_u^{-p}} q_k(r)x_r < \varphi_{uk}$), or to delivery services (case $\varphi_{ku} < \sum_{r \in \overline{\mathscr{R}}_u^{+d}} q_k(r)x_r$); and those that are not involved. In the following, we refer, for instance, to deliveries of commodity k: let $O_{dk}^+ \subset O$ be the subset of chosen UDCs with availability and $O_{dk}^- \subseteq O \backslash O_{dk}^+$ be the subset of those with a request. In addition, let $O_{dk}^2 = \{\{u,v\} : u \in O_{dk}^+, v \in O_{dk}^-\}$ be the set of source–sink pairs $\{u,v\}$, and Φ_{uv}^{dk} be the flow we send from source u to sink v, again for the delivery of commodity k. The total availability of sites $u \in O_{dk}^+$ is equal to the total demand of sites $v \in O_{dk}^-$; goods must be sent from the former to

the latter, in such a way that each $u \in O^+_{dk}$ completely consumes its availability and the need of each $v \in O^-_{dk}$ is fulfilled. This results in a series of straightforward flow balance constraints on terms Φ^{dk}_{uv} (and, in general, for terms Φ^{dk}_{uv} and Φ^{pk}_{uv} of each k). Now, let $\theta'_{\{uv\}}$ and $\theta''_{\{uv\}}$, respectively, be the clockwise path and the counterclockwise path from u to v, $u, v \in O$, along the ring. Any flow we must send from u to v can be split and sent partly clockwise and partly counterclockwise, i.e. along $\theta'_{\{uv\}}$ or $\theta''_{\{uv\}}$. This means that any clockwise-oriented arc of the ring a must transport all of the flows (regardless of the commodity) from u to v of any couple $\{u, v\}$ such that $a \in \theta'_{\{uv\}}$; the analogous situation applies to counterclockwise-oriented arcs. The last stage of GALW consists of a ring multiflow problem with multiple sources and sinks for each commodity: we solve it by means of a another LP model, in which we impose the balance between availabilities and requests due to pick-up and delivery of each commodity and decide how to split the flows in such a way as to respect the arc capacities and minimize the flow costs. The flow on each ring arc a is:

$$
\begin{aligned}
f_a &= \textstyle\sum_k (\varphi^{dk}_a + \varphi^{pk}_a) = \\
&= \begin{cases}
\displaystyle\sum_k \left(\sum_{\substack{\{u,v\} \in O^2_{dk}: \\ a \in \theta'_{\{uv\}}}} (\Phi^{dk}_{uv})' + \sum_{\substack{\{u,v\} \in O^2_{pk}: \\ a \in \theta'_{\{uv\}}}} (\Phi^{pk}_{uv})' \right) & , \ a \text{ clockwise-oriented} \\
\displaystyle\sum_k \left(\sum_{\substack{\{u,v\} \in O^2_{dk}: \\ a \in \theta''_{\{uv\}}}} (\Phi^{dk}_{uv})'' + \sum_{\substack{\{u,v\} \in O^2_{pk}: \\ a \in \theta''_{\{uv\}}}} (\Phi^{pk}_{uv})'' \right) & , \ \text{otherwise}
\end{cases}
\end{aligned}
\tag{2.33}
$$

where $(\Phi^{dk}_{uv})'$ and $(\Phi^{dk}_{uv})''$ are the parts of Φ^{dk}_{uv} that we send clockwise and counterclockwise, respectively. Flows φ^{dk}_a and φ^{pk}_a are the same as the outflows and inflows of the general model for the MRLRP: therefore, the constraints of the ring multiflow LP model are simply the capacity constraints on the arcs a of the ring, which we derive from the general model (see (2.14)–(2.15)), along with the aforementioned flow balance constraints on variables Φ^{dk}_{uv} and Φ^{pk}_{uv}. The objective function reduces to term (D) of that of the general model. Since we are dealing with a linear problem, this fourth stage of GALW turns out to be polynomial. Figure 2.3 shows an example.

2.7 Computational results

In this section, we show the computational experiments that were conducted to evaluate the three methods developed:

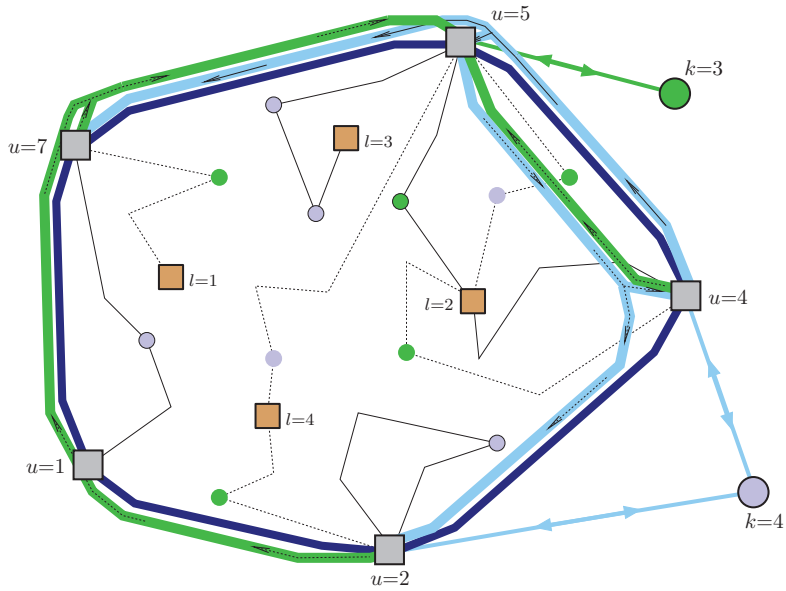


FIGURE 2.3: A view of a solution to the ring multiflow stage. The assignment step provides a selection of UDCs, determines service paths and UDC-gate flows; once the ring has been built, the ring multiflow LP subproblem looks for the way to accomplish indirect shipments with the minimum cost. In this example, the service paths and UDC-gate flows are the same as in Figure 2.1. Similarly to what has been done for the service paths, the ring flows due to delivery duties are denoted by continuous stroke arrows, while dashed stroke arrows indicate flows intended to serve pick-up duties. Here we have a close-up view regarding commodities k=3 and k=4.

- the *exact method* corresponds to the complete MILP model of Section 2.4 strengthened with the valid inequalities proposed in Section 2.5. It will be referred to also as *X method*;
- the *heuristic* method is the algorithm called GALW presented in section 5. The coded algorithm has been improved with two implementation tricks in the MILP assignment subproblem. When an instance contains b arcs $(u, v) \in A'_U$ with installation costs g_{uv} considerably greater than the average, we slightly modify the objective function by imposing to pay g_{uv} if both endpoints u and v are activated. The second one consists of stopping CPLEX when the gap to optimality reaches a threshold α, then the first S solutions are stored and the last two GALW stages are applied to the whole set. Only the best global solution is saved.
- the *hybrid* method, which we will also refer to as *Y method*, consists of solving the MILP model using $\overline{\mathscr{R}}$ instead of $\mathscr{R}$. This method is useful for checking the effectiveness of the GALW route generator.

The instance generation process is briefly illustrated below. Then, the computational results are reported and discussed. The exact algorithm and the assignment and ring multiflow steps of GALW have been implemented using the CPLEX Callable Library of the IBM ILOG CPLEX suite, version 12.5.

2.7.1 Test instances

No previously generated instances can be found in the literature since to the best of our knowledge the MRLRP is a new problem proposed here for the first time. Neither could we take advantage of the WLRP instances proposed in [Perl and Daskin, 1985], since they are a small number of very simple instances. Therefore, we derive a set of MRLRP instances from a set of existing benchmark CLRP instances proposed by [Prins et al., 2006], which we complete with additional MRLRP features.
In the following, a brief explanation of the instance generation strategy is presented: for a detailed description, we refer the reader to Appendix D.2. The computational sessions were conducted with the aim of:

1. establishing a relation between the hardness of an instance and its dimensional features, notably the size of U, K, L, D and P;
2. evaluating the degree to which GALW performances depend on the decomposition process.

For this reason, we create not one but a *collection* of instances from each CLRP instance. To pursue point 1, the instances in a collection are divided into five *scenarios*: the *base scenario* or *scenario 0*, which features an initial size of sets U, K, L, D and P, and four more scenarios in which, one at a time, each of the aforementioned sets is enlarged with additional elements: thus, we have *scenario 1* with additional demands in both D and P, *scenario 2* with additional UDCs in U, *scenario 3* with additional gates in K, and lastly *scenario 4* with additional SPLs in L. By the way, each instance verifies $|D| = |P|$. To address point 2, each scenario contains a set of four *economic instances* that differ by the ring construction costs (high/low) and transportation costs (high/low). In addition, for each scenario we also create what we call an *ecological instance*. In an ecological instance, routing and flow transportation costs are replaced by pollution indicators, whereas installation costs are null for both UDCs and ring arcs, so as to evaluate, given a scenario, which system configuration has the highest degree of environmental sustainability. Note that with such an assumption on installation costs, the budget constraint has a greater importance. Economic types will be referred to with the acronym of the corresponding cost structure: type L|L (low construction costs, low transportation costs), type L|H (low/high costs), type H|L (high/low) and type H|H (high/high costs); the ecological type will be referred to as type G (for *green*).
Table 2.1 maps benchmark CLRP instances and derived MRLRP collections, showing for each collection the main features of its scenarios such as the minimal and maximum values of $|K|$, $|U|$, $|L|$ and $|P| = |D|$, along with the features that are common to each scenario: the values of q, M and the number of clusters into which customers are

grouped. As a reminder, in [Prins et al., 2006], the benchmark names follow the pattern n-m-c[b], where n is the number of customers, m the number of depots and c the number of clusters, while the final 'b' indicates whether the capacity of the vehicle is 150 or 70. Therefore, the number of clusters and the value of q are always the same as the original benchmark. In each scenario, we always assume $N = |U|$ for the economic instances,

benchmark	collection	#instances	$\|K\|$	$\|U\|$	$\|L\|$	$\|P\|, \|D\|$	#clusters	N_G	q	M
20-5-1	galwc01	5×(4+1)	5,10	5,10	5,10	15,20	1	4,6	70	50
20-5-2	galwc02	5×(4+1)	5,10	5,10	5,10	15,20	2	4,6	70	50
20-5-1b	galwc03	5×(4+1)	5,10	5,10	5,10	15,20	1	4,6	150	65
50-5-1	galwc06	5×(4+1)	5,10	5,10	5,10	25,40	1	4,6	70	60
50-5-3b	galwc07	5×(4+1)	5,10	5,10	5,10	25,40	3	4,6	150	65
100-5-1	galwc08	5×(4+1)	5,10	5,10	10,15	50,80	1	4,6	70	50
100-5-3	galwc10	5×(4+1)	5,10	5,10	10,15	50,80	3	4,6	70	50
100-10-1	galwc11	5×(4+1)	5,10	10,15	10,15	50,80	1	6,10	70	50

TABLE 2.1: Correspondence between original CLRP benchmarks and the collections of MRLRP instances.

i.e. we relax the budget constraint, whereas we keep it for the ecological instance by imposing $N < |U|$. The column N_G shows the minimal and maximum values of N (according to $|U|$) for all of the ecological instances of the collection. Since each of the 8 collections accounts for 5 scenarios of 5 instances each, the complete instances set amounts to 200 MRLRP instances.

2.7.2 Conducted tests

The tests were performed on a Pentium dual core 2.6 Ghz machine with 3.76Gb RAM. The assignment GALW subproblem and the hybrid method were systematically imposed a time limit of 3600s, whereas for the exact method the time limit was 3600s for small instances, and 10800s for medium- and large-sized instances. Throughout the whole test campaign, the values used for the GALW route generator parameters are $\beta = 0.9$, $\tau = 0.4$, $p_q = p_M = p_\omega = 1$, while ω varies according to n in the original benchmark CLRP and the law $\omega = 10 \cdot \lceil 2 \cdot \log_5 n \rceil - 20$. The parameters of the GALW assignment subproblem are $S = 3$, $b = 3$, $\alpha = 2\%$.

The first part of our analysis follows the guidelines introduced in 2.7.1. First, we focus on one economic cost structure, then we take all instances having this cost structure in all scenarios of each collection. By doing so, we analyse how the dimensional aspects of an instance affect the performances of the three methods, all the cost parameters being the same. The chosen cost structure is that of type L|L. Further, we take the base scenario of each collection and show the results for all of its economic instances, in order to verify point 2 of section 2.7.1. Finally, we take the ecological instance in all scenarios of each collection to perform an analysis based on environmental criteria.

Tables 2.3, 2.4 and 2.5 present these results for small instances, tables 2.6, 2.7 and 2.8 concern medium-sized collections (galwc06, galwc07), whereas tables 2.10, 2.11 and 2.12 deal with large-sized (galwc08, galwc10, galwc11) collections. The complete results can be found in Tables D.5 to D.7 in Appendix D.3.
Table 2.2 offers a key for each column of the aforementioned results' tables.

notation	description
Instance	instance identifier, in the *collection-scenario-type* format
r	routes generation CPU time (s)
t	CPU preprocessing time for calculating STSP (s; GALW)
a	CPU time for solving the MILP assignment problem (s; GALW)
T	total CPU computational time, out of I/O and intermediate elaborations times (s)
$\%^r$	gap at root (X method)
$\%^z$	gap still to close, if the time limit has been exceeded
$\%^X$	gap of GALW or the Y method with the optimal solution of the X method
$\%^Y$	gap of GALW with respect to the Y method
#ps	number of paths variables in the MILP models (exact, hybrid, GALW assignment problem)

TABLE 2.2: Key of the results' tables.

2.7.3 Discussion on the results

One can observe that:

	Features							Exact(X)				GALW							Hybrid(Y)	
Instance	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	$\%^r$	r	$T/\%^z$	#ps	$\%^X$	t	r	a	T	#ps	$\%^Y$	$\%^X$	T
galwc01-0-L|L	5	5	5	15	5	70	50	3.5	9.6	24.8	21503	2.1	0.0	0.4	2.9	3.6	3122	0.0	2.1	3.6
galwc01-1-L|L	5	5	5	20	5	70	50	3.3	14.0	48.7	37839	3.3	0.0	0.7	4.2	5.2	4033	0.1	3.2	4.8
galwc01-2-L|L	5	10	5	15	10	70	50	4.5	24.9	642.5	64737	6.2	1.1	0.4	28.5	29.2	8388	2.4	3.9	37.3
galwc01-3-L|L	10	5	5	15	5	70	50	4.7	9.6	16.8	21503	3.8	0.0	0.4	4.0	4.7	3068	2.5	1.4	2.3
galwc01-4-L|L	5	5	10	15	5	70	50	4.4	13.6	31.0	31503	2.5	0.0	0.4	4.2	5.0	4348	0.0	2.5	2.7
average								4.1	14.3	152.8	35417	3.6	0.2	0.5	8.8	9.5	4592	1.0	2.6	10.1
galwc02-0-L|L	5	5	5	15	5	70	50	5.7	3.0	8.0	14066	1.4	0.0	0.3	1.6	2.2	2319	0.0	1.4	1.6
galwc02-1-L|L	5	5	5	20	5	70	50	6.0	9.0	13.3	33616	0.9	0.0	0.6	2.4	3.4	3253	0.1	0.8	1.9
galwc02-2-L|L	5	10	5	15	10	70	50	5.0	7.6	59.7	39438	3.6	1.1	0.3	4.7	5.3	4707	0.1	3.5	7.5
galwc02-3-L|L	10	5	5	15	5	70	50	3.2	3.1	2.8	14066	0.3	0.0	0.3	2.1	2.7	2324	0.1	0.2	1.4
galwc02-4-L|L	5	5	10	15	5	70	50	6.3	5.2	8.0	23190	0.2	0.0	0.3	2.4	3.1	3478	0.0	0.2	2.0
average								5.2	5.6	18.4	24875	1.3	0.2	0.4	2.6	3.3	3216	0.1	1.2	2.9
galwc03-0-L|L	5	5	5	15	5	150	65	18.6	6.6	10.8	18761	2.1	0.0	0.5	1.9	2.7	2082	0.0	2.1	2.2
galwc03-1-L|L	5	5	5	20	5	150	65	16.0	51.3	113.8	68987	3.7	0.0	1.2	2.5	4.0	3513	1.5	2.2	3.8
galwc03-2-L|L	5	10	5	15	10	150	65	19.2	24.1	181.9	79140	0.6	2.0	0.6	26.2	27.1	8449	0.0	0.6	13.7
galwc03-3-L|L	10	5	5	15	5	150	65	18.6	6.7	11.3	18761	2.7	0.0	0.5	1.7	2.5	2085	0.0	2.7	1.8
galwc03-4-L|L	5	5	10	15	5	150	65	18.1	11.0	22.7	31590	1.9	0.0	0.5	2.4	3.2	3319	0.0	1.9	2.7
average								18.1	19.9	68.1	43448	2.2	0.4	0.7	6.9	7.9	3890	0.3	1.9	4.8

TABLE 2.3: Numerical results on small instances of type L|L.

▶ On *small-sized* instances of Table 2.4, where all dimensional features are equal (scenario 0) and only the cost structure varies, the GALW heuristic has a gap ($\%^X$) to the optimal solution found by the exact method which is in most of the cases between 2% and 7%. These gaps are obtained with much lower computational time (4 times quicker on average), as expected, since the number of routes generated by GALW is 11% to 17% the number of routes found by X. Furthermore, those gaps are mostly due

Instance	Features							Exact(X)				GALW							Hybrid(Y)	
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	$\%^r$	r	$T/\%^z$	#ps	$\%^X$	t	r	a	T	#ps	$\%^Y$	$\%^X$	T
galwc01-0-L|L	5	5	5	15	5	70	50	3.5	9.6	24.8	21503	2.1	0.0	0.4	2.9	3.6	3122	0.0	2.1	3.6
galwc01-0-L|H	5	5	5	15	5	70	50	5.0	9.5	30.8	21503	4.9	0.0	0.4	4.2	4.9	3121	0.6	4.3	4.9
galwc01-0-H|L	5	5	5	15	5	70	50	6.5	9.6	16.3	21503	6.2	0.0	0.4	3.0	3.7	2926	4.6	1.7	2.4
galwc01-0-H|H	5	5	5	15	5	70	50	9.6	9.6	15.4	21503	9.3	0.0	0.4	7.4	8.1	3021	5.8	3.7	2.9
average								6.2	9.6	21.8		5.6	0.0	0.4	4.4	5.1	3048	2.8	3.0	3.5
galwc02-0-L|L	5	5	5	15	5	70	50	5.7	3.0	8.0	14066	1.4	0.0	0.3	1.6	2.2	2319	0.0	1.4	1.6
galwc02-0-L|H	5	5	5	15	5	70	50	10.0	3.0	10.3	14066	5.9	0.0	0.3	2.3	3.0	2418	1.3	4.6	2.0
galwc02-0-H|L	5	5	5	15	5	70	50	5.8	3.0	9.5	14066	1.7	0.0	0.3	2.3	2.9	2352	0.0	1.7	1.9
galwc02-0-H|H	5	5	5	15	5	70	50	7.4	3.0	4.0	14066	4.7	0.0	0.3	2.4	3.1	2421	0.2	4.5	1.7
average								7.2	3.0	8.0		3.4	0.0	0.3	2.1	2.8	2378	0.4	3.0	1.8
galwc03-0-L|L	5	5	5	15	5	150	65	18.6	6.6	10.8	18761	2.1	0.0	0.5	1.9	2.7	2082	0.0	2.1	2.2
galwc03-0-L|H	5	5	5	15	5	150	65	16.6	6.6	23.6	18761	6.3	0.0	0.5	4.9	5.7	2099	1.5	4.8	2.5
galwc03-0-H|L	5	5	5	15	5	150	65	26.5	6.6	20.2	18761	6.6	0.0	0.5	1.9	2.7	2012	1.2	5.4	1.8
galwc03-0-H|H	5	5	5	15	5	150	65	25.3	6.6	23.5	18761	6.9	0.0	0.5	5.0	5.8	2129	2.4	4.6	2.2
average								21.8	6.6	19.5		5.5	0.0	0.5	3.4	4.2	2080	1.3	4.2	2.2

TABLE 2.4: Numerical results on instances of all types in scenario 0 of small-sized collections.

Instance	Features							Exact(X)				GALW							Hybrid(Y)	
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	$\%^r$	r	$T/\%^z$	#ps	$\%^X$	t	r	a	T	#ps	$\%^Y$	$\%^X$	T
galwc01-0-G	5	5	5	15	4	70	50	1.1	10.1	8.2	21503	8.1	0.0	0.4	1.9	2.5	3072	2.4	5.8	2.8
galwc01-1-G	5	5	5	20	4	70	50	1.4	14.8	22.5	37839	6.3	0.0	0.7	3.0	3.9	4094	0.3	6.0	4.0
galwc01-2-G	5	10	5	15	6	70	50	1.8	26.1	86.6	64737	8.1	0.9	0.4	5.5	6.2	8242	2.5	5.8	41.5
galwc01-3-G	10	5	5	15	4	70	50	6.7	10.0	9.7	21503	6.1	0.0	0.4	2.1	2.9	3097	0.7	5.4	2.6
galwc01-4-G	5	5	10	15	4	70	50	0.7	14.2	9.2	31503	3.8	0.0	0.5	2.3	3.0	4284	0.5	3.3	2.2
average								2.3	15.0	27.3	35417	6.5	0.2	0.5	3.0	3.7	4558	1.3	5.3	10.6
galwc02-0-G	5	5	5	15	4	70	50	1.7	3.2	4.2	14066	8.5	0.0	0.3	1.5	2.1	2427	2.0	6.6	2.2
galwc02-1-G	5	5	5	20	4	70	50	1.4	9.6	13.1	33616	15.0	0.0	0.6	1.7	2.6	3243	1.0	14.1	2.7
galwc02-2-G	5	10	5	15	6	70	50	2.1	8.1	32.1	39438	8.3	0.9	0.3	3.2	3.8	4919	4.6	3.9	9.7
galwc02-3-G	10	5	5	15	4	70	50	1.8	3.3	5.2	14066	5.8	0.0	0.3	1.3	1.9	2303	1.3	4.6	1.6
galwc02-4-G	5	5	10	15	4	70	50	1.7	5.4	9.9	23190	6.4	0.0	0.3	1.9	2.5	3392	0.0	6.4	2.4
average								1.7	5.9	12.9	24875	8.8	0.2	0.4	1.9	2.6	3257	1.8	7.1	3.7
galwc03-0-G	5	5	5	15	4	150	65	1.6	6.9	3.6	18761	2.1	0.0	0.5	1.1	1.8	2124	0.0	2.1	1.4
galwc03-1-G	5	5	5	20	4	150	65	NS	54.2	NS	68987	$-\infty$	0.0	1.1	1.9	3.3	3388	0.0	$-\infty$	2.7
galwc03-2-G	5	10	5	15	6	150	65	2.6	25.1	56.2	79140	10.4	1.6	0.6	6.0	6.9	8834	6.1	4.6	17.2
galwc03-3-G	10	5	5	15	4	150	65	0.4	7.0	3.3	18761	2.6	0.0	0.5	1.1	2.0	2085	0.0	2.6	1.5
galwc03-4-G	5	5	10	15	4	150	65	0.5	11.8	5.2	31590	2.9	0.0	0.5	1.5	2.3	3111	0.5	2.4	1.9
average								*1.3*	21.0	*17.1*	43448	*4.5*	0.3	0.6	2.3	3.2	3908	1.3	*2.9*	4.9

TABLE 2.5: Numerical results on small instances of type G.

to the route generation phase, since the gap of GALW best solution to the one of Y ($\%^Y$) is less than 2% in most of the instances of the table. We can also notice that the H|H cost structure is often the most difficult to solve, with gaps to optimality that can vary by a factor 4 compared to other types of costs. Now, if we look at the impact of dimensional features in Table 2.3, we remark that the number $|U|$ of potential UDCs to open seems to have a significant impact on the difficulty to solve the problem: in scenario 2 instances (i.e. those in which only $|U|$ has been increased), either the gap to optimality of GALW or its computational time are much higher with respect to instances with the same cost structure from other scenarios. Scenario 2 instances also exhibit the highest computational time of the exact method X (from 5 to 20 times longer). This is explained by the fact that for both X and GALW, the number of routes generated directly depends on the number of UDCs. This is also the case for the number of demands $|D| = |P|$, which increases the number of routes generated. It is consistent to note that the number of gates $|K|$ has a low impact as it only plays a role on the first-level flows optimization but not on the route generation. On the

other hand, an increased number of SPLs $|L|$ has an impact on the number of paths but not on the performances of the methods, as SPLs are only involved by routing and fleet rebalancing aspects without any implication related to capacity or load issues. Therefore, the dimensional features that directly impact the route generation and the capacity/load issues are more critical than the others with respect to the tractability of the methods. Finally, we can observe that for these small-sized instances, the hybrid method performs better on average than GALW as it obtains slightly better gaps with reduced computational time. Nevertheless, the GALW heuristic remains robust to find acceptable solutions in a short time.

On small-sized ecological instances (table 2.5), both GALW and Y exhibit a worse behavior w.r.t. the economic instances. The final gap of both GALW and Y to the optimal solution ($\%^X$) is higher, and so it is for the gap of the former w.r.t. the latter. This is certainly due to the particular cost structure of green instances: there are no strategic costs, and even if the ratio between the flow transportation costs on the ring and those between gates and UDCs is similar to the same ratio in L|L instances, the relative weight of second level routing is from 1.3 to 5.6 times lower. This suggests that X makes use of longer routes than those produced by the route generation step in order to accomplish much lower flow transportation costs at first level. Both Y and GALW pay the price of having only shorter paths, the latter being the most penalized since it is unaware of the ring. However, ecological instances have drawbacks for the X method too. The absence of major, strategic cost makes the root gap higher, and even if the solving times are generally reasonable, solving a small-sized green instance within a short time limit may become hard for X. This is the case for instance galwc03-1-G, i.e. the one with the higher number of customers and thus the strongest combinatorics in terms of demand sequences. This makes sense, since the reduced routing costs force X to consider a wider range of routes, whereas Y and GALW only have shorter ones. Therefore, the performances of both GALW and Y remain valuable, both being able of always yielding a solution with overall reasonable gap and with a computation time which is even shorter w.r.t. the economic instances.

	Features							GALW						Hybrid(Y)
Instance	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%^Y$	T
galwc06-0-L|L	5	5	5	25	5	70	60	0.0	1.7	2.0	4.0	3180	3.0	3.9
galwc06-1-L|L	5	5	5	40	5	70	60	0.0	4.0	5.7	10.0	7376	2.0	8.3
galwc06-2-L|L	5	10	5	25	10	70	60	1.1	1.4	11.0	12.7	9428	1.4	33.2
galwc06-3-L|L	10	5	5	25	5	70	60	0.0	1.7	2.4	4.4	3155	1.1	4.1
galwc06-4-L|L	5	5	10	25	5	70	60	0.0	1.5	2.8	4.7	4448	0.6	4.2
average								0.2	2.1	4.8	7.2	5517	1.6	10.7
galwc07-0-L|L	5	5	5	25	5	150	65	0.0	2.6	6.0	8.9	4791	0.7	7.4
galwc07-1-L|L	5	5	5	40	5	150	65	0.0	7.7	18.2	26.2	10902	0.3	23.1
galwc07-2-L|L	5	10	5	25	10	150	65	1.2	2.6	22.7	25.7	13286	3.0	31.1
galwc07-3-L|L	10	5	5	25	5	150	65	0.0	2.6	5.8	8.7	4752	0.0	8.9
galwc07-4-L|L	5	5	10	25	5	150	65	0.0	2.6	7.7	10.6	6936	0.0	9.0
average								0.2	3.6	12.1	16.0	8133	0.8	15.9

TABLE 2.6: Numerical results of GALW and the hybrid method on medium size instances of type L|L.

Instance	Features							GALW						Hybrid(Y)
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%Y$	T
galwc06-0-L|L	5	5	5	25	5	70	60	0.0	1.7	2.0	4.0	3180	3.0	3.9
galwc06-0-L|H	5	5	5	25	5	70	60	0.0	1.6	2.4	4.3	3107	8.7	5.5
galwc06-0-H|L	5	5	5	25	5	70	60	0.0	1.7	3.7	5.7	3135	0.1	4.0
galwc06-0-H|H	5	5	5	25	5	70	60	0.0	1.6	3.5	5.5	3092	2.5	5.5
average								0.0	1.6	2.9	4.9	3128	3.6	4.7
galwc07-0-L|L	5	5	5	25	5	150	65	0.0	2.6	6.0	8.9	4791	0.7	7.4
galwc07-0-L|H	5	5	5	25	5	150	65	0.0	2.7	7.3	10.3	4880	0.1	7.1
galwc07-0-H|L	5	5	5	25	5	150	65	0.0	2.6	8.1	11.0	4813	0.2	7.8
galwc07-0-H|H	5	5	5	25	5	150	65	0.0	2.6	8.0	10.9	4839	0.1	9.6
average								0.0	2.6	7.3	10.3	4831	0.3	8.0

TABLE 2.7: Numerical results of GALW and the hybrid method on all scenario 0 instances of medium-sized collections.

Instance	Features							GALW						Hybrid(Y)
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%Y$	T
galwc06-0-G	5	5	5	25	4	70	60	0.0	1.7	2.7	4.7	3148	1.4	5.4
galwc06-1-G	5	5	5	40	4	70	60	0.0	4.1	7.6	12.0	7537	1.5	14.3
galwc06-2-G	5	10	5	25	6	70	60	1.5	1.4	8.5	10.2	9396	4.2	22.2
galwc06-3-G	10	5	5	25	4	70	60	0.0	1.6	1.8	3.7	3116	1.3	3.7
galwc06-4-G	5	5	10	25	4	70	60	0.0	1.5	2.7	4.4	4410	1.6	4.3
average								0.3	2.1	4.6	7.0	5521	2.0	10.0
galwc07-0-G	5	5	5	25	4	150	65	0.0	2.6	3.9	6.8	4714	1.3	7.3
galwc07-1-G	5	5	5	40	4	150	65	0.0	7.8	12.2	20.3	10733	0.9	19.2
galwc07-2-G	5	10	5	25	6	150	65	1.1	2.5	12.1	14.9	13666	1.5	38.8
galwc07-3-G	10	5	5	25	4	150	65	0.0	2.6	5.5	8.4	4731	0.0	10.3
galwc07-4-G	5	5	10	25	4	150	65	0.0	2.5	5.6	8.4	6975	0.5	7.5
average								0.2	3.6	7.9	11.7	8164	0.9	16.6

TABLE 2.8: Numerical results of GALW and the hybrid method on medium size instances of type G.

▶ On *medium-sized* instances of Tables 2.6, 2.7 and 2.8, we did not report results of the exact method X. Indeed, the results of a preliminary analysis, which are illustrated by Table 2.9, suggested that when dealing with such instances, CPLEX can possibly terminate the search without finding a feasible solution for X. This analysis was conducted on a small sample of instances mainly taken from scenarios 0 and 2 of medium- and large-sized collections – except for instance galwc06-1-L|L. Only L|L instances were considered in order to simplify the computation; yet we wanted to test the X method for larger values of $|U|$. We could only compute the X value for just a few of the instances of the sample. Indeed, due to the very large number of routes generated by the exact method (hundreds of thousands of routes for medium-sized instances, a few millions for large-sized instances), CPLEX was not even able to determine the value of the root LP relaxation within the time limit for some of the instances (i.e. those marked with *NS* in the table).

On the other hand, it can be observed that the gap of GALW vs X remains acceptable for these larger instances where it can be computed (between 1% and 5% for all instances but one). So one can reasonably expect a gap of the same order for the large-sized instances where X gives no solution. For medium-sized instances of Tables 2.6, 2.7 and 2.8, as we just explained, we could only compare GALW with the hybrid method. We observe that Y is still competitive, but now the gap between GALW and Y is less than 3% for most of the instances in all the tables. Note that even if Y never

Instance	Features							Exact(X)				GALW							Hybrid(Y)	
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	$\%^r$	r	T	#ps	$\%^X$	t	r	a	T	#ps	$\%^Y$	$\%^X$	$T/\%^z$
galwc06-1-L|L	5	5	5	40	5	70	60	0,1	75,8	252,5	219879	3,8	0,0	4,0	5,7	10,0	7376	2,0	1,8	8,3
galwc06-2-L|L	5	10	5	25	10	70	60	6,7	21,4	216,4	100049	3,8	1,1	1,4	11,0	12,7	9428	1,4	2,4	33,2
galwc07-0-L|L	5	5	5	25	5	150	65	*NS*	434,6	*NS*	457812	$-\infty$	0,0	2,6	6,0	8,9	4791	0,7	$-\infty$	7,4
galwc07-2-L|L	5	10	5	25	10	150	65	*NS*	1833,4	*NS*	1566535	$-\infty$	1,2	2,6	22,7	25,7	13286	3,0	$-\infty$	31,1
galwc08-0-L|L	5	5	10	50	5	70	50	2,2	30,3	430,1	111289	4,9	0,0	9,2	11,2	20,7	6350	0,5	4,4	20,1
galwc08-2-L|L	5	10	10	50	10	70	50	2,7	102,3	2864,2	387820	3,4	1,6	7,3	56,5	64,2	15019	0,0	3,4	69,8
galwc10-0-L|L	5	5	10	50	5	70	50	2,1	163,5	4979,7	352647	1,7	0,0	7,7	17,1	25,2	10814	0,0	1,7	18,0
galwc10-2-L|L	5	10	10	50	10	70	50	*NS*	8537,6	*NS*	885801	$-\infty$	1,4	6,8	242,9	250,1	26212	0,2	$-\infty$	281,1
galwc11-0-L|L	5	10	10	50	10	70	50	2,9	83,5	6837,2	295662	10,4	1,4	7,9	466,4	474,8	14835	2,0	8,5	2681,4
galwc11-2-L|L	5	15	10	50	15	70	50	*NS*	208,6	*NS*	535059	$-\infty$	69,7	8,1	612,8	621,3	27451	-18,6	$-\infty$	(28,9%)

TABLE 2.9: Numerical results on a small sample of medium and big size instances.

requires more than 40s to end, GALW is much faster. Here again, in Tables 2.6 and 2.8, we observe that the critical dimensional parameters are the numbers of UDCs and demands. This holds for both the L|L and the green instances. The number of routes generated by GALW is twice to three times the one for small-sized instances, yet the computational times remain low enough.

Instance	Features							GALW						Hybrid(Y)
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%^Y$	$T/\%^z$
galwc08-0-L|L	5	5	10	50	5	70	50	0.0	9.2	11.2	20.7	6350	0.5	20.1
galwc08-1-L|L	5	5	10	80	5	70	50	0.0	28.1	31.1	59.7	16593	0.7	271.0
galwc08-2-L|L	5	10	10	50	10	70	50	1.6	7.3	56.5	64.2	15019	0.0	69.8
galwc08-3-L|L	10	5	10	50	5	70	50	0.0	9.6	11.6	21.6	6412	0.1	16.9
galwc08-4-L|L	5	5	15	50	5	70	50	0.0	9.0	13.2	22.7	9334	0.5	21.0
average								0.3	12.6	24.7	37.8	10742	0.4	79.8
galwc10-0-L|L	5	5	10	50	5	70	50	0.0	7.7	17.1	25.2	10814	0.0	18.0
galwc10-1-L|L	5	5	10	80	5	70	50	0.0	24.0	106.7	131.2	28020	0.6	768.3
galwc10-2-L|L	5	10	10	50	10	70	50	1.4	6.8	242.9	250.1	26212	0.2	281.1
galwc10-3-L|L	10	5	10	50	5	70	50	0.0	8.2	31.0	39.7	10912	0.2	55.7
galwc10-4-L|L	5	5	15	50	5	70	50	0.0	7.8	19.5	27.7	13963	0.1	23.8
average								0.3	10.9	83.4	94.8	17984	0.2	229.4
galwc11-0-L|L	5	10	10	50	10	70	50	1.4	7.9	466.4	474.8	14835	2.0	2681.4
galwc11-1-L|L	5	10	10	80	10	70	50	1.1	21.6	476.7	499.0	36996	$-\infty$	$(+\infty)$
galwc11-2-L|L	5	15	10	50	15	70	50	69.7	8.1	612.8	621.3	27451	-18.6	(28.9%)
galwc11-3-L|L	10	10	10	50	10	70	50	1.1	8.3	834.0	842.8	14691	-0.1	(2.0%)
galwc11-4-L|L	5	10	15	50	10	70	50	1.2	7.6	391.6	399.7	19571	5.8	879.2
average								14.9	10.7	556.3	567.5	22709	-2.7	2690.2

TABLE 2.10: Numerical results of GALW and the hybrid method on big instances of type L|L.

Instance	Features							GALW						Hybrid(Y)
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%^Y$	$T/\%^z$
galwc08-0-L|L	5	5	10	50	5	70	50	0.0	9.2	11.2	20.7	6350	0.5	20.1
galwc08-0-L|H	5	5	10	50	5	70	50	0.0	9.4	6.3	16.1	6424	1.3	19.6
galwc08-0-H|L	5	5	10	50	5	70	50	0.0	9.7	8.9	19.0	6386	6.1	15.2
galwc08-0-H|H	5	5	10	50	5	70	50	0.0	9.5	8.1	18.0	6373	1.8	20.8
average								0.0	9.4	8.6	18.4	6383	2.4	18.9
galwc10-0-L|L	5	5	10	50	5	70	50	0.0	7.7	17.1	25.2	10814	0.0	18.0
galwc10-0-L|H	5	5	10	50	5	70	50	0.0	7.8	16.0	24.1	10647	3.2	23.3
galwc10-0-H|L	5	5	10	50	5	70	50	0.0	7.7	26.5	34.6	10820	0.5	42.2
galwc10-0-H|H	5	5	10	50	5	70	50	0.0	7.9	15.2	23.5	11015	0.9	39.0
average								0.0	7.8	18.7	26.9	10824	1.2	30.6
galwc11-0-L|L	5	10	10	50	10	70	50	1.4	7.9	466.4	474.8	14835	2.0	2681.4
galwc11-0-L|H	5	10	10	50	10	70	50	1.8	8.0	514.2	522.7	14663	3.1	(0.6%)
galwc11-0-H|L	5	10	10	50	10	70	50	1.1	8.1	414.7	423.1	14671	2.6	3211.2
galwc11-0-H|H	5	10	10	50	10	70	50	1.1	8.0	548.9	557.3	14532	2.6	1628.3
average								1.4	8.0	486.1	494.5	14675	2.6	2780.2

TABLE 2.11: Numerical results of GALW and the hybrid method on all the instances in scenario 0 of big-sized collections.

Instance	Features							GALW						Hybrid(Y)
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%Y$	$T/\%^z$
galwc08-0-G	5	5	10	50	4	70	50	0.0	9.4	5.8	15.6	6243	2.3	16.6
galwc08-1-G	5	5	10	80	4	70	50	0.0	27.3	22.1	49.9	16622	1.7	100.6
galwc08-2-G	5	10	10	50	6	70	50	1.0	7.0	24.6	32.0	14872	3.8	203.8
galwc08-3-G	10	5	10	50	4	70	50	0.0	9.4	6.8	16.5	6333	2.1	19.6
galwc08-4-G	5	5	15	50	4	70	50	0.0	9.1	10.0	19.5	9210	1.5	27.1
average								0.2	12.4	13.8	26.7	10656	2.3	73.5
galwc10-0-G	5	5	10	50	4	70	50	0.0	7.8	14.9	23.1	10807	1.4	36.9
galwc10-1-G	5	5	10	80	4	70	50	0.0	23.6	56.5	80.6	27978	2.8	115.7
galwc10-2-G	5	10	10	50	6	70	50	0.9	6.8	41.2	48.3	26042	1.5	340.4
galwc10-3-G	10	5	10	50	4	70	50	0.0	7.9	12.7	20.9	10645	0.6	24.9
galwc10-4-G	5	5	15	50	4	70	50	0.0	7.8	16.9	25.0	13290	2.0	36.6
average								0.2	10.8	28.4	39.6	17752	1.7	110.9
galwc11-0-G	5	10	10	50	6	70	50	1.4	7.9	90.3	98.7	14641	2.6	423.2
galwc11-1-G	5	10	10	80	6	70	50	0.9	22.4	66.5	89.4	37138	3.4	2983.0
galwc11-2-G	5	15	10	50	10	70	50	62.2	8.2	48.4	57.1	28059	3.0	(4.2%)
galwc11-3-G	10	10	10	50	6	70	50	0.9	7.8	52.1	60.4	14573	1.6	737.1
galwc11-4-G	5	10	15	50	6	70	50	1.7	7.8	72.2	80.5	19628	3.6	2058.4
average								13.4	10.8	65.9	77.2	22808	2.8	1960.4

TABLE 2.12: Numerical results of GALW and the hybrid method on big instances of type G.

▶ On *large-sized* instances of Tables 2.10, 2.11 and 2.12, we reach the limits of the hybrid method Y. In most of the cases GALW finds solutions that have a gap between 1% and 3% to the hybrid method, but in comparable or much shorter computational times. For the largest instances, GALW finds solutions that are much better than Y which reaches its time limit, or cannot even find a feasible solution. However computational times of GALW are higher for the largest instances as the number of routes generated can exceed 30000 for some of them. A comparison of tables 2.10 and 2.12 shows once more the impact of the ecologic cost structure: GALW is faster on green instances, but its gap to Y is higher, due to the minor relative weight of routing costs. Hence, the assignment subproblem is somehow simpler, but this leads GALW to a worse global solution. However, even with green instances, for which the impact of the decomposition in stages is more dramatic, the gap of GALW to Y is always under 4%.

The above analysis on various sizes of instances consistently shows that the GALW heuristic is a good compromise to find solutions with both reasonable quality and computational time, compared to an enumerative method. The number of routes generated can also be limited by an upper bound in order to control more efficiently computational time for the largest instances. To finish the analysis, figure 2.4 shows how fastly the number of route variables grows in function of M in the general MILP model, taking two medium-sized instances as examples. Not surprisingly, the higher the length limit on routes, the higher the number of ways to combine clients to make a route with a very high growth rate of the number of paths in the figure.

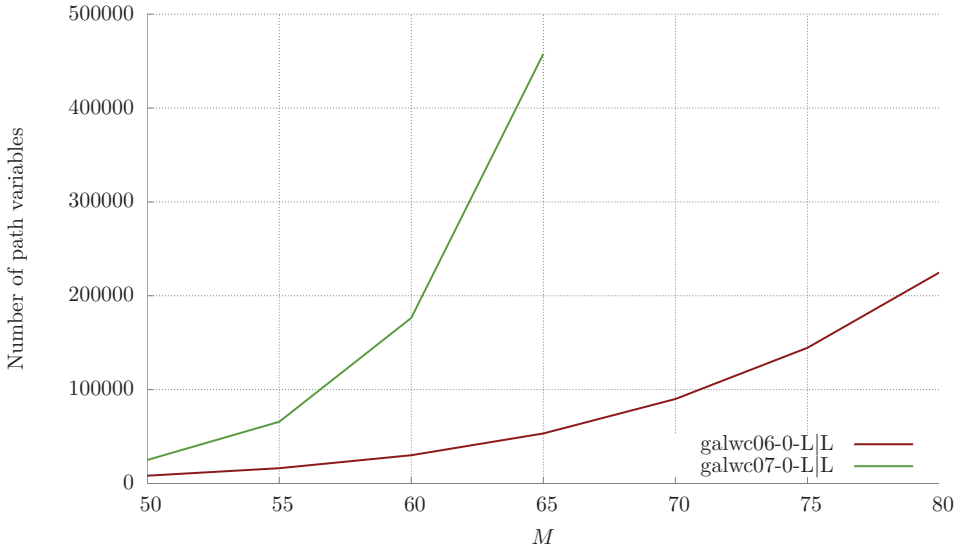


FIGURE 2.4: The relation between the number of routes and the maximum trip length M.

2.8 Conclusions

In this chapter, a new strategic location NP-hard problem, the Multicommodity-Ring Location Routing Problem (MRLRP), has been presented. The aim of this problem is to approach an issue as challenging as the last mile problem and to study from a strategic perspective a distribution system based on a ring of Urban Distribution Center for cross-docking operations. The MRLRP is a rich and complex problem as it involves a number of different, strongly interconnected decision layers (location, network design, vehicle routing, flow transportation). We propose a MIP Set-Partitioning-like formulation that could be solved efficiently by a Branch&Bound exact method for small-sized instances only. Instead of an implicit enumeration method such as column generation, given the high difficulty of the problem, we designed GALW, a four-stage decomposition matheuristic. GALW tackles the different decisional components of the MRLRP sequentially and solves most of them to optimality. We also developed a hybrid method which consists of running the exact method on the subset of routes generated by GALW. A wide set of 200 MRLRP instances with various size parameters and cost structures has been designed in order to outline an exhaustive computational experience: among these, a *green* cost structure with no strategic cost and routing and transportation costs expressed with pollution indicators has allowed to use the three methods to look for the system configuration with the highest degree of environmental sustainability.

Both the exact and the hybrid methods have been used as landmarks to assess the quality of the solutions provided by GALW: the conclusions hold regardless of whether the cost criteria are economic or ecological. On small-sized instances, both GALW and

the hybrid method have relatively low gaps compared to the exact method, with short computation times. The hybrid method is slightly better than GALW for medium-size instances, for which the exact method starts to fail finding feasible solutions. On large-sized instances, GALW clearly offers the best overall compromise between solution quality and computational time.

In a City Logistics context as the one we have been dealing with, a number of decision-making problems, ranging from the strategic level (as is the case of the MRLRP) to more tactical problems and operational, day-by-day planning, can arise. An example is given by the *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)*, a more tactical problem that arises in City Logistics as well as in long distance transport. In a tactical context, we consider the number and the size of the UDCs as given, while we would like to have a feedback concerning the behavior of electrical vehicles. Therefore, we consider the *multi-trip* feature, i.e. the possibility for a vehicle to be reloaded at a facility in terms of both electric power and load. This feature arise frequently in City Logistics, where routes typically are much shorter than the time horizon, e.g. the workday. The multi-trip feature characterizes the VRPIRF, that represents the core of Part II.

Part II

The Vehicle Routing Problem with Intermediate Replenishment Facilities

Chapter 3

The Vehicle Routing Problem with Intermediate Replenishment Facilities

3.1 Introduction

One of the most prominent families of Vehicle Routing Problems is that of Multi-Depot VRPs. Such problems derive from one of the most natural extensions of CVRP, namely the decentralization of the fleet of vehicles. This is also the case of another important class of VRPs, the Multi-Trip VRPs, where a multiple use of vehicles is allowed. The *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)* that we are about to introduce can be considered to have elements of both classes and therefore to belong to the intersection of them. Admittedly, this is the case also for the Multi-Depot Vehicle Routing Problem with Inter-Depot Routes, of which the VRPIRF turns out to be a particular case, as we will see later in a brief literature insight devoted to all of the mentioned VRP classes.

The VRPIRF has a quite straightforward characterization. It is defined on a graph where the node set consists of a *central depot*, n *customers* and f *replenishment facilities*. The aim is to find a least cost set of routes that visits each client exactly once, the cost of a route being the sum of the costs of the visited arcs. Each client has a *demand* and can be served by one of the homogeneous, fixed capacity *vehicles* based at the depot. Furthermore, vehicles can recharge at replenishment facilities so as to perform not one but a sequence of routes called a *rotation*. However, the rotation of a vehicle must start and end at the depot and its total *duration* (the sum of the travel, service and recharge times associated with the visited arcs, clients, and depots, respectively) must not exceed

a given shift length. Figures 3.1 and 3.2 show an example of VRPIRF instance and a feasible solution for it.

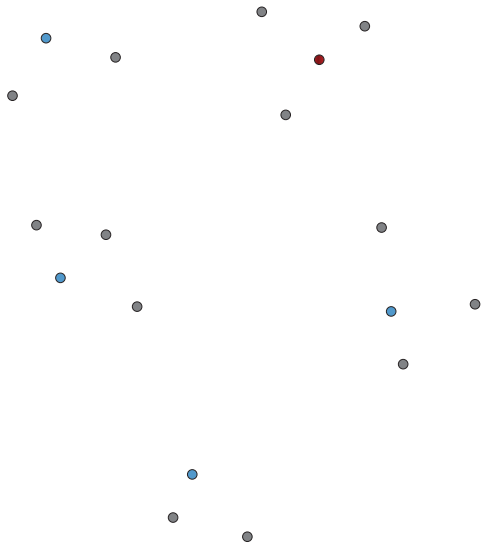


FIGURE 3.1: A VRPIRF instance: the depot (red), the facilities (blue) and the customers.

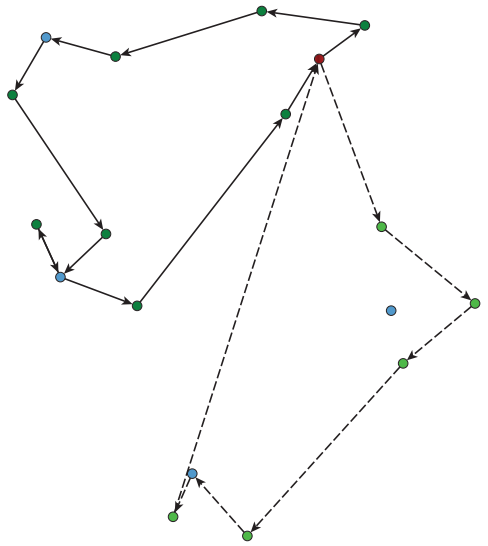


FIGURE 3.2: A solution to the previous instance, with the rotations of the two vehicles.

The remainder of Part II is structured as follows:
The extensive review of the literature proposed in section 3.2 positions the VRPIRF in the wide spectrum of the Vehicle Routing Problems, firstly investigating the problems from which VRPIRF descends, then focusing on the efforts that have been made to develop efficient approaches to VRPIRF itself and other problems which are similar or related to it. Then, Chapters 4 and 5 will present a family of exact approaches that have been designed to solve the VRPIRF: the former will introduce two algorithms of type *Branch&Cut,* while the latter will propose two algorithms of type *Branch&Price.*

3.2 Literature Review

VRPIRF is the particular case of the *Multiple Depot VRP with Inter-Depot routes (MDVRPI)* with only one depot. MDVRPI itself is a generalization of the *Multi-Depot VRP (MDVRP)* in which each depot acts both as the base for the vehicles of its own fleet, and as a facility for vehicles based at other depots. Hence, VRPIRF turns out to belong to the family of Multi-Depot VRPs, one of the most investigated families of VRPs, which is briefly dealt with in section 3.2.1. The multiple use of vehicles is an element that VRPRIF has in common with the *Multi-Trip VRP (MTVRP)*: section 3.2.2 covers the main reference works on the subject. Lastly, section 3.2.3 delves into the works in the literature that concern the VRPIRF itself and the VRP variants which are closer to it.

3.2.1 Multi-Depot VRP

The statement of MDVRP does not differ substantially from that of classical CVRP. In the latter, a central depot with a fleet of homogeneous vehicles of finite capacity is given, with the aim of determining a least cost set of routes that visit and deliver a set of customers, each being characterized by a demand. The difference is that in the former we have a set of depots, each one with its own fleet, and the decision of how to partition the customers over the depots must be taken. Few exact algorithms are known for the MDVRP: in [Arpin et al., 1984], the symmetric MDVRP is solved by a Branch&Bound algorithm based on a compact formulation, solving instances with up to 50 customers and 8 depots, while [Laporte et al., 1987] tackles asymmetric instances by transforming the problem into a constrained assignment problem and then solving it by Branch& Bound: the algorithm solves to optimality cases with 80 customers and 3 depots. More recently, [Baldacci and Mingozzi, 2009] defines a unified framework for solving many different VRPs, notably several variants of the *Heterogeneous VRP (HVRP)* and the MDVRP. Following a consolidated exact methodology (used also, e.g., in [Baldacci et al., 2008] and in [Mingozzi et al., 2013]), the authors present an exact solving approach based on a Set Partitioning formulation and a series of bounding procedures used in sequence in an additive fashion, in order to raise the dual bound. The problems are then solved as MIP by using only the columns whose reduced cost is less than the optimality gap, which can be exhaustively generated as the additive bounding nearly closes the gap.
Most of the approaches to MDVRP are heuristic. Among the first approaches we cite the constructive algorithm of [Tillman and Cain, 1972], based on a generalized version of the Clarke and Wright savings criterion that takes into account the multiple depots; that of [Wren and Holliday, 1972], which generates different initial solutions based on geometric

and least insertion criteria and later uses improvement procedures; and the approach of [Gillett and Johnson, 1976], where customers are initially clustered, then a feasible solution is obtained by solving CVRP subproblems, and the final phase tries to improve the solution by reassignment of customers in the regions between two depots. More recently, [Chao et al., 1993] proposes a multi-phase heuristic that first assigns customers to the closest depot and solves separate CVRPs by a modified savings-based heuristic, then swaps customers between routes in a local search phase that allows deteriorations of the solution. Diversification is provided by reinitialization procedures. Results for instances with up to 360 customer nodes and 9 depots are presented. Two of the most relevant works of the last two decades are based on *Tabu Search (TS)*, one of the most effective and used heuristic strategies to explore the solution space of a combinatorial problem. At each step, TS moves from a solution s to the best one of its neighborhood $N(s)$, allowing deteriorating moves to help moving out of local optima. Solutions are characterized by *attributes*, and the attributes of recently visited solutions are declared *tabu* and forbidden to avoid cycling. To avoid tabus being too restraining, they can be overridden according to *aspiration criteria*, e.g. whenever a tabu solution improves the incumbent. *Intensification* in promising regions or *diversification* to broaden the search to less explored ones can be used; intermediate infeasible solutions can be allowed. See [Gendreau and Potvin, 2003] for a comprehensive introduction to TS. The authors of [Renaud et al., 1996] propose a heuristic that first builds an initial solution, then improves it by means of TS. The former phase assigns each customer to its nearest depot and solves VRP subproblems via an *Improved Petal heuristic*, while the second relies on FIND, a TS-based procedure that alternates improvement, intensification and diversification steps: these latter are all based on subroutines that exchange customers belonging to the same route, or to two different routes, or three. Even in its fastest version, i.e. the one that does not apply diversification, the algorithm outperforms that of [Chao et al., 1993] in at least 19 out of 23 benchmark instances used in [Gillett and Johnson, 1976] and [Chao et al., 1993]. The work presented in [Cordeau et al., 1995] is been the most competitive for almost a decade. The authors propose a general MILP model for the *Periodic VRP (PVRP)*, proving that it is also suitable for the *Periodic TSP (PTSP)* and the MDVRP. Then they present a unified TS-based algorithm that solves the general problem. The method makes use of *GENI*, a 4-opt-based tour construction heuristic, as a base for every insertion of a client in a route, or removal. To build an initial solution, customers are assigned to the nearest depot and then progressively inserted with GENI into routes according to a geometric or an arbitrary order. In the TS phase, infeasible solutions are allowed and evaluated by adding penalty terms in the objective function with parametric weights that can be changed to diversify the search, and GENI is used again to perform customer relocations. An extensive sensitivity analysis is conducted to conveniently tune the algorithm, which outperforms the

TS heuristic of [Renaud et al., 1996] in 17 out of 23 of the aforementioned benchmark instances. Besides, 10 new benchmark instances are defined. Later, the authors of [Pisinger and Røpke, 2007] proposed a heuristic method to solve the *Rich Pickup and delivery Problem with Time Windows (RPDPTW)*. They show how five different VRPs, among which are the CVRP, the *VRP with Time Windows (VRPTW)* and the MDVRP, can be transformed into the RPDPTW, hence yielding a unified method. The proposed algorithm is based on *Adaptive Large Neighborhood Search (ALNS)*, a technique first introduced in [Ropke and Pisinger, 2006]) that uses a set of *large neighborhood* operators, i.e. such that a move from a solution to a neighbor one can impact many problem variables at a time. Large neighborhoods are divided into *destroy* and *repair* ones: they are used alternatively to alter and fix the current solution, and chosen according to their previously achieved performances. ALNS can use any local search tool, like simulated annealing, TS or the introduction of noise in the objective function as diversification tool. The ALNS heuristic of [Pisinger and Røpke, 2007] is tested on the 33 previously used instances, obtaining 15 new best solutions and 16 ties over the remaining instances. The last work that we mention is the *Hybrid Genetic Algorithm* of [Vidal et al., 2012]. Genetic Algorithms consider the solutions of a problem to be individuals of a population. Hence, the search in the solution space is an evolutionary-based search driven by operators typically performing parent selection, crossover, generation of new individuals, selection and education of individuals, population management. We refer the reader to [Holland, 1975] for the foundations of this algorithmic paradigm. The authors of [Vidal et al., 2012] outline a unified approach for the PVRP, the MDVRP and the *Multi-Depot Periodic VRP (MDPVRP)*, while proposing new contributions as to the used evolutionary operators. On the MDVRP, the method proved very effective, improving or tying the best known solutions delivered by the methods of [Cordeau et al., 1995] or [Pisinger and Røpke, 2007] on the 33 benchmark instances with comparable execution times.

3.2.2 Multi-Trip VRP

The *Multi-Trip VRP (MTVRP)* is a tactical problem that extends the CVRP in that vehicles are allowed to perform more than one service route, i.e. to be refilled each time they come back to the depot and to start another route, under a *maximum shift length* or *maximum duration* constraint over the set of routes that each vehicle performs. This extension arises naturally when considering situations, like e.g. urban contexts, where the available fleet to deliver or collect goods is made up of vehicles with tight limitation in terms of size, load and/or autonomy, thus reducing the number of points (most often *customers*) that can be visited within one same route. As a consequence we have trips

much shorter than the time horizon and the requested size of the fleet grows considerably, unless multiple trips are allowed for each vehicle. Of course, this calls for the assignment of routes to vehicles, which is somehow implicit in the CVRP and becomes nontrivial in the MTVRP. The MTVRP has therefore been widely studied over the last decades. In the following, we review some of the most prominent works in the literature of the most general version of the MTVRP, before briefly mentioning some important variant.

The first work to address this problem is to the best of our knowledge [Fleischmann, 1990], which tackles the problem with the most natural decomposition approach, i.e. by first generating a set of suitable service routes with a heuristic initial step, namely a modified savings procedure, and then assigning routes to vehicles by means of a *Bin Packing Problem (BPP)*. In [Taillard et al., 1995], a heuristic algorithm for the MTVRP is proposed, which relies on the TS algorithm for the VRP presented in [Taillard, 1993]. The latter is used to generate an initial pool of VRP solutions, whose routes are inserted into a list. Routes are randomly chosen from the list according to the quality of the solutions that use them, and their usage frequency, so as to generate new VRP solutions in a second TS phase. The routes of the large set obtained so far are used to form MTVRP solutions by means of a BPP heuristic. This paper offers an important set of benchmark instances, derived from nine instances for the CVRP. For each of these latter, different values of the fleet size m are used: the time horizon T_H is given two different values derived from the solution to the original CVRP instance obtained in [Rochat and Taillard, 1995]. The overall set amounts to 104 MTVRP instances and stands as the reference instance set for the MTVRP to date. The algorithm is run five times for each instance, proving capable to obtain a feasible solution at each run in 74 cases. For the instances for which no feasible solution has been found, the authors also describe the best solution, i.e. the one with the lowest *longest route ratio (LRR)* w.r.t. to T_H.

In [Petch and Salhi, 2003], a multi-phase heuristic approach is proposed which aims at minimizing the maximum overtime. First, a pool of VRP solutions is created by repeatedly executing the parametrized savings algorithm proposed by Yellow [Yellow, 1970] and, separately, a route population approach. Then, routes are assigned to vehicles by means of a BPP heuristic. Finally, the obtained MTVRP solutions are improved using local search based, among others, on 2-opt, 3-opt, and customer reallocation moves. The same authors propose in [Salhi and Petch, 2007] an approach that combines genetic procedures, client clusterization, the Clarke and Wright savings algorithm to solve some VRP subproblems and again a BPP heuristic to pack routes.

The work [Brandão and Mercer, 1998] proposes *TSMTVRP*, a TS-based heuristic approach. An initial solution is provided by a starting heuristic that makes use of nearest neighbor and insertion procedures. Then, the tabu search, which consists of two phases applied in sequence, is applied. Phase 1 and phase 2 differ in that the former allows

intermediate infeasible solutions. Therefore the objective functions of the two phases are the same except for a penalizing term. In the search, insert and swap moves are considered: in the former, the client extracted from one route can be reallocated either in an existing route, or in another one generated by means of GENI. The computational results are compared to those of [Taillard et al., 1995], obtaining a higher rate of instances for which a feasible solution is found. On the instances where both algorithm produce an infeasible solution, those provided by TSMTVRP are on average more costly in time and routing cost, but have a lower LRR.

The heuristic approach described by the authors of [Olivera and Viera, 2007] relies on the Adaptive Memory principle of [Rochat and Taillard, 1995]. A *memory* is initialized with a set of randomly generated VRP solutions. Then, new solutions are built (*construction*) by probabilistically selecting routes from the memory, i.e. by choosing them randomly but with probability that depends on the quality of the solutions they appear in. A Tabu Search stage improves the constructed solution: intermediate infeasible solutions are allowed and evaluated by a generalized objective function that penalizes capacity and duration violations with dynamic weights. Customer swap and customer reallocation neighborhoods are considered in the search. After each TS move, a MTVRP solution is built by applying a *Best Fit Decreasing (BPF)* heuristic for the BPP and possibly corrected if the assignment of routes to vehicles violates the duration restriction. Finally, the memory is updated with the routes of the new found solutions. Results over the 104 instances of [Taillard et al., 1995] are reported, with a rate of feasible solutions found that is higher than other previous works, namely [Taillard et al., 1995], [Petch and Salhi, 2003] and [Brandão and Mercer, 1998], and a LRR on the unsolved instances which is better than the same works.

The Memetic Algorithm of [Cattaruzza et al., 2014b] represents the state of the art for what concerns heuristic algorithms for the MTVRP. A *Memetic Algorithm (MA)* is a genetic algorithm that makes use of local search techniques to improve solutions. One of the most important contributions of [Cattaruzza et al., 2014b] is exactly the definition of a problem-tailored local search operator, named *Combined Local Search (CLS)*. Its original feature is the choice to combine non-improving moves and reassignment of routes to vehicles, as this can lead to an overall improvement of the solution. *Chromosomes*, i.e. the starting elements to generate an individual of the population of solutions, are sequences of customer nodes without a start and an end point, like the *giant tours* (i.e. large tours that visits all the customers) used e.g. in [Prins, 2004]. The same paper also introduces the *Split Procedure*, which consists in optimally dividing a giant tour in feasible routes. By adapting this procedure, the algorithm in [Cattaruzza et al., 2014b] obtains MTVRP solutions from a giant tour deciding the split points and the assignment of the obtained routes to vehicles. The quality of a chromosome is denoted by the derived MTVRP solution which is the best w.r.t. a weighted function, called the

fitness, that considers the cost, the time, the load, the *overtime* and the *overload*. After the selection of *parent* solutions, new individuals are obtained by means of a *crossover* operator. The *education* is a LS phase that relies on 10 different operators and considers also non-improving moves as said before. Computational experiments are conducted on the 104 benchmark MTVRP instances subdivided in three groups: G1, i.e. the 42 solved to optimality by [Mingozzi et al., 2013] (which we will review in a short while), G2, i.e. the other 56 for which a feasible solution exists, and G3, those for which no solution exists to date. On each of the G1 instances, the algorithm is run 5 times, both with and without using the CLS. Both variants always find the optimal solution within the fifth run, with a higher rate of successful runs for the one that uses CLS. On G2 instances, the algorithm improves previously best known solutions (when existing) and outperforms [Olivera and Viera, 2007] in the GAP measure of the solution quality, which considers the solutions of the corresponding VRP instance as lower bound. Finally, a remarkable result is achieved by finding a feasible solution for one of the G3 instances.
The only two papers that tackle the MTVRP with an exact method that we are aware of are [Karaoglan and Koc, 2011] and [Mingozzi et al., 2013]. The first proposes a two-index formulation that imposes capacity and subtour elimination constraints in a Miller-Tucker-Zemlin form [Miller et al., 1960] and keeps track of the shift duration of a vehicle by means of big-M-based constraints inspired from the VRPTW. A prominent feature of the model is the use of particular arc variables to denote the return to the depot of a vehicle before starting a new route. We will use a similar technique in one of our approaches to the VRPIRF (see section 4.2.1). A Branch&Cut algorithm is defined that introduces classical inequalities from the VRP literature. The authors of [Karaoglan and Koc, 2011] build an initial solution with a heuristic that combines a *Modified Clarke and Wright Savings constructive algorithm (MCWS)* and *Simulated Annealing*, a well-known stochastic search technique. The obtained initial solution acts as initial incumbent for the search in the Branch&Bound tree. The algorithm is tested on 8 benchmark instances, on which the results are slightly better than [Olivera and Viera, 2007]; further, the optimality of three previously best known solutions is shown.
More recently, [Mingozzi et al., 2013] has proposed an exact method that relies on two Set-Partitioning-like formulations for the MTVRP. The first one uses binary variables x_r^k associated to routes $r \in \mathscr{R}$ that denote whether a vehicle k performs a route r, $\mathscr{R}$ being the set of all feasible routes. The second one uses binary variables y_s^k which equal to 1 if the *schedule* $s \in \mathscr{H}$ is assigned to vehicle k: a schedule is a feasible arrangement of a sequence of routes in a workday. The authors define a family of bounding procedures to improve the lower bounds yielded by the LP relaxations of the two formulations to obtain near-optimal dual bounds. This allows to generate restricted sets of routes or schedules that contain any optimal MTVRP solution. In the process, the bounding procedure are also enhanced by valid inequalities, either taken from the literature of

the VRP or newly defined ones, namely the *working time* inequalities. The method is tested on a subset of 52 of the 104 instances of [Taillard et al., 1995], explicitly avoiding, among others, the cases that had been proven infeasible by previous works. The method is fed with the previously best known solution as initial incumbent and is able to solve to optimality 42 out of 52 cases.
To conclude our brief review, we cite some known variants of the MTVRP:

- the *MTVRP with Time Windows (MTVRPTW)*, for which [Azi, 2010] provides a series of both heuristic and exact approaches;
- the *Heterogeneous Fleet MTVRP*, for which [Prins, 2002] presents a two phase heuristic;
- the *Site-Dependent and Periodic MTVRP*, tackled in [Alonso et al., 2008] by means of a TS algorithm;
- the *Multi-Zone MTVRP*, a variant that arises in two-echelon systems studied in [Nguyen et al., 2013];
- the *MTVRPTW with Release Dates*, studied in [Cattaruzza et al., 2014a] where vehicles must wait the *release date* of the goods to be delivered, i.e. the arrival time of goods at the starting depot of the delivery trip.

3.2.3 Vehicle Routing Problems with Facilities for Recharge and Refill

In the following, we first trace the evolution of the VRPIRF from the problems it originates from to the definition of section 3.1 and review some reference papers. Then, a survey of similar problems, mostly derived from real-life applications, is presented.

3.2.3.1 Multi-Depot VRP with Inter-Depot Routes

As previously said, the VRPIRF descends from the MDVRPI, which generalizes the MDVRP by allowing a vehicle to refill during its shift at a depot different from the one at which the vehicle itself is based. The problem was proposed in [Crevier et al., 2007], but the first work that we are aware of to have ever appeared on the subject is [Jordan, 1987]. In this work, the author introduces the problem of deciding whether a vehicle of a multi-terminal delivery system should stop at other terminals to load goods and deliver clients on its road back to the base terminal. The so-called *Multiple Terminal Backhauling Problem* aims at reducing the empty truck-miles and is tackled with both a heuristic based on a matching problem to solve large-scale instances, and a Lagrangian relaxation-based method to compute a lower bound and thus prove the effectiveness of the heuristic. A real-world case-study is presented, along with the results of the

proposed methods. Differently from [Crevier et al., 2007], the work of [Kek et al., 2008] deals with a distance-constrained version of the MDVRP and proposes a variant with a so-called *flexible assignment*: not only a vehicle can replenish, but also terminate its shift at any other depot different from the starting one, under a driving range constraint. Hence, such flexible assignment is the main difference w.r.t. the MDVRPI, though it is an essential one. MILP models are proposed for both variants and then used to implement a Branch&Bound algorithm, which is then tested on a reduced set of small-sized (up to 8 customers) instances. Another less recent work is [Bard et al., 1998], where the *VRP with Satellite Facilities (VRPSF)* is defined. The authors had previously undertaken the *Inventory Routing Problem with Satellite Facilities (IRPSF)*. The *Inventory Routing Problem (IRP)* (see the tutorial paper [Bertazzi and Speranza, 2012]) is a time-dependent version of CVRP, motivated by the so-called *Vendor-Managed Inventory (VMI)* technique, where customers (in this context more often called *retailers* or even *VMI customers*) must be delivered all along a time horizon to ensure that none of them stocks out at any time, while minimizing the overall routing costs: the IRPSF generalizes it by adding refill stations. In their previous works, the authors had implemented a decomposition approach to the IRPSF, where three heuristic algorithms were used to solve the VRPSF, i.e. the resulting routing subproblems. The VRPSF consists in finding a least cost set of feasible rotations that deliver a set of customers by means of the vehicles of an homogeneous fleet. A rotation is feasible if it is made up of a continuous sequence of routes that starts and ends at the central depot within a given shift duration, and no route exceeds the vehicle capacity. [Bard et al., 1998] proposes a two-index formulation with commodity flow elements to impose the capacity constraints, a big-M-based representation of the time resource consumption similar to those used in VRPTWs, and no vehicle index. However, a maximum number of visits to each facility is imposed, and replenishments are forbidden as long as the load on a vehicle exceeds a lower threshold. A Branch&Cut algorithm is proposed, where subtour elimination constraints and valid inequalities from the TSP (namely Lifted $\overrightarrow{D_k}$ and $\overleftarrow{D_k}$ inequalities) are considered and separated by means of tailored heuristic procedures. Results on randomly generated instances with up to 20 customers and 2 facilities are presented.

In [Crevier et al., 2007], a MIP model is proposed that makes use of route variables and enforces the connection of a *rotation* (i.e. the sequence of routes performed by a vehicle) by means of degree and subtour elimination constraints. Then a heuristic method based on such formulation is introduced. The problem is first decomposed into subproblems to obtain a pool of both single-depot and inter-depot routes that are likely to appear in a MDVRPI solution: the subproblems are an MDVRP, a VRP for each depot, and a hybrid *inter-depot VRP* subproblem for each pair of depots. The VRPs subproblems are generated assigning customers to depots according to geometric criteria,

and similarly it is done for the inter-depot VRPs. These subproblems are solved a fixed number of times with different customer domains in order to obtain different solutions. All the subproblems are solved by means of a TS inspired to that of [Cordeau et al., 1995] and to *adaptive memory programming* [Rochat and Taillard, 1995]. The pool of routes returned by the solutions of the subproblems are combined to form a MDVRPI solution by means of a set partitioning algorithm based on the MIP model. Then a post-optimization phase tries to further improve this solution. A set of 22 benchmark instances is proposed: 12 (named *a1–l1*) are newly generated ones, while the other 10 (named *a2–j2*) are adaptations of MDVRP instances from [Cordeau et al., 1995]. All the instances exhibit the feature of having all the vehicles based at a *central depot*: the other depots act as replenishment facilities only, as preliminary tests have shown that inter-depot routes are more unlikely to be used when vehicles are based in two or more depots.

The authors of [Tarantilis et al., 2008] incorporated this characteristic as a problem feature rather than an instance feature and derived the VRPIRF as we have presented it, giving it this name. They proposed a local search heuristic that combines TS and *Variable Neighborhood Search (VNS)*. VNS is one of the most known *metaheuristic* approaches: the reader can refer to [Hansen and Mladenović, 2001] for a primer. A basic VNS is based on a set of neighborhood structures N_k, $k = 1,\ldots,k_M$, which are typically nested, i.e. $N_k \subset N_{k+1}$, $k = 1,\ldots,k_M-1$. Moreover, in a basic VNS a whatsoever stopping condition (e.g. a CPU time or an iteration number limit) and an initial solution x are needed. The neighborhood index k is set to 1, then the first operation is a *shaking move*: a random solution x' is chosen from the N_1 neighborhood of x. A local search step follows that applies a local search method L, up to a local optimum x'': if x'' is better than the incumbent, set $x \leftarrow x''$, otherwise $k \leftarrow k + n_k$, with $n_k = 1$, and then restart from the shaking step. One possible local search is the *Variable Neighborhood Descent*, which makes a combined use of the k_M neighborhoods even in the local search step. The procedure is metaheuristic in the sense that it offers a framework, whereas both the local search method L and the neighborhood N_k may vary, as well as the other ingredients, like e.g. the choice of the initial neighborhood, or the value of n_k. In the case of [Tarantilis et al., 2008], the initial solution is provided by a construction heuristic, based on cost savings weighted with stochastic coefficients. First, rotations are built in order to visit all the customers with the maximum shift length as only constraints; then, facilities are inserted in order to break rotations into routes that respect the capacity bound. Finally, if the maximum duration is violated by some rotation, customers are relocated into another one or inserted into a new one. VNS is then fed with 5 neighborhoods based either on the relocation of customers between two routes, or on the exchange of routes between rotations. The local search method used by VNS is TS. The last step of the algorithm is a *Guided Local Search (GLS)*, a metaheuristic that consists of a controlled

local search with penalization of undesired features of the current solution in order to move out from local optima. The local search is based again on the aforementioned neighborhood structures. The algorithm is compared with that of [Crevier et al., 2007] on the *a1–l1* instances, outperforming it in 11 out of 12 cases. Finally, the heuristic is tested on 54 newly created instances whose features range from 50 to 175 customer nodes, 2 to 7 replenishment facilities and 2 to 8 vehicles. The instances are named according to a pattern that features the number of customers, of depots (central depot and facilities) and of vehicles, e.g. the instance *50c5d2v* has 50 customers, 4 facilities and 2 vehicles.

More recently, [Muter et al., 2014] proposed what is to the best of our knowledge the only exact approach to the MDVRPI so far. It is a sophisticated Branch&Price method based on a Set Covering formulation with a binary variable for each possible rotation. A ternary branching strategy is implemented, that branches on the number of used rotations (i.e. vehicles) first, then on the assignment of customers to depots, and finally on arcs. The authors proposed two pricing algorithms. The first one is an adaptation of the multi-ESPPRC label algorithm of [Akça, 2010], which extends the well-known algorithm for ESPPRC of [Feillet et al., 2004]. The most important contribution of this work, however, is the second one, a two-level decomposition scheme to generate rotations. In this latter, the Pricing Problem (PP) is set up as a problem with route variables based on the MIP formulation proposed in [Crevier et al., 2007]. Then, due to the huge number of route variables, a second formulation for the PP is given which relies on a *route graph*, i.e. a graph with a node for each possible route. To generate the nodes and arcs of this graph, a *secondary pricing problem* is formulated as an ESPPRC on the original graph. Further, a third approach to the *main* PP is given as a minimum circulation problem with additional constraints on the *depot graph*, a dense multigraph with a node for each depot and an arc for each route connecting two depots. To avoid solving the primary PP with a secondary Branch&Price, a two-phase approach is proposed. In the first phase, a revisited ESPPRC on the original graph is solved for each pair p and $p' \leq p$ of depots. All the nondominated routes generated in the first phase are used to construct the depot graph, so as to solve the primary PP as an ESPPRC or an SPPRC. Preliminary tests show that the depot graph leads to better results, therefore this approach is further improved, whereas the route graph approach is disregarded. The improved two-phase depot graph-based algorithm outperforms the one-level inspired by [Akça, 2010] and is therefore implemented as primary PP. The overall Branch&Price framework has been tested with a reduced version of the a1-k1 and a2-j2 instances, with the first 25 or 40 customers only, with a time limit of 10 hours. Most of the instances have been solved to optimality. The same instances have been transformed into *Multi-Depot Multi-Trip VRP (MDMTVRP)* instances by forbidding inter-depot routes and thus test the effectiveness of these latter.

3.2.3.2 Similar problems

The number of practical routing applications in real-life industrial requiring the vehicles to perform intermediate stops is rapidly growing. Intermediate stops are required essentially to allow a vehicle to:

- replenish with goods to be delivered, or unload collected goods or waste;
- refuel, or recharge in the case of *battery electric vehicles (BEV)*

Therefore, a number of recent works is devoted to study new variants of VRP where such features are consistently taken into account. Most of these works tackle the respective problems proposing heuristic or metaheuristic approaches. Indeed, we are not aware of other exact methods than the previously reviewed [Kek et al., 2008], [Bard et al., 1998] and [Muter et al., 2014].

In [Prescott-Gagnon et al., 2010], a problem called the *Oil delivery VRP (ODVRP)* is studied. It is inspired from a real-life application concerning a propane supplier which must deliver both VMI (inventory) retailers and *spot* customers, which can at any time make a request to be fulfilled within 48 hours. The real-life instances of this hybrid inventory problem are very huge (up to 20 vehicles, 30000 customers, 50 visited customers per day per vehicle and visit frequencies that can reach one visit every 40 days in the winter), hence the real problem is intractable. Forecasting systems allow to estimate which VMI customers have their stock level close to a lower safety threshold and therefore to approximate the problem to a daily basis, with *mandatory* customers (VMI and spot customers to be visited the next day) and *optional* customers. With the latter is associated a coefficient that takes into account the profitability to serve them in advance. The ODVRP is the one-day problem of finding a set of routes to visit mandatory and optional customers so as to minimize the total traveled distance minus the distance savings to serve optional customers. Several other features, among which time windows, heterogeneous vehicles and the possibility of vehicle refills, are taken into account. The authors propose a TS heuristic and two *Large Neighborhood Search (LNS)* heuristic that use, respectively, the TS and another CG-based local search tool to explore the neighborhoods. Instances from the historical database of a partner real-life oil distributor were created with more than 4000 customers. One-day and one-week problems were tackled, the latter in a rolling horizon fashion.

Both [Kim et al., 2006] and [Benjamin and Beasley, 2010] tackle the *Waste-Collection VRP with Time Windows*, a variant of the VPRTW that arises in the collection of commercial waste. While collecting waste at commercial shops, the vehicles (which are homogeneous and their number is unlimited) can stop at given dump facilities for waste disposal. Time windows are associated with commercial shops for the collection, with

facilities for the disposal and with the vehicles for the ready time and the due return time at the depot. Moreover, the driver's shift has a limit w.r.t. the total amount of work and must comprise a one-hour lunch break within a two-hours period. The first work aims at minimizing the number of used vehicles and the total travel time, and at optimizing the compactness of the routes and the balancing of the workload among the vehicles. Since the datasets come from a real-life application context, two heuristic approaches are proposed. The first one relies on a modified version of the Solomon's insertion algorithm for the VRPTW [Solomon, 1987], whereas the second one is a clustering-based algorithm obtained from a variant of the K-means algorithm commonly used in data-mining. A set of benchmark instances for the Waste-Collection VRP with Time Windows is also proposed. The second work aims at minimizing the total distance traveled and proposes a construction algorithm to generate an initial solution, which is then improved by two local search procedures or one among three metaheuristics based on TS and VNS. The methods are tested on the instances proposed in [Kim et al., 2006] and outperforms the two heuristics presented there.

In [Erdoğan and Miller-Hooks, 2012], a problem named *Green VRP (G-VRP)* problem is proposed. The work is motivated by the growing interest in the so-called *Alternative Fuel Vehicles (AFVs)*, both for obvious environmental purposes, and for economic reasons, the use of such vehicles being more and more encouraged by regulations and tax incentives. Challenges exist about the deployment of fleets of AFVs, due to both to the lack of *Alternative Fuel Stations (AFSs)* and to the limited driving range of such vehicles. Therefore, the authors develop some methods with the aim to deal with real-world problems. A MILP model is presented based on an undirected graph whose node set comprises a node for the depot, one for each customer, and one for each possible visit to one of the AFSs, as in [Bard et al., 1998]. With each edge are associated a distance, a travel cost and a travel time. The depot can act as a refueling point. The fleet is limited to m homogeneous vehicles with a tank of limited capacity, which is completely filled at every stop to a AFS or the depot. The vehicles are imposed a maximum shift duration. In spite of the similarities with some previous works, [Erdoğan and Miller-Hooks, 2012] is perhaps among the first to deal with both the possibility of a vehicle to perform multiple trips and to extend its driving range. The authors propose two heuristic approaches. The first is a MCWS that starts from one-customer tours with the insertion of at most one AFS visit and progressively merge them according to a set of savings-driven rules. The solution obtained may not meet the maximum number m of allowed vehicles. The second is a *Density-Based Clustering Algorithm (DBCA)*, i.e. a cluster-first, route-second heuristic. First, a clustering of the nodes of an instance is yielded for each value of two parameters of a geometric neighborhood measuring its radius and density; then, for each of such clusterings, the MCWS heuristic is applied. The best overall solution is saved. Again, the solution obtained this way may require a

number $m' > m$ of vehicles. However, the authors claim that such a possibility can still help the decision-maker to consider the possibility of a conversion to a fleet of m' AFVs. In both cases, a post-optimization phase based on customer relocation techniques allows to further improve the produced solution. Four sets of 10 small-sized instances each were randomly generated in order to be solved by both CPLEX and the proposed heuristics to assess the performances of the latter. Finally, 12 instances based on real-life scenarios were designed, five of which differ in the number of AFSs, and the other 7 in the number of customers, the biggest one counting 492 customers. On real-life instances, the MCWS heuristic has been run with and without the tank capacity constraint, to evaluate the profitability of using AFVs.

The *Electric VRP with Time Windows and Recharging Stations (E-VRPTW)* is proposed and investigated in [Schneider et al., 2012]. The motivation is given by the fact that one of the most prominent issue of real-life BEV-based delivery systems, other than the battery charge, is related to customer time windows. Vehicles are BEV and recharging facilities are available to realign the battery charge level of the vehicles to its capacity. The aim of the E-VRPTW is to minimize the number of used vehicles first, and then the total distance traveled; however, in the proposed MILP model the objective function considers the total distance only. To solve the E-VRPTW, a metaheuristic approach based on VNS and TS is presented. The initial solution is generated via a constructive insertion algorithm that guarantees that all the time windows are fulfilled and the capacity and battery capacity constraints are satisfied by all the routes but the last. In the local search phases, a generalized cost function with penalties is used to allow intermediate infeasible solutions. The procedures to compute the penalties assure an efficient update of the cost function when evaluating neighborhoods move. The neighborhoods of the shaking phase of the VNS are all based on the cyclic-exchange operator [Thompson and Psaraftis, 1993]. The new optima at the end of the local search are always accepted if they improve the incumbent, or accepted according to probabilistic rules, otherwise. The local search phase is performed via TS, which makes use of a family of neighborhood operators either taken from the literature or newly designed. The computational evaluation relies on two sets of E-VRPTW instances derived from benchmark VRPTW instances presented in [Solomon, 1987]. On the small-sized instances of the first set, both a Branch&Bound computation with CPLEX and the proposed VNS-TS metaheuristic are run to assess the performances of the latter. On the larger ones, an analysis to evaluate the impact of the components of the metaheuristic is conducted. Finally, performances on benchmark MDVRPI and G-VRP instances are evaluated, finding several new best known solutions for both problems.

The recent work [Schneider et al., 2014] introduces the *VRP with Intermediate Stops (VRPIS)*, which generalizes many other similar problems in the literature in that all the types of intermediate stops are taken into account: replenishment/unloading locations,

refuel/recharge points and combined facilities. The VRPIS incorporates several details to model the refill/unload/refuel/recharge operations in a way as realistic as possible. The homogeneous vehicles are based at a central depot and feature a fixed usage cost, a maximum shift length, a load limit and a fuel limit, the latter being expressed in terms of travelable distance. Customers have a demand and a service time. Each facility j has a fixed docking time t_j^d, i.e. the waiting time before operations can start, and two functions $\Phi^f(f_j)$ and $\Phi^l(j_j)$ that respectively measure the time to refuel and reload-/unload; combined facilities are further characterized by functions that link refuel time and quantity to the load/unload process. Vehicles are supposed to be fully reloaded and/or refueled at facilities, while facilities have an unlimited availability of fuel and/or capacity to load/unload and can receive any vehicle, even if the total number of visits to a facility has an upper bound for modeling purposes, as it was in [Bard et al., 1998]. A MILP model is delivered, based on a graph that considers a node for each customer and each different visit to a facility, in which with every arc are associated a distance, a travel time and a cost. Other than common two-index arc variables x_{ij}, the model features variables concerning arrival time, fuel level and load level for each vertex. The objective is to minimize the sum of the total travel cost and the vehicles deployment cost. As a particular case of VRPIS, the *Electric VRP with Recharging Facilities (EVRPRF)* is defined, where there are only recharging facilities and the recharge time function is linear in the fuel level via an average recharge speed factor. The EVRPRF can be considered as a particular case of the previously mentioned E-VRPTW where customer time windows are disregarded. The authors of this challenging work propose an *Adaptive VNS (AVNS)* algorithm to tackle the VRPIS. The paradigm of the VNS which we have briefly recalled is followed. An initial solution is first determined by means of a modified savings algorithm. During the VNS, the local search operator is based on greedy algorithms: the new solution replaces the incumbent if an acceptance criterion is satisfied. The attribute *adaptive* is justified by the shaking phase, which is not completely randomized but partially guided: a subset of routes and nodes to be involved in the shaking are chosen by means of problem-tailored criteria, while specific neighborhood operator is selected according to historical performances. The algorithm is tested on the 52 benchmark G-VRP instances of [Erdoğan and Miller-Hooks, 2012] and the 76 benchmark VRPIRF instances [Crevier et al., 2007, Tarantilis et al., 2008] as particular cases of VRPIS. On all the G-VRP instances, the AVNS clearly outperforms the MCWS and DBCA algorithms, and improves the results of [Schneider et al., 2012] by either finding new best known solution, or terminating faster in tied cases. On the VRPIRF instances of [Crevier et al., 2007], the AVNS is highly competitive with the algorithms proposed by [Crevier et al., 2007], [Tarantilis et al., 2008] and [Schneider et al., 2012], but has a lower average execution time; on the VRPIRF instances of [Tarantilis et al., 2008], the AVNS is competitive for what concerns the average execution time and

able to find 15 new best known solutions, mostly among instances with 125 customers or more.

The *Periodic VRP with Intermediate Facilities (PVRP-IF)* proposed in [Angelelli and Speranza, 2002a] is another example of work where the possibility of stopping at intermediate facilities is taken into account. The paper is related to an extension of the PVRP to a collection system where the depot only acts as the base of the vehicles, while a set of intermediate facilities are distributed in the concerned area to allow the vehicles to unload and continue for other collection tours until the end of the shift. The presented TS approach is based on the heuristic algorithm of [Cordeau et al., 1995], which at that time represented the start-of-the-art for PVRP. Tests are conducted on benchmark PVRP instances, proving that the proposed technique is competitive with that of [Cordeau et al., 1995]. Moreover, a set of 54 new instances for the PVRP-IF is created. In the later work [Angelelli and Speranza, 2002b], the same authors showed how the approach proposed in [Angelelli and Speranza, 2002a] can be used to model three different collection systems and describe how the TS algorithm should be adapted to the three cases. Finally, the results derived from two real-world case studies concerning the waste collection in two areas of Northern Italy and Belgium are reported.

A real-life routing problem of wasted food collection is the motivation of the variant of the PVRP proposed in [Coene et al., 2010], which is close to [Angelelli and Speranza, 2002a]. Other than the depot and the customers, a set of disposal facilities is considered, some of which have a time window. Each customer features (a) a number of days over the considered time horizon in which it must be visited, as typical in the PVRP, (b) an average quantity to be collected, and (c) a service time. Vehicles are heterogeneous in that they have different capacities, and cannot exceed given day shift length and weekly number of driving hours. A four-index MILP formulation is proposed, along with three decomposition approaches. The first two assign customers to days, first, and then solve a VRP with problem-specific constraints using a construction-and-improvement heuristic offered by a commercial solver. They differ in how they solve the scheduling phase: the first one relies on a min-max covering technique that assigns a visit pattern to customers, whereas the second one uses a clustering approach. The third algorithm inverts the hierarchy and determines giant routes first, then decompose them into feasible routes by assigning customers to days according to the clustering technique. The computational tests are conducted on instances obtained from real data of a partner company, yielding to a substantial improvement w.r.t. the routes previously deployed by the latter.

Both the multi-trip and the multi-depot features are present in the *Synchronized, Scheduled, Multi-Depot, Multiple-Tour, Heterogeneous Vehicle Routing Problem with Time Windows (SS-MDMT-VRPTW)* and the *Two-Echelon, Synchronized, Scheduled, Multi-Depot, Multiple-Tour, Heterogeneous Vehicle Routing Problem with Time Windows (2SS-MDMT-VRPTW)* introduced in [Crainic et al., 2009] and arising in City Logistics,

that have been briefly reviewed in section 1.3. The workday plan of a city-freighter considers starting from a depot, loading freight at a satellite, performing a delivery trip to serve a subset of customers, then either go back to a depot to wait for a new scheduled duty, or meet a urban-truck to a satellite, undergo consolidation operations and start another delivery trip. The two problems can therefore be considered to all purposes as examples of rich VRPs with replenishment facilities.

In [Amaya et al., 2007], the *Capacitated Arc-Routing Problem with Refill Points (CARP-RP)* is introduced. It is an extension of the *Capacitated Arc-Routing Problem (CARP)*, a well-known problem which differs from the CVRP in that quantities to deliver are associated with edges (or arcs) instead of nodes. The CARP-RP, inspired from real-life road network maintenance applications, considers two vehicles: the *servicing (SV)* vehicle, which actually visits the compulsory arcs, and the *refilling (RV)* vehicle, both based at a central depot. The purpose of the latter is to meet the servicing vehicle to refill it: after each refill, though, the RV is compelled to return at the depot. CARP-RP is formulated on a mixed graph with arcs and edges, both having a required subset of elements which must be visited at least once. The objective is to find the route of the SV and the circuits of the RV that serve the compulsory arcs and edges at minimum overall cost while respecting the capacity of the SV. Note that the refill points too make the object of decision. The authors propose a MIP formulation and a cutting-plane algorithm. Tests have been conducted on randomly generated instances with arcs only, and on instances obtained by adapting benchmark CARP instances. The instances considered up to 70 nodes and nearly 600 arcs.

The work [Salazar-Aguilar et al., 2013] proposes the *Synchronized Arc and Node Routing Problem (SANRP)*, which extends the CARP-RP in that a fleet of SVs is considered and the RV does not need to go back to the depot after a refill, as we suppose that the RV has unlimited capacity. Therefore, many service routes are to be interlaced with the only RV route. Again, the crossing (refill) point must be decided; moreover, the SV and the RV routes must be synchronized so as to reduce the waiting time at the refill points. The objective function consists in minimizing the duration of the longest SV route. The authors propose an ALNS metaheuristic, defining three algorithms for the construction of the initial solution, and seven destroy/repair operators for the improvement phase. Sixty instances with 60-100 nodes and 200-350 arcs, and twenty larger instances with 200-400 nodes and 700-1500 arcs, were randomly generated, to test the proposed ALNS and infer a large set of performance indicators, such as the overall computation time, the effectiveness of the main algorithmic blocks, the impact of different considered replenishment policies and the position of the depot.

In [Conrad and Figliozzi, 2011], the *Recharging VRP (RVRP)* is introduced. The motivation comes from real-life applications in which BEVs are used for *less-than-truckload (LTL)* deliveries in urban areas. It is an extension of the *Distance Constrained VRP*

(DCVRP) where the vehicles are allowed to recharge at some customer locations incurring in a time penalty. The authors further refine the problem definition by yielding a three-index MILP formulation for the *Capacitated RVRP with Time Windows (CRVRP-TW)* version, where the vehicle has both a battery and a load limit and the customers have time windows. The objective function is hierarchical, aiming at the minimization of the number of used vehicles first, and of the weighted sum of the traveled distance, the service time and the vehicles recharging time, then. Thirty cases are derived from the R101 instance of [Solomon, 1987] by randomly choosing 40 customers (out of 100) and adding the driving range limit and the recharge time. Tests are made for both the CRVRP-TW and the version without time windows (CRVRP) on such testbed: results are given in terms of average number of used vehicles, traveled distance and number of recharge. Moreover, a study is conducted to establish upper and lower bounds on the solution of both the CRVRP and the CRVRP-TW.

Finally, the work [Hemmelmayr et al., 2013] proposes an heuristic approach for three problems, namely: the *Periodic VRP with Intermediate Facilities (PVRP-IF)*; a variant of the PVRP-IF where vehicles are not compelled to be empty when they go back to the depot – a feature inspired by a real-life application; and the VRPIRF - although the authors claim to deal with the MDVRPI. In spite of not being the most recent work to tackle the VRPIRF, [Hemmelmayr et al., 2013] is the last one of our review since the presented results on the VRPIRF benchmark instances are the best that we are aware of. A MIP model for the PVRP-IF is presented, based on the formulation for the PVRP of [Cordeau et al., 1995]. The MDVRPI is then presented as a particular case of the PVRP-IF with a single-day planning horizon: since the PVRP-IF has only one central depot, such particular case is actually not the MDVRPI, but the VRPIRF instead. The proposed solution method is a VNS. The initial solution is built by first randomly assigning customers to days (this step is not required for the VRPIRF), then running a Clarke and Wright savings heuristic to solve a VRP under the maximum duration constraint, and finally inserting the intermediate facilities with an exact procedure based on *Dynamic Programming (DP)*. The used neighborhoods operators for the VRPIRF can be of two type: *move*, i.e. the reallocation of a sequence of customers from a route to another, or *swap*, i.e. two sequences of two routes are exchanged between them. Both intra- and inter-rotation moves are considered, whereas this is not the case for intra-route moves, for the sake of computation tractability. Moreover, a *2-opt**-based [Potvin and Rousseau, 1995] operator is especially designed for the VRPIRF. The neighborhood size is given by the maximum number of customers of the moved/swapped route segments. A set of local search tools, mainly based on 2-opt and 3-opt, are used alone or combined; moreover, the aforementioned DP-based procedure to insert facilities in the vehicles' rotations is also used in the LS phase. For the real-life variant of the PVRP-IF, however, this latter tool is replaced by a greedy heuristic. Neighborhood operators do

not tolerate infeasible intermediate solutions, and neither does the LS phase. At the end of the local search, the incumbent x is replaced by the new solution x'' both in case of improvement and of deterioration, the latter case according to a probability which depends on $f(x) - f(x'')$, f being the objective function. A set of new benchmark instances is proposed for the PVRP-IF. Curiously, this set is obtained by combining the features of the PVRP instances of [Cordeau et al., 1995] and those of the MDVRPI instances of [Crevier et al., 2007], as the sets of customers are the same for both datasets. On the VRPIRF benchmark instances, the VNS of [Hemmelmayr et al., 2013] outperforms all the other review methods and achieves remarkable results. When limited to 10 runs, the algorithm ties or improves the previously best known solutions for 64 out of 76 instances, with an average +0.32% gap over the remaining instances. Moreover, the best known solution is improved or tied on 74 cases through sensitivity analysis.

3.3 Conclusions

This chapter has introduced the Vehicle Routing Problem with Intermediate Replenishment Facilities, a tactical Vehicle Routing problem. An extensive review of the literature of the VRPIRF, as well as of that of the VRPs it descends from, and the problems which are most similar to it, has been later proposed. This review has clearly highlighted how the VRPIRF arises in the context of real-world applications. Therefore, even if more difficult variants have been widely studied, and numerous heuristic methods have been developed to deal with large-scale instances, the literature is still scarce for what concerns exact methods, even for the basic version. This motivates the work presented in the remainder of Part II, which is devoted to the introduction of four exact approaches to the VRPIRF.

Chapter 4

Branch & Cut algorithms for the Vehicle Routing Problem with Intermediate Replenishment Facilities

4.1 A first Branch & Cut Algorithm based on a 3-index formulation

In the first part of this chapter, we will introduce some basic notation for the VRPIRF (section 4.1.1), a first MILP compact formulation (section 4.1.2), the Branch & Cut algorithm based on it (4.1.3) and its computational evaluation (4.1.4). Section 4.2 will then introduce a second Branch & Cut-based approach. Finally, section 4.3 will conclude the chapter and draw future perspectives.

4.1.1 Notations

In its most frequent definition, VRPIRF is defined on an oriented graph $G = (V, A)$ where the node set $V = \{0,\dots,n+f\}$ consists of the customers' set $C = \{0,\dots,n-1\}$, the depot, denoted by $\Delta \equiv n$, and the facilities' set $F = \{n+1,\dots,n+f\}$, while the arc set is $A = V \times C \cup C \times V$, i.e. the graph is complete except for links between Δ and the facilites, between the facilities – and of course loops. Each arc $ij \in A$ has associated cost d_{ij} and travel time τ_{ij}, both of which are supposed to respect the triangle inequality. The demand and the unload time of client $i \in C$ are denoted by q_i and τ_i. $K = \{1,\dots,n_K\}$

denotes the set of n_K vehicles, whereas their load and duration limits are referred to as Q and T. Recharge at facility $p \in F$ requires a time τ_p, while the time to load a vehicle at the central depot before the beginning of a rotation is τ_Δ. One can define *extended* arc travel times as follows in order to include node service times:

$$(\forall ij \in A)\ t_{ij} = \tfrac{1}{2}\ \tau_i + \tau_{ij} + \tfrac{1}{2}\ \tau_j$$

We assume that for each pair $i, j \in C$ of customer it is possible to determine a facility $p \in F$ which is the most convenient to recharge in between i and j in terms of both cost and (extended) travel time, i.e. such that the recharge cost $d_{ip} + d_{pj}$ and the recharge time $t_{ip} + t_{pj}$ are minimum:

$$(\forall i, j \in C,\ i \neq j)\ (\exists p \in F)\ (\forall q \in F \backslash \{p\})\ d_{ip} + d_{pj} \leq d_{iq} + d_{qj} \wedge t_{ip} + t_{pj} \leq t_{iq} + t_{qj}$$

This assumption is reasonable in most of the cases: for instance, it holds for most of the VRPIRF instances in the literature. However, its removal could easily be dealt with.
It is useful, for the sake of conciseness, to introduce the following notations.
Given two sets of customers $S_1, S_2 \subseteq C$, let:

- $\overline{S_1}$ denote the node set $C \backslash S_1$
- $S_1 : S_2$ denote the oriented cut-set $\{ij \in A : i \in S_1, j \in S_2\}$,
- $\delta^+(S_1)$ and $\delta^-(S_1)$ represent the cut-sets $(V \backslash S_1) : S_1$ and $S_1 : (V \backslash S_1)$,
- $A(S_1)$ denote $S_1 : S_1$, i.e. the arcs with both endpoints in S_1,

In addition, we also define:

- $i : S := \{ij \in A : j \in S\}$, $S : i := \{ji \in A : j \in S\}$, $\delta^-(i) := \delta^-(\{i\})$ and $\delta^+(i) := \delta^+(\{i\})$ to denote some particular arc sets concerning node $i \in V$,
- $\mathcal{S}(C)$ as the collection $\{S \subset C : 2 \leq |S| \leq |C| - 2\}$ of subsets of customers,
- $\kappa(S)$ as the minimum number of routes needed to serve customers in $S \subseteq C$ – the solution of a *Bin Packing Problem* (BPP) instance on S, and $r(S) = \left\lceil \frac{\sum_{i \in S} q_i}{Q} \right\rceil$ a trivial lower bound on $\kappa(S)$.

4.1.2 The 3-Index Vehicle-Flow Formulation

A first approach for modelling VRPIRF relies on a three-index formulation. The decision variables are:

- binary *arc-flow variables* x_{ij}^k, with $x_{ij}^k = 1 \Leftrightarrow$ arc $ij \in A$ is visited by vehicle $k \in K$;
- binary *activity variables* y_p^k, $y_p^k = 1 \Leftrightarrow$ vehicle k recharges at least once at $p \in F$.

Let notation $x^k(A')$ denote the sum $\sum_{ij\in A'} x^k_{ij}$ for any $A' \subseteq A$.
The first proposed MILP model for VRPIRF is the following:

$$
\begin{aligned}
&(\mathcal{M}^{BC})\\
&\min \quad \sum_{k\in K}\sum_{ij\in A} d_{ij}x^k_{ij} && && (4.1)\\
&\text{s.t.} \quad \sum_{ij\in A} t_{ij}x^k_{ij} \le T && \forall k\in K && (4.2)\\
&\qquad \sum_{k\in K} x^k(\delta^+(i)) = 1 && \forall i\in C && (4.3)\\
&\qquad x^k(\delta^+(i)) = x^k(\delta^-(i)) && \forall i\in C,\ k\in K && (4.4)\\
&\qquad x^k(\Delta{:}C) \le 1 && \forall k\in K && (4.5)\\
&\qquad x^k(\Delta{:}C) = x^k(C{:}\Delta) && \forall k\in K && (4.6)\\
&\qquad x^k(p{:}C) = x^k(C{:}p) && \forall k\in K && (4.7)\\
&\qquad \sum_{k\in K} x^k(A(S)) \le |S| - \kappa(S) && \forall S\in\mathcal{S}(C) && (4.8)\\
&\qquad x^k_{ip} \le y^k_p && \forall i\in C,\ k\in K,\ p\in F && (4.9)\\
&\qquad x^k(\overline{S}{:}S) \ge y^k_p && \forall k\in K,\ p\in F,\ S\subseteq V\backslash\{p\} && (4.10)\\
&\qquad x^k(S{:}\overline{S}) \ge y^k_p && \forall k\in K,\ p\in F,\ S\subseteq V\backslash\{p\} && (4.11)\\
&\qquad \sum_{k\in K}(x^k(\delta^-(\Delta)) + \sum_{p\in F} x^k(\delta^-(p))) \ge \kappa(C) && \forall k\in K && (4.12)\\
&\qquad x^k(\delta^+(i)) \le x^k(C{:}\Delta) && \forall i\in C,\ k\in K && (4.13)\\
&\qquad x^k_{ij}\in\{0,1\} && \forall k\in K, ij\in A &&\\
&\qquad y^k_p\in\{0,1\} && \forall k\in K, p\in F &&
\end{aligned}
$$

Constraints (4.2) impose the shift length limit for each vehicle. Relations (4.3) and (4.4) require each client to be visited exactly once by exactly one vehicle. Constraints (4.5) and (4.6) state that a vehicle may not leave the depot, but if it is used, it must return at it. Similarly to (4.6), (4.7) assert that each time a vehicle k enters a facility p, it must leave it. *Capacity inequalities* (4.8) impose that no route violates the vehicle load limit. Constraints (4.9)–(4.11) are needed to prevent connection issues. Indeed, constraints (4.3)–(4.7) fail to ensure that routes performed by a vehicle k form a *continuous sequence* starting and ending at the depot: any solution with one vehicle performing one or more routes that visit a subset $F' \subseteq F$ of facilities, and such that no sequence of routes connects the depot to any of the elements of F', is easily seen not to violate (4.3)–(4.7). An example is given in Figure 4.1.
This instrinsic weakness is mainly due to the chosen support graph and the arc variables, which do not contain route information. Activity variables and constraints (4.9)–(4.11)

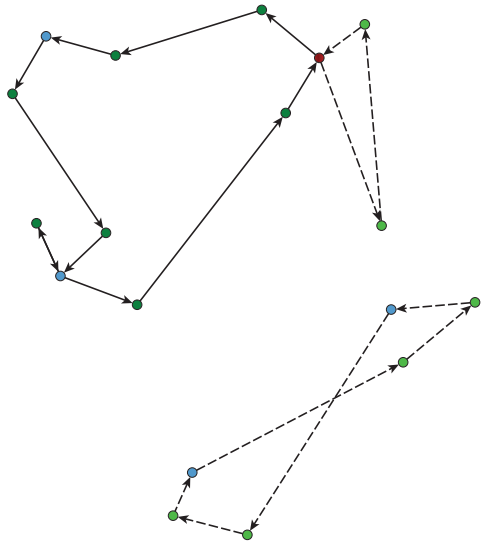


FIGURE 4.1: An example of non-connected solution of the previous instance. Two vehicles are used, but the rotation of the first one (continuous stroke) is connected, whereas that of the second one (dashed stroke) is not.

allow to overcome these problems. Each constraint (4.9) –which we call *activity constraints*– sets corresponding variable y_p^k to 1 whenever vehicle k performs some route that ends up at facility p; if so, constraints (4.10)–(4.11) –which we call *connectivity constraints*– guarantee that there is a path (possibly passing through other facilities) from Δ to p and another one from p to Δ.

Finally, constraints (4.12) and (4.13) are modeling tricks, as they aim at raising the lower bound: the former express that there must be at least as many arcs leaving the depot and the facilities (i.e., as many routes) as the solution of BPP on the whole customers' set, whereas the latter force a vehicle to leave the depot whenever it serves a client.

Note that constraints (4.9) could be aggregated as $\sum_{i \in C} x_{ip}^k \leq M y_p^k$, but they would be much weaker due to their *big-M* form, also because M coefficient could not even be tailored as we cannot know a priori the number of times vehicle k will recharge at facility p – unless we explicitly bound such number, as it has been done in [Bard et al., 1998].

Since (4.8), (4.10) and (4.11) are in exponential number, solving model $\mathcal{M}^{BC}$ calls for the design of a Branch & Cut algorithm in order to progressively separate and add them to the model as *lazy* constraints (i.e. they are introduced only when they are found to be violated), both at the root node and in the nodes of the Branch & Bound tree.

4.1.3 Separation and Branch & Cut Strategy

In this section we focus on the separation of constraints (4.8) and (4.10)–(4.11), before giving some insights on the Branch & Cut algorithm.

4.1.3.1 Separation of capacity inequalities

Constraints (4.8) are commonly found in algorithms to solve capacitated VRPs, like for instance in the Branch&Cut algorithm designed in [Lysgaard et al., 2004]. In the same paper, the authors coded *CVRPSEP* (see [Lysgaard, 2003]), a library of routines to separate various families of valid cuts for symmetric CVRP. In order to use such routines, a transformation of the support graph G used so far is needed. We call it *α-transformation* and define it as follows.
Given a non-oriented support graph $G_\alpha = (V_\alpha, E_\alpha)$ with $V_\alpha = \{\Delta\} \cup C$, $E_\alpha = \{ij \mid i,j \in V_\alpha : i < j\}$, the generic solution $\mathbf{x}$ of $\mathcal{M}^{BC}$ is transformed as:

$$(\forall i,j \in C : i<j)\;\; x_{ij}|^\alpha = \sum_{k\in K} x^k_{ij} + x^k_{ji}\;;\; x_{i\Delta}|^\alpha = \sum_{k\in K} \left(x^k_{i\Delta} + x^k_{\Delta i} + \sum_{p\in F} x^k_{ip} + x^k_{pi}\right) \quad (4.14)$$

i.e. α-transformation collapses the facilities and the depot in one node and changes replenishments into trips to the depot and back. CVRPSEP routines are then fed with the transformed solution $\mathbf{x}|^\alpha$, the vector $q_i|i \in C$ of customer demands and the vehicle capacity Q, returning one or more collection of subsets of C for which a violation occurs w.r.t. the sought type of cut. For capacity inequalities, on each one of the identified subsets we impose the corresponding capacity cuts, which must be formulated in one of the following equivalent forms, according to which one is the sparsest:

$$x(\delta(S)) \geq 2\kappa(S) \quad (4.15)$$

$$x(E(S)) \leq |S| - \kappa(S) \quad (4.16)$$

$$x(E(\overline{S})) + \tfrac{1}{2}x(\overline{S}:\Delta) - \tfrac{1}{2}x(S:\Delta) \leq |\overline{S}| - \kappa(S) \quad (4.17)$$

where $\delta(S)$, $E(S)$, $S:\Delta$ replace $\delta^+(S)$, $\delta^-(S)$, $A(S)$, $S:\Delta$, $\Delta:S$ in the symmetric case. (4.16) descends from (4.15) and from:

$$\begin{aligned} & 2x(E(S)) + x(\delta(S)) = 2|S| \\ \Rightarrow\;\; & 2|S| - 2x(E(S)) = x(\delta(S)) \geq 2\kappa(S) \\ \Rightarrow\;\; & 2|S| - 2\kappa(S) \geq 2x(E(S)) \end{aligned}$$

To find (4.17), consider identity $2x(E(C)) + x(\delta(\Delta)) = 2|C|$, then we have:

$$\begin{aligned} & 2\big(x(E(S)) + x(E(\overline{S})) + x(S:\overline{S})\big) + x(S:\Delta) + x(\overline{S}:\Delta) = 2|C| \\ \Rightarrow\; & x(E(S)) + x(E(\overline{S})) + x(S:\overline{S}) + \tfrac{1}{2}x(S:\Delta) + \tfrac{1}{2}x(\overline{S}:\Delta) = |C| \\ \Rightarrow\; & x(S:\overline{S}) + x(S:\Delta) = |C| - x(E(S)) - x(E(\overline{S})) - \tfrac{1}{2}x(\overline{S}:\Delta) + \tfrac{1}{2}x(S:\Delta) \end{aligned}$$

and since $x(S{:}\overline{S}) + x(S{:}\Delta) = x(\delta(S))$ and $|S| = x(E(S)) + \frac{1}{2}x(\delta(S))$ we obtain:

$$\begin{aligned} & |S| + |\overline{S}| - x(E(S)) - x(E(\overline{S})) - \tfrac{1}{2}x(\overline{S}{:}\Delta) + \tfrac{1}{2}x(S{:}\Delta) - x(\delta(S)) = 0 \\ \Rightarrow\ & |\overline{S}| - x(E(\overline{S})) - \tfrac{1}{2}x(\overline{S}{:}\Delta) + \tfrac{1}{2}x(S{:}\Delta) - \tfrac{1}{2}x(\delta(S)) = 0 \\ \Rightarrow\ & x(E(\overline{S})) + \tfrac{1}{2}x(\overline{S}{:}\Delta) - \tfrac{1}{2}x(S{:}\Delta) = |\overline{S}| - \tfrac{1}{2}x(\delta(S)) \leq |\overline{S}| - \kappa(S) \end{aligned}$$

In order to use (4.15)–(4.17) in our framework, they need to be adapted to the directed case, so as to obtain:

$$x(\delta^+(S)) = x(\delta^-(S)) \geq \kappa(S) \tag{4.18}$$

$$x(A(S)) \leq |S| - \kappa(S) \tag{4.19}$$

$$x(A(\overline{S})) + \tfrac{1}{2}\big(x(\Delta{:}\overline{S}) + x(\overline{S}{:}\Delta) - x(\Delta{:}S) - x(S{:}\Delta)\big) \leq |\overline{S}| - \kappa(S) \tag{4.20}$$

(4.19) descends from (4.18) and identity $|S| = x(A(S)) + x(\delta^+(S)) = x(A(S)) + x(\delta^-(S))$. To derive (4.20), expand identity $x(A(C)) + \frac{1}{2}x(\delta^+(\Delta)) + \frac{1}{2}x(\delta^-(\Delta)) = |C|$ as:

$$\begin{aligned} & x(A(S)) + x(A(\overline{S})) + x(S{:}\overline{S}) + x(\overline{S}{:}S) + \\ & + \tfrac{1}{2}x(S{:}\Delta) + \tfrac{1}{2}x(\Delta{:}S) + \tfrac{1}{2}x(\overline{S}{:}\Delta) + \tfrac{1}{2}x(\Delta{:}\overline{S}) = |S| + |\overline{S}| \end{aligned}$$

and since $|S| = x(A(S)) + x(\delta^+(S)) = x(A(S)) + x(\overline{S}{:}S) + x(\Delta{:}S)$ we have:

$$\begin{aligned} & x(A(\overline{S})) + x(S{:}\overline{S}) - \tfrac{1}{2}x(\Delta{:}S) + \tfrac{1}{2}x(\Delta{:}\overline{S}) + \tfrac{1}{2}x(S{:}\Delta) + \tfrac{1}{2}x(\overline{S}{:}\Delta) = |\overline{S}| \\ \Rightarrow\ & x(A(\overline{S})) - \tfrac{1}{2}x(\Delta{:}S) + \tfrac{1}{2}x(\Delta{:}\overline{S}) - \tfrac{1}{2}x(S{:}\Delta) + \tfrac{1}{2}x(\overline{S}{:}\Delta) = \\ & = |\overline{S}| - x(S{:}\Delta) - x(S{:}\overline{S}) = |\overline{S}| - x(\delta^-(S)) \leq |\overline{S}| - \kappa(S) \end{aligned}$$

In practice, the sparsest form is always one of (4.19) and (4.20).
For each of the customers' subset for which the capacity inequality (4.16) is found to be violated, the CVRPSEP routines provide both the set itself and the right-hand side of the cut which, for the sake of computational tractability, is $|S| - r(S)$ instead of $|S| - \kappa(S)$; as a consequence, when we impose (4.19) and (4.20) in our Branch&Cut algorithm, we do it in the form of *rounded capacity inequalities* (RCI), which are weaker but still valid:

$$x(A(S)) \leq |S| - r(S) \tag{4.21}$$

$$x(A(\overline{S})) + \tfrac{1}{2}\big(x(\Delta{:}\overline{S}) + x(\overline{S}{:}\Delta) - x(\Delta{:}S) - x(S{:}\Delta)\big) \leq |\overline{S}| - r(S) \tag{4.22}$$

4.1.3.2 Separation of connectivity constraints

The separation of (4.10)–(4.11) simply calls for the solution of as many maxflow-mincut problems as twice the number of activity variables y_p^k which have value greater than 0. If $y_p^k > 0$, i.e. if vehicle k has some *activity* at facility p, then two maxflow-mincut problems are solved, with the current $\mathcal{M}^{BC}$ solution as support graph, Δ as source and p as sink for the first, and the opposite for the second. If the maximum flow of the first problem is under a given threshold parameter then the constraint (4.10) is imposed on the corresponding mincut set, and similarly for (4.11) w.r.t. the second problem.

4.1.3.3 Branch & Cut Strategy

The Branch & Cut algorithm has been implemented using the *CPLEX 12.5* framework with the following Branch & Cut strategy:

- at root node: given the current fractional solution, separate both RCI and cuts (4.10)–(4.11); add each found violated constraint to the model if any, then reoptimize, otherwise terminate root node computation
- at a generic node of the Branch & Bound tree: after CPLEX has added its cuts, look for violated RCIs; if any of them is found, add them to the model and let CPLEX reoptimize. If the node solution is integer, before returning control to CPLEX, separate cuts (4.10)–(4.11), and add to the model any violated one

4.1.4 Computational evaluation

Preliminar tests have been conducted to stress the effectiveness of the Branch & Cut algorithm based on model $\mathcal{M}^{BC}$. The instances used were those generated by the authors of [Tarantilis et al., 2008], where instance features range from 50 to 175 customers, 2 to 7 facilities, 2 to 8 vehicles. Results were indeed very promising for the smallest (50 customers) instances, for which solutions very close to the best known reported in the same article were found in reasonable computational times, in spite of poor quality lower bounds: even if the optimality gap was sometimes hard to close, it seemed that such algorithm could at least be used to fastly produce good upper bounds. Unfortunately, this behavior dramatically deteriorated with the growth of instances' size, and even 75 customers instances resulted to be nearly intractable.

4.1.4.1 Connection Issues

Computational experience suggested that the way $\mathcal{M}^{BC}$ deals with connection problems, in spite of being theoretically correct, is strongly ineffective. Indeed, when a fractional solution is found to violate some (4.10) or (4.11) constraints and they are added, they are easy to circumvent when reoptimizing. A very huge number of such constraints could then be needed, but then the computation would suffer, whereas a desired feature of a good Branch & Cut algorithm is to be able to separate and add as few and effective cuts as possible. The aforementioned strategy, where connectivity cuts are separated and added only in case of integer solution of a Branch & Cut node, tries to limit such massive cuts' generation, but with the drawback to allow fractional solutions to be nonconnected and therefore to worsen the lower bound and slow down the convergence.

4.1.4.2 Vehicle-related Issues

The weakness of $\mathcal{M}^{BC}$, and especially of its lower bound, can certainly be also ascribed to the symmetries of the problem due to the multiplicity of identical vehicles. Attempts have been made to deal with this issue, for example trying to rewrite constraints (4.2) in a symmetry-breaking form like (4.23):

$$\sum_{ij \in A} t_{ij} x_{ij}^k \leq \begin{cases} \sum_{ij \in A} t_{ij} x_{ij}^{k+1} & , \quad k < n_K \\ T & , \quad k = n_K \end{cases} \tag{4.23}$$

Another weak point of the two models is the fact that fractional solutions can make a fractional use of vehicles, and this is mainly due to the fact that constraints (4.5) are loose. The way this can impact the effectiveness of the algorithm can be explained with a simple reasoning. Suppose we have a VRPIRF instance $\overline{I}$ with T and Q both large enough to allow to visit all the clients with a single route. The problem degenerates to a TSP and its optimal solution $\overline{\mathbf{x}}$ will certainly make use of only one vehicle $k = 0$ which does not even need to stop at any of the facilities to recharge. Let $\overline{T}$ be the duration of the only route of $\overline{\mathbf{x}}$, and $\overline{z}$ the cost of $\overline{\mathbf{x}}$. In a VRPIRF instance $\overline{I}'$ identical to $\overline{I}$ but with $T = \lfloor \frac{\overline{T}}{m} \rfloor$ and $m \in \mathbb{N}, m \geq 2$, the optimal solution $\overline{\mathbf{x}}'$ will require at least $m + 1$ vehicles since the end of k-th vehicle's rotation and the start of the shift of the $(k + 1)$-th will require an extra length w.r.t. $\overline{z}$. If such extra length is not negligible, the optimality gap at the root node will be large, as the solution of the LP relaxation of the root node problem will not have a value greater than $\overline{z}$, i.e. the value of a fractional solution where:

$$(\forall ij \in A : \overline{x_{ij}^0} = 1, \ \forall k) \quad \overline{x_{ij}^k}' = \frac{1}{m+1}$$

i.e., each of the $m+1$ used vehicles retraces the same only route of $\overline{\mathbf{x}}$ but at a fractional value $\frac{1}{m+1}$. Again, attempts have been made to overcome this further weakness of the model. For example, given a VRPIRF instance I that satisfies the assumption:

$$(\forall i_1j_1, i_2j_2 \in A : i_1j_1 \neq i_2j_2) \quad d_{i_1j_1} > d_{i_2j_2} \Rightarrow t_{i_1j_1} > t_{i_2j_2} \tag{4.24}$$

then given any two solutions $\mathbf{x}_1$, $\mathbf{x}_2$ of I, one could infer the following:

$$\sum_{k\in K}\sum_{ij\in A} d_{ij}((x^k_{ij})_1 - (x^k_{ij})_2) > 0 \Rightarrow \sum_{k\in K}\sum_{ij\in A} t_{ij}((x^k_{ij})_1 - (x^k_{ij})_2) > 0$$

i.e. if the cost of $\mathbf{x}_1$ is greater than that of $\mathbf{x}_2$, then the same applies for the sum of the shift lengths over the set of all vehicles, no matter if $\mathbf{x}_1$ and $\mathbf{x}_2$ are fractional or integer. Note that assumption (4.24) can be restrictive but is worth to be considered since it holds for most of the VRPIRF instances that are found in the literature. If such assumption holds, then given the solution $\overline{\mathbf{x}}$ of a node $\overline{n}$ of the Branch&Bound tree of model $\mathcal{M}^{BC}$, it is not restrictive to say that any integer solution in the subtree with root in $\overline{n}$ will make use of at least $\overline{n_K} = \left\lceil \frac{1}{T}\sum_{k\in K}\sum_{ij\in A} t_{ij}\overline{x^k_{ij}} \right\rceil$ vehicles. This is valid even at the root node, and can be easily detected and imposed by changing constraints (4.5) into equalities for a subset of n_K vehicles, both for problem at node $\overline{n}$ and its offspring; the vehicles must be the n_K with higher indexes, to be compliant to (4.23).
All of these tricks have been implemented, yet they turned out to be ineffective.

4.2 A new Branch&Cut Algorithm with Replenishment Arcs and Arrival Times

In this section we propose another Branch&Cut algorithm that relies on a new compact MILP formulation for VRPIRF based on *replenishment arcs* (section 4.2.1) and *arrival times* (section 4.2.2) which allows to overcome the weaknesses of model $\mathcal{M}^{BC}$. The new formulation is explained in section 4.2.3, while sections 4.2.4 and 4.2.5 introduce the Branch&Cut algorithm and the computational results.

4.2.1 Replenishment Arcs

Replenishment arcs are a powerful modeling concept which has been used, as far as we know, by just a few works in the literature. The paper we refer to is [Smith et al., 2012], which makes use of such concept in a generalization of the Shortest Path Problem with Resource Constraint (SPPRC) called the Weight Constrained Shortest Path Problem

with Replenishment (WCSPP-R) to model activities that reset a cumulated amount of a given resource. Another work that exploits a similar feature is the aforementioned [Karaoglan and Koc, 2011] for a Branch & Cut algorithm for the MTVRP. In the context of VRPIRF, a *replenishment arc* is a special arc, which we denote by $i^p j$, that connects two customer nodes i and j and which represent a stop at facility p to perform a recharge operation in between the two customers. Therefore, replenishment arc $i^p j$ carries much information at the same time, as its use means that:

- i is the last visited customer of its route, and j is the first of the following one;
- the two routes are performed by the same vehicle.

This has two fundamental consequences, both of which are well-shown in figure 4.2:

1. the information about the stop at facility p is embedded in the arc, so facility nodes are no more needed;
2. rotations can be represented as a sequence of arcs –possibly of both type, base and replenishment– where each node has indegree and outdegree equal to 1, with the exception of the depot.

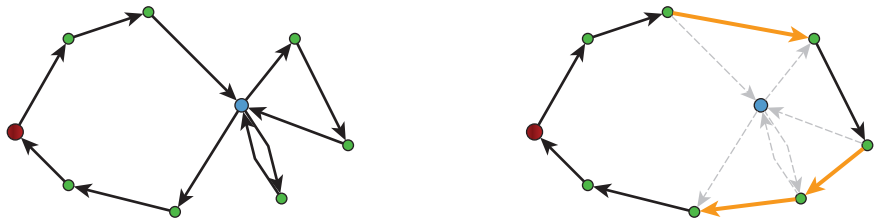


FIGURE 4.2: The same rotation with replenishment arcs (thicker and drawn in orange, right image) and without.

The above point 2 is the most crucial, as it states that with replenishment arcs a rotation becomes very similar to a common route in CVRP, the only difference being the fact that the former is composed of both base and replenishment arcs.
A replenishment arc $i^p j$ has cost $d_{i^p j} = d_{ip} + d_{pj}$ and extended travel time $t_{i^p j} = t_{ip} + t_{pj}$. Given two customer nodes $i, j \in C$ and a facility p, $i^p j$ is said to be *dominated* if there exists a facility q such that $d_{iqj} \leq d_{i^p j}$ and $t_{iqj} \leq t_{i^p j}$, otherwise $i^p j$ is said to be *nondominated*. According to the assumptions of section 4.1.1, for each pair $i, j \in C$ there is only one replenishment arc that connects them which is nondominated. Roughly speaking, for any two customer nodes there is a facility which is the most convenient to recharge in between them in terms of both time and cost. Hence, we can discard notation $i^p j$, use notation ij for both base ($ij \in A_0$) and replenishment ($ij \in A_P$) arcs

and define:

$$(\forall i,j \in C) \qquad f_{ij} = \min_{p \in P} \; d_{i^p j} \; ; \qquad u_{ij} = \min_{p \in P} \; t_{i^p j}$$

For ease of notation, we will reuse symbols V and A without ambiguity. Node set $V = \{0,\dots,n\} = C \cup \{\Delta\}$ consists now only of the depot node and the customer nodes; we have a *facility set* $P = \{1,\dots,f\}$ rather than facility nodes, while the arc set $A = A_0 \cup A_P$ is now composed of the set $A_0 = V \times C \cup C \times V$ of *base arcs* and the set A_P of nondominated replenishment arcs:

$$A_P = \{i^p j : \; i,j \in C, \; i \neq j, \; p \in P, \; (\nexists q \in P \backslash \{p\}) \; d_{i^q j} \leq d_{i^p j} \wedge t_{i^q j} \leq t_{i^p j}\}$$

We extend some of the arc set symbols introduced before so as to distinguish between sets made of only base arcs (subscript 0), and sets made of only replenishment arcs (subscript P). Given $S_1, S_2 \subseteq C$, let:

- $(S_1 : S_2)_0$ denote the cut-set of only base arcs $\{ij \in A_0 : i \in S_1, j \in S_2\}$,
- $(S_1 : S_2)_P$ denote the cut-set of only replenishment arcs $\{ij \in A_P : i \in S_1, j \in S_2\}$,
- $\delta_0^+(S_1) := ((V \backslash S_1) : S_1)_0$ and $\delta_0^-(S_1) := (S_1 : (V \backslash S_1))_0$,
- $\delta_P^+(S_1) := ((V \backslash S_1) : S_1)_P$ and $\delta_P^-(S_1) := (S_1 : (V \backslash S_1))_P$,
- $A_0(S_1) := (S_1 : S_1)_0$ and $A_P(S_1) := (S_1 : S_1)_P$

while for what concerns a generic node $i \in V$ we define:

- $(i : S)_0 := (\{i\} : S)_0$, $(S : i)_0 := (S : \{i\})_0$, $\delta_0^+(i) := \delta_0^+(\{i\})$, $\delta_0^-(i) := \delta_0^-(\{i\})$
- $(i : S)_P := (\{i\} : S)_P$, $(S : i)_P := (S : \{i\})_P$, $\delta_P^+(i) := \delta_P^+(\{i\})$, $\delta_P^-(i) := \delta_P^-(\{i\})$,

4.2.2 Arrival Times

In order to deal with the vehicle-related issues depicted in section 4.1.4.2, it would be desirable to drop the vehicle index out of the decision variables. We observe that in $\mathcal{M}^{BC}$ such vehicle index is only needed to compute the total travel time of each $k \in K$ and thus impose the duration constraint. The recent literature on *Asymmetric Distance-Constrained VRP* (or *ADVRP*) proposes an interesting technique to keep track, given any partial path from the depot to a customer i, of the distance travelled so far. The paper we refer to is [Almoustafa et al., 2013], even though its authors claim to have taken the core idea from other works; we refer the reader to [Kara, 2011], among the

others. Something similar has also been used by the authors of [Bard et al., 1998].
Such technique makes use of $\mathcal{O}(n^2)$ variables z_{ij} associated to arcs and can be summarized as follows. In a oriented graph $G = (V, A)$, let x_{ij}, $ij \in A$ be the usual binary arc-flow variables and let $x_{ij} = 1$ if we go from node i to node j. Let also z_{ij}, $ij \in A$ be real nonnegative variables, with z_{ij} representing the distance travelled from the depot to node j *if* its predecessor is node i, or 0 otherwise: its value is imposed by means of the following equation:

$$\sum_{j \in V \setminus \{i\}} z_{ij} = \sum_{j \in V \setminus \{i\}} z_{ji} + \sum_{j \in V \setminus \{i\}} t_{ij} x_{ij} \tag{4.25}$$

The relation is easily seen to be recursive, therefore by adding a family of base-case relations one can compute the distance travelled up to any node of a route; an additional family of relations to treat return-to-depot cases allows to impose that the total route distance does not exceed the given bound. Section 4.2.3 will define in detail these relations and how they have been adapted to our case, and will illustrate with a small example (figure 4.5) how they work.
This technique can be adapted to our case to handle *arrival times* at each node along a route, i.e. the time the vehicle arrives at a node and –in case of a customer node– before service starts. By imposing that no node arrival time exceeds T, the maximum shift length can be enforced without needing to distinguish the vehicles.
It is interesting to note that one could also make use of techniques commonly used in *VRP with Time-Windows* to express arrival times; however, the relations used in there are nonlinear and often dealt with by means of big-M-based linearization, with the result of weakening the LP relaxation of the model, whereas relations (4.25) are linear.
Figures 4.3 and 4.4 have been taken from some preliminary tests and aim at giving a hint of the higher effectiveness of the two-index formulation w.r.t. the three-index one. They depict two fractional solutions, both obtained during the root node processing of the instance previously shown in figures 3.1, 3.2 and 4.1. The first one is delivered by the $\mathcal{M}^{BC}$-based Branch&Cut algorithm, while the second one refers to a solution yielded by a two-index-formulation-based Branch&Cut algorithm. In spite of being two different solutions, their values are very close, allowing us to make a comparison between them. It appears clearly that the first solution is much more scattered and prone to symmetry issues: therefore, its solving is likely to either have a slower convergence to the optimal fractional solution or deliver a worse lower bound.
Let us make one last observation. Not surprisingly, the use of relations (4.25) to enforce the duration maximum length prevents subtours as a side-effect, as the arrival times along a path can only be nondecreasing. Indeed, such relations represent a generalization of some subtour elimination techniques used in the context of the *Traveling Salesman*

Problem, the best-known being the one proposed by Miller, Tucker and Zemlin in [Miller et al., 1960].

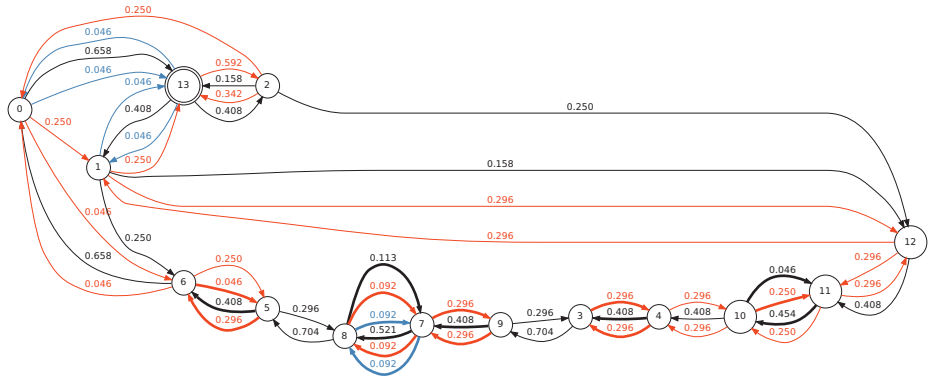


FIGURE 4.3: A fractional solution delivered by the $\mathcal{M}^{BC}$-based Branch&Cut algorithm during the root node processing of the previous instance. Bold arrows denote replenishment arcs; different colors refer to different vehicles.

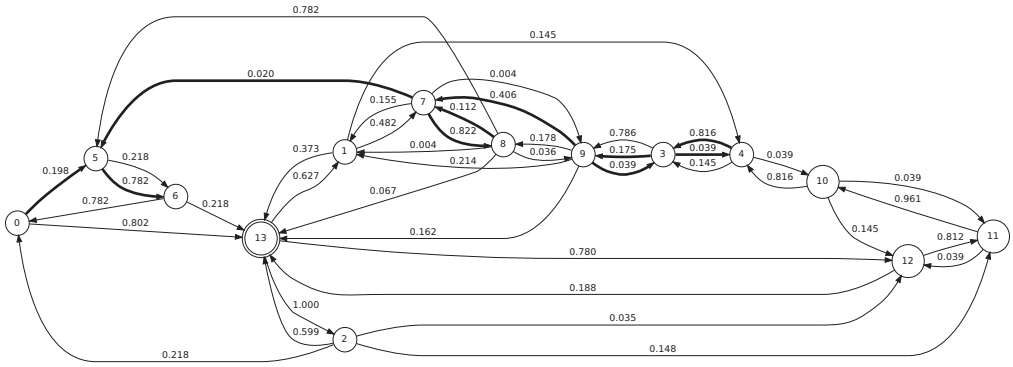


FIGURE 4.4: Another fractional solution of the root node processing of the previous instance. This time, the solution is yielded by a two-index-formulation-based Branch& Cut algorithm.

4.2.3 A Two-Index Formulation

A new MILP model for VRPIRF, $\mathcal{M}^{BC}_{RA}$, is now proposed. Its decision variables are:

- ▶ binary *base arc variables* x_{ij}, $ij \in A_0$, $x_{ij} = 1 \Leftrightarrow$ node j follows node i in the same route;
- ▶ binary *replenishment arc variables* w_{ij}, $ij \in A_P$, $w_{ij} = 1 \Leftrightarrow$ vehicle recharges (at the least cost facility) between clients i, j;
- ▶ real nonnegative *arrival time variables* z_{ij}, $i \in V$, $j \in V \backslash \{i\}$, which represent the arrival time at node j if its predecessor is node i.

The aggregated notation $x()$ introduced before is extended in the following sense:

- $x(A')$ will denote the sum $\sum_{ij \in A'} x_{ij}$ whenever $A' \subseteq A_0$, and
- $w(A')$ will denote the sum $\sum_{ij \in A'} w_{ij}$ whenever $A' \subseteq A_P$.

Model $\mathcal{M}_{RA}^{BC}$ is:

$$
\begin{aligned}
&(\mathcal{M}_{RA}^{BC}) \\
&\min \quad \sum_{ij \in A_0} d_{ij} x_{ij} + \sum_{ij \in A_P} f_{ij} w_{ij} && && (4.26)\\
&\text{s.t.} \quad x(\delta_0^-(\Delta)) + w(A_P) \geq \kappa(C) && && (4.27)\\
&\qquad x(\delta_0^+(i)) + w(\delta_P^+(i)) = x(\delta_0^-(i)) + w(\delta_P^-(i)) && \forall i \in C && (4.28)\\
&\qquad x(\delta_0^+(i)) + w(\delta_P^+(i)) = 1 && \forall i \in C && (4.29)\\
&\qquad x(\delta_0^-(\Delta)) \leq n_K && && (4.30)\\
&\qquad x(\delta_0^+(\Delta)) = x(\delta_0^-(\Delta)) && && (4.31)\\
&\qquad x(A_0(S)) \leq |S| - \kappa(S) && \forall S \in \mathcal{S}(C) && (4.32)\\
&\qquad z_{\Delta i} = t_{\Delta i} x_{\Delta i} && \forall i \in C && (4.33)\\
&\qquad z_{ij} \geq (t_{\Delta i} + t_{ij}) x_{ij} + (t_{\Delta i} + u_{ij}) w_{ij} && \forall i \in C, j \in C \backslash \{i\} && (4.34)\\
&\qquad z_{ij} \leq (T - t_{j\Delta})(x_{ij} + w_{ij}) && \forall i \in C, j \in C \backslash \{i\} && (4.35)\\
&\qquad z_{i\Delta} \geq (t_{\Delta i} + t_{i\Delta}) x_{i\Delta} && \forall i \in C && (4.36)\\
&\qquad z_{i\Delta} \leq T x_{i\Delta} && \forall i \in C && (4.37)\\
&\qquad \sum_{j \in V \backslash \{i\}} z_{ij} = \sum_{j \in V \backslash \{i\}} z_{ji} + \sum_{j \in V \backslash \{i\}} t_{ij} x_{ij} + \sum_{j \in C \backslash \{i\}} u_{ij} w_{ij} && \forall i \in C && (4.38)\\
&\qquad x_{ij} \in \{0, 1\} && \forall ij \in A_0 &&\\
&\qquad w_{ij} \in \{0, 1\} && \forall ij \in A_P &&\\
&\qquad z_{ij} \geq 0 && \forall i \in V, j \in V \backslash \{i\} &&
\end{aligned}
$$

The objective function (4.26) takes into account both base and replenishment arcs. The contraints can be subdivided into three major families:

- *degree constraints*:
  - (4.27) enforces a lower bound on the number of required routes, similarly to (4.12);
  - (4.28) and (4.29) impose to visit each client exactly once;
  - (4.30) and (4.31) bound to n_K the number of service rotations that can be performed and force each performed rotation to leave and enter the depot exactly once;
- *capacity constraints*:
  (4.32) force each route to respect the vehicle load limit. Since capacity constraints

are the only ones in exponential number, they form the core of the new Branch&Cut algorithm: we will discuss them more in depth later;

- *arrival times constraints*:
  - (4.33) determine the arrival time at the first customer of each rotation;
  - if customer j follows customer i on a route, (4.34) and (4.35) respectively impose tailored lower and upper bounds on the variable z_{ij}; otherwise, (4.35) sets z_{ij} to 0;
  - (4.37) impose that for each arc that enters the depot, i.e. the last one of the rotation of a vehicle, the arrival time must be less than, or equal to, T, while (4.36) set a trivial lower bound on the same arrival time;
  - (4.38) recursively impose the arrival times at intermediate customer nodes of a rotation other than the first one, according to the principle briefly explained in section 4.2.2.

Figure 4.5 can help the understanding of how (4.33)–(4.38) work. In this example,

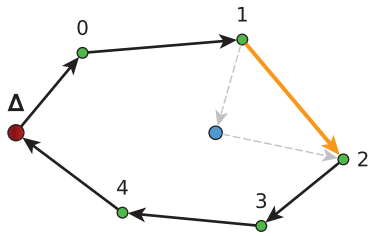


FIGURE 4.5: An instance with $f = 1$ and $n = 5$ and a solution with only one rotation.

$x_{ij} = w_{ij} = 0$ for every $ij \in A$ except for $x_{\Delta 0} = x_{01} = w_{12} = x_{23} = x_{34} = x_{4\Delta} = 1$; (4.35) sets every z variable to 0 with the exception of $z_{\Delta 0}$, z_{01}, z_{12}, z_{23}, z_{34} and $z_{4\Delta}$. (4.33) imposes $z_{\Delta 0} = t_{\Delta 0}$, whereas (4.38) for node $i = 0$ is:

$$\sum_{j \in V \setminus \{0\}} z_{0j} = z_{01} = \sum_{j \in V \setminus \{0\}} z_{j0} + \sum_{j \in V \setminus \{0\}} t_{0j} x_{0j} + \sum_{j \in C \setminus \{0\}} u_{0j} w_{0j} = z_{\Delta 0} + t_{01}$$

and similarly we get $z_{12} = z_{01} + u_{12}$, $z_{23} = z_{12} + t_{23}$, $z_{34} = z_{23} + t_{34}$ and $z_{4\Delta} = z_{34} + t_{4\Delta}$. Since arrival time $z_{4\Delta}$ is imposed the bound T by (4.37), the vehicle performing the rotation is ensured to be back at the depot within the maximum shift length.
The small example above also shows how arrival times constraints are sufficient to enforce the connection of the rotation of a vehicle. Suppose by contradiction to have a solution with $x_{\Delta 0} = x_{04} = x_{4\Delta} = 1$, $w_{12} = x_{23} = x_{31} = 1$: this would result in $z_{12} = z_{31} + u_{12} = z_{23} + t_{34} + u_{12} = z_{12} + t_{23} + t_{34} + u_{12}$, which is impossible unless $t_{23} + t_{34} + u_{12} = 0$.

4.2.4 The new Branch & Cut Algorithm

In the following, we explain in detail how the separation capacity constraints (4.32) is performed. Then, we introduce some families of *valid inequalities* which are used in the separation procedure performed at each node of the Branch & Bound tree, and we illustrate how they are separated. Lastly, we provide some insights on the separation procedure itself and the overall Branch & Cut algorithm.

4.2.4.1 Separation of Capacity Constraints

Relations (4.32) are *problem-defining constraints*, i.e. they are all required to define the solution space of VRPIRF. In other words, by removing a part or all of them, the solution space of $\mathcal{M}_{RA}^{BC}$ would contain solutions which do not match feasible VRPIRF solutions, hence $\mathcal{M}_{RA}^{BC}$ would not be a correct model of the problem.
However, to find an optimal solution of a VRPIRF instance it is generally not necessary to add them all. The purpose of a separation algorithm is precisely to gradually detect and add violated problem-defining constraints, until the feasibility of the currently best found solution is proven – and thus so is its optimality. It is therefore a desirable feature of such an algorithm to be able to detect as few and effective cuts as possible at each step, in order to allow a fast convergence while adding a least possible number of constraints.
Such constraints are often referred to as *lazy constraints*, because even if they are necessary to define the problem, they are added only when they are violated, in order to cut off a current solution which is infeasible w.r.t. the definition of the problem; the reason of this *lazy* insertion in the model is their number, which is exponential in the problem instance size. Capacity inequalities (4.32) are separated in the same way as in section 4.1.3.1. Since the support graph of the problem has considerably changed, the transformation of a solution $\mathbf{x}$ of $\mathcal{M}_{RA}^{BC}$ which is required to feed CVRPSEP routines must be revised too. We call this new transformation *β-transformation* and define it as follows.
Given a non-oriented support graph $G_\beta = (V, E_\beta)$ with $E_\beta = \{ij \mid i, j \in V : i < j\}$, the generic solution $\mathbf{x}$ of $\mathcal{M}_{RA}^{BC}$ is transformed as:

$$(\forall i, j \in C : i < j) \quad x_{ij}|^\beta = x_{ij} + x_{ji} \; ; \; x_{i\Delta}|^\beta = x_{i\Delta} + x_{\Delta i} + \sum_{j \in C \setminus \{i\}} (w_{ij} + w_{ji}) \quad (4.39)$$

The specific CVRPSEP routine to separate capacity cuts is fed with the transformed solution $\mathbf{x}|^\beta$ and returns one or more subsets $S \in \mathcal{S}(C)$ for which the capacity constraint is violated: for such a subset S we impose (4.32) in the form of RCI. For ease of language, this separation procedure will be referred to in the following as *β-separation*.

4.2.4.2 Separation of Connectivity Valid Inequalities

Along with problem-defining contraints, there are other families of contraints, most often called *valid inequalities*, which are not required to define the solution space of VRPIRF, as any integer solution which is compliant to the former (and to the other constraints in the model) automatically satisfies the latter – which are then redundant. This is not the case, however, for what concerns the LP relaxation of the problem, the solution space of which can be considerably reduced by the addition to the model of such inequalities. This is why, when solving the LP relaxation of a node of the Branch & Bound tree during a Branch & Cut algorithm, it is very common to separate and add them in order to refine the fractional solution of the node and tighten the lower bound.
This is the case of *connectivity inequalities* (4.40):

$$(\forall S \in \mathcal{S}(C)) \quad x(A_0(S)) + w(A_P(S)) \leq |S| - 1 \tag{4.40}$$

They are a generalization of *subtour elimination constraints (SECs)* where both base and replenishment arcs are taken into account, so as to exploit the previously recalled structural similarity between a rotation expressed with replenishment arcs, and a CVRP route.
In order to separate them, a further transformation of a solution of $\mathcal{M}_{RA}^{BC}$ is required. We call it γ-*transformation* and define it as follows. Given support oriented graph $G_\gamma = (V, A_0)$, a solution $\mathbf{x}$ of $\mathcal{M}_{RA}^{BC}$ is transformed as:

$$(\forall i, j \in C : i \neq j) \quad x_{ij}|^\gamma = x_{ij} + w_{ij} \ ; \ x_{i\Delta}|^\gamma = x_{i\Delta} \ ; \ x_{\Delta i}|^\gamma = x_{\Delta i} \tag{4.41}$$

The separation of (4.40) calls for the separation of SECs on the transformed solution $\mathbf{x}|^\gamma$. This is usually achieved with a maxflow algorithm, like for instance in [Fischetti et al., 1997]. Given the transformed solution $\mathbf{x}|^\gamma$, a maximum flow problem on the support graph G_γ is formulated where each arc ij has associated capacity $x_{ij}|^\gamma$: for each customer node $i \in C$, the determination of the maximum flow from Δ to i yields the Δ–i cut $S \in \mathcal{S}(C)$ such that $i \in S$ and whose capacity $x(\delta_0^+(S))|^\gamma$ is minimal. If such capacity is less than 1, the connectivity constraint on S can be imposed. Therefore, by solving $|C| = n$ maxflow problems, one can find up to n (without taking into account identical sets) of such sets $S \in \mathcal{S}(C)$. An alternative way to separate connectivity constraints consists in using the CVRPSEP routine to separate capacity constraints and feeding it with the transformed solution $\mathbf{x}|^\gamma$, a vector $q_i = 1|_{i \in C}$ of *unit customer demands* and a vehicle load limit $Q = |C|$. In spite of being less conventional, such method is efficient and more homogeneous w.r.t. the β-separation procedure. We have adopted and implemented this latter method. For brevity, in the following we will talk about

γ-*separation* to denote it.
The imposition of the connectivity constraints on the sets S obtained this way can be done either in the previously shown form (4.40) or in the equivalent one:

$$x(\delta_0^+(S)) + w(\delta_P^+(S)) \geq 1 \tag{4.42}$$

depending on which one is the sparsest.

4.2.4.3 Separation of Homogeneous Multistar Inequalities

Multistar inequalities and *partial multistar inequalities* are well-known examples of valid inequalities for CVRP. They were first introduced in [Araque et al., 1990] in the context of a study on *Symmetric Unit demand CVRP (CVRPUD)*: the report still offers a strong introductory reference on the subject. Partial multistar inequalities have the form:

$$\lambda x(E(N)) + x(E(C:S)) \leq \gamma$$

where $N \subset C$, $C \subset N$ and $S \subset C \backslash N$ are subsets of customer nodes respectively called *nucleus*, *connectors* and *satellites*. A partial multistar is the subgraph $E(N) \cup E(C:S)$. By setting parameters λ and γ according to $|N|$, $|S|$ and $|C|$, in many cases it is possible to cut fractional solutions that would otherwise respect capacity cuts. Similar considerations apply to multistars, which can be considered as the particular case with $C \equiv N$.
The authors of [Letchford et al., 2002] later generalized such inequalities to CVRP and determined some *homogeneous* multistar and partial multistar inequalities, which follow the same principle but take into account customer demands and the vehicle capacity. The attribute *homogeneous* denotes the fact that the edges of $E(C:S)$ have the same coefficient regardless of the concerned customers, as it was in the original form for CVRPUD. The same authors later embedded the separation of homogeneous multistar and partial multistar inequalities in the Branch&Cut framework presented in [Lysgaard et al., 2004]: the heuristic separation procedure used to detect violated inequalities is again part of the CVRPSEP package.
We have used this routine in our Branch&Cut algorithm to cut and possibly improve the fractional solution $\mathbf{x}$ of the generic Branch&Bound node; however, in order to do so, the CVRPSEP procedure must be fed with the β-transformation $\mathbf{x}|^\beta$ since it is tailored for the Symmetric CVRP.

4.2.4.4 Separation Policy and Branch & Cut Algorithm

The Branch & Cut algorithm is based on the separation policy outlined in Figure 4.6. After solving the LP relaxation of the subproblem associated with the current node of

NodeSeparation(T_l, θ_β, θ_γ, θ_μ)

```
argtype  T_l: timeLim;  θ_β, θ_γ, θ_μ: threshold;
declare  t: time;  x, x|^β, x|^γ: solution;  S_β, S_γ, S_μ: setCollection;  S: set;
1   startTime(t)
2   x ← LPoptimize(M_RA^BC)
3   while(true)
4     x|^β ← β-transformation(x)
5     S_β ← Separate(x|^β, θ_β, 'B')
6     if(S_β ≠ ∅)
7       foreach(S in S_β) addToModel(M_RA^BC, capacityC(S))
8       x ← LPoptimize(M_RA^BC)
9       if(t > T_l) exit else continue
10    if(x is integer feasible) exit
11    x|^γ ← γ-transformation(x)
12    S_γ ← separate(x|^γ, θ_γ, 'G')
13    if(S_γ ≠ ∅)
14      foreach(S in S_γ) addToModel(M_RA^BC, connectionC(S))
15      x ← LPoptimize(M_RA^BC)
16      if(t > T_l) exit else continue
17    S_μ ← separate(x|^β, θ_μ, 'M')
18    if(S_μ ≠ ∅)
19      foreach(S in S_μ) addToModel(M_RA^BC, hMultistarC(S))
20      x ← LPoptimize(M_RA^BC)
21      if(t > T_l) exit else continue
22  exit
```

Separate(x, θ, type)

```
argtype  x: solution;  θ: threshold;  type: char;
argcond  type ∈ {'B', 'G', 'M'};
returns  setCollection;
declare  S: setCollection; S: set;
1   S ← ∅
2   if(type = 'B') perform β-separation on x
3   if(type = 'G') perform γ-separation on x
4   if(type = 'M') perform Homogeneous Multistar separation on x
5   foreach(set S whose constraint is violated by more than θ) S ← S ∪ {S}
6   return S
```

Figure 4.6: Separation policy performed at each node of the Branch & Bound tree.

the Branch & Bound tree, the algorithm iteratively performs a sequence of calls to the

described separation procedures, each with an associated threshold violation value θ. Following a classical general scheme, the separation procedures are invoked according to a given order: whenever one of them finds one or more violated constraints, they are added to the model and a new LP optimization is called. The algorithm updates the relaxed solution of the node and jumps to the following iteration, seeking for further violated constraints, unless the given time limit T_l has been exceeded. Note that the time limit check only occurs at the end of the separation procedures. The node computation terminates when either no more constraints can be added, or the integrality of the current solution is proved. In the former case, the final lower bound of the node is delivered, whereas in the latter a new valid solution has been found and the node will not be branched on further.

As for the order of the procedure calls, first we separate capacity constraints (line 4). If no violated constraint has been found, the integrality of the current solution is checked (line 10). If the check succeeds, the computation can end. If it fails, we look for violated connectivity valid inequalities first (line 11), and possibly violated homogeneous multistar valid inequalities (line 17), then. If none is found, the computation ends.

Figures 4.7 show some details of the procedure NodeSeparation. Subfigure (a) represents a nontransformed solution that violates the capacity constraint on set S: since $|S| = 8$ and $r(S) = 3$, we add the RCI $x(A_0(S)) \leq 5$ to $\mathcal{M}_{RA}^{BC}$ and we reoptimize. Subfigures (b) and (d) are possible cases of the nontransformed solution of the following iteration: (b) does not violate any other constraint (as shown by its β-transformation (c)), and since it is integer, the procedure NodeSeparation ends. On the contrary, (d) violates the capacity constraint on S_1 (as shown by (e)), which is then added.

The Branch&Cut algorithm itself consists of three phases:

1. root node separation, i.e. the procedure NodeSeparation at the root node;
2. root node refinements, i.e. the further processing performed by CPLEX by adding its standard cuts, prior to branching on the root node;
3. the Branch&Bound tree search, during which the procedure NodeSeparation is invoked by CPLEX at each node when no more standard cuts can be added and without time limit.

4.2.5 Computational results

The Branch&Cut algorithm based on model $\mathcal{M}_{RA}^{BC}$ and procedure NodeSeparation has been implemented using the CPLEX Callable Library of the IBM ILOG CPLEX suite, version 12.6. This section describes the results of the computational experiments that have been conducted to assess its performances. The tests have been run on a Intel

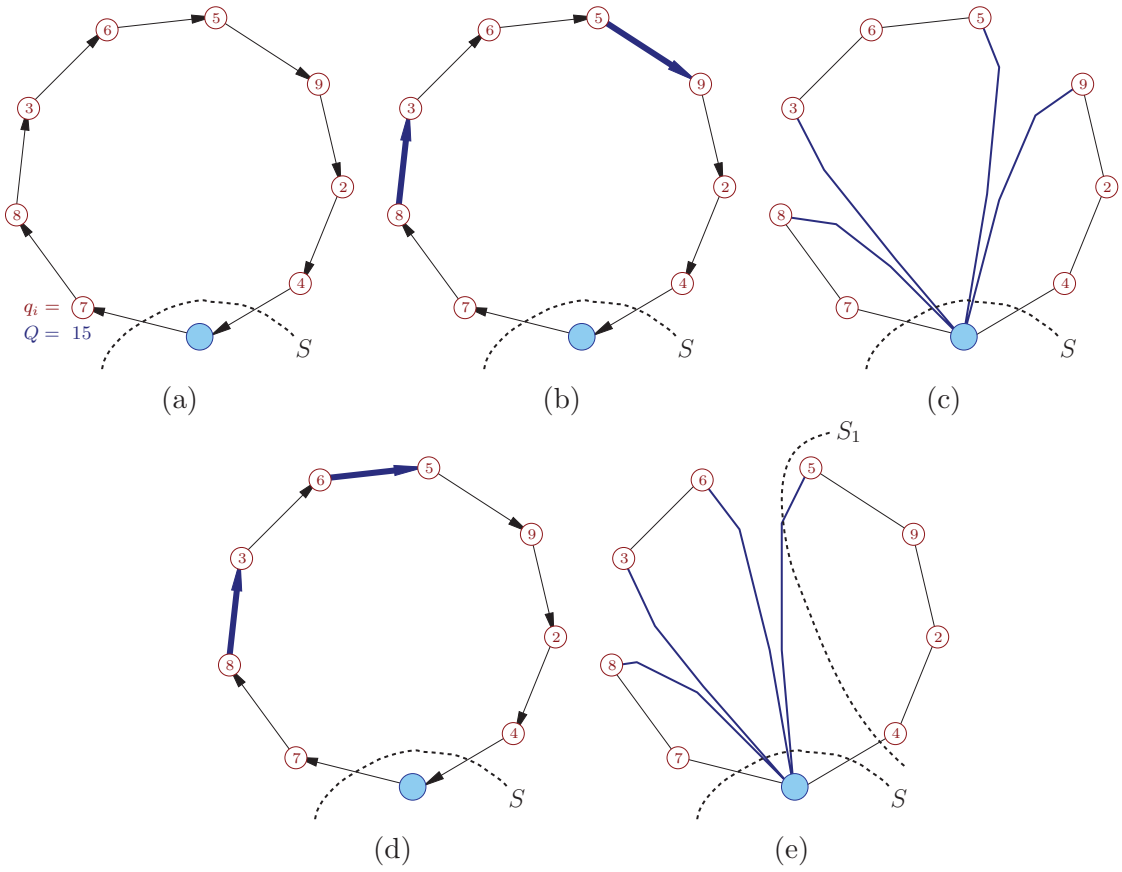


FIGURE 4.7: Different stages of the separation procedure on an example instance with $Q = 15$; customer demands are shown in red circles. The original support graph is represented by means of a diagram with black and thick blue arcs, which stand for base and replenishment arcs, respectively. The graph representing the transformed solution output of the β-transformation features black and blue edges.

Core i7-4770 3.4 Ghz machine with 7.76Gb RAM. A single-threaded computation has been imposed.

The VRPIRF instances that have been used for the test sessions are the 22 instances *a1–l1* and *a2–j2* presented in [Crevier et al., 2007], and the 54 instances *50c3d2v–175c8d8v* proposed in [Tarantilis et al., 2008]: they will be respectively referred to as *CCL* and *TZK* instances, after the names of the authors, for brevity. In these instances, features range from 48 to 288 customers, 2 to 7 replenishment facilities, and 2 to 8 vehicles.

The Branch&Cut algorithm has always been given the best known solution (which will be referred to as BKS from now on) as term of comparison. As far as we are aware of, the BKS is that of [Hemmelmayr et al., 2013] in 74 out of 76 cases, i.e. with the exception of TZK instances *125c4d7v* and *175c8d8v* for which the best known result is the one achieved by the AVNS of [Schneider et al., 2014]. In both cases, however, the gap between the two solutions is less than 0.1%.

On small-sized instances, i.e. with 48 to 75 customer nodes, a complete computation (phases 1-3, cf 4.2.4.4) has been performed, with a time limit of 3600 or 5400 seconds

on both phase 1 (i.e. the root node separation) and the remainder of the computation. The Branch&Bound search has been given the BKS as initial upper bound. Tables 4.2 and 4.4 show the results on small-sized TZK and CCL instances, respectively. No new best solution has been found, hence only the BKS is reported. The *initial gap* is the one between the BKS and the lower bound after phase 1, while the *best gap* is the one at the end of the computation. The cases in which optimality is proved are highlighted. The reader is referred to table 4.1 to have the names of the table columns explained.
On bigger instances, i.e. with 96 customer nodes or more, only phases 1 and 2 have been conducted with an overall time limit of 3600 or 7200 seconds depending on whether the instances have less than 200 customers, or more. Tables 4.3 and 4.5 provide a full detail of these tests. Only the gap between the BKS and the root lower bound has been evaluated on these instances. Nevertheless, such analysis still offers strong indicators on the strength of the MILP formulation, even without exploring the Branch&Bound tree. Moreover, to prove the quality of the NodeSeparation procedure, both the gap after separation (phase 1) and that after CPLEX root refinements (phase 2) are reported, respectively as initial and best gap. Whenever the root separation exceeded the overall time limit (which is possible, see section 4.2.4.4), the root refinement is not performed and the corresponding time, as well as the best gap, do not appear.
Finally, table 4.6 shows the effect of the valid inequalities. On a sample of instance families, we have performed phase 1 computation after removing them from procedure NodeSeparation, and we have compared the results to those obtained with the full separation procedure. The same time limit 3600 seconds has been imposed.

notation	description
instance	instance name
T_1	time limit(s) for phase 1 (small instances)
T_{23}	time limit(s) for phases 2 & 3 (small instances)
T_{12}	time limit(s) for phases 1 & 2 (big instances)
t_1	computational time(s) for phase 1
t_{23}	computational time(s) for phases 2 & 3 (small instances)
t_2	computational time(s) for phase 2 (big instances)
BKS	previously best known solution
$\%^1$	initial gap after phase 1
$\%^{23}$	best gap after phases 2 & 3 (small instances)
$\%^2$	best gap after phase 2 (big instances)
$\underline{\%^1}$	initial gap after phase 1 without valid inequalities (table 4.6)
$\underline{123}$	proof of optimality

Table 4.1: Key of the results' tables.

The computational results are reported and discussed in the following.
Table 4.2 shows that with small-sized TZK instances the proposed Branch&Cut algorithm is in general capable of very good gaps at the end of the root node computation: in 12 instances out of 18, such gap is less than 8%, and it is not much greater in the

instance	n	f	n_K	$T_1 = T_{23}$	t_1	t_{23}	BKS	$\%^1$	$\%^{23}$
50c3d2v	50	2	2	3600	4.51	42.44	2209.83	1.89	0.00
50c3d4v	50	2	4	3600	1.37	3600.12	2368.33	11.07	9.61
50c3d6v	50	2	6	3600	2.07	3600.01	2999.29	11.57	11.11
50c5d2v	50	4	2	3600	3.75	386.21	2608.25	2.18	0.00
50c5d4v	50	4	4	3600	2.01	3600.01	3086.58	8.27	6.98
50c5d6v	50	4	6	3600	1.97	3600.01	3548.88	12.44	9.90
50c7d2v	50	6	2	3600	5.78	437.26	3353.08	2.11	0.00
50c7d4v	50	6	4	3600	3.92	3600.01	3380.27	2.88	0.25
50c7d6v	50	6	6	3600	2.33	3600.01	4074.43	11.94	10.08
75c3d2v	75	2	2	5400	36.19	4101.01	2678.79	1.62	0.00
75c3d4v	75	2	4	5400	26.44	5400.31	2746.73	2.94	1.44
75c3d6v	75	2	6	5400	59.33	5400.04	3393.88	7.85	7.65
75c5d2v	75	4	2	5400	40.71	5400.28	3373.68	3.48	2.43
75c5d4v	75	4	4	5400	16.78	5400.25	3553.46	6.07	5.54
75c5d6v	75	4	6	5400	20.24	5400.06	4184.65	8.13	7.97
75c7d2v	75	6	2	5400	42.71	5400.01	3569.01	1.90	0.63
75c7d4v	75	6	4	5400	13.64	5400.22	3822.09	4.99	4.22
75c7d6v	75	6	6	5400	12.49	5400.10	4239.76	7.62	6.76

TABLE 4.2: Results on TZK instances with 50 to 75 customers.

other cases. Moreover, in 14 cases, the difference between the initial and the best gap, which we refer to as the *gap difference*, is less than 2%, to remark that the contribution of the root separation phase is important. Table 4.2 also shows that n_K still impacts the algorithm's performances, as the gap grows with it. It is noteworthy to say that TZK instances with equal n and f have the same customers, although they differ in maximum vehicle load and maximum shift duration: this makes the analysis of the impact of n_K possible. It is more difficult to determine whether there is a dependence on the number of facilities, as instances with same n and different f have different datasets, e.g. the client nodes of instances *50c3d2v*, *50c5d2v* and *50c7d2v* are located differently. In some cases, the gaps decrease as the number f of facilities grows, like e.g. with instances *50c_d4v*, whereas in other cases the behaviour is more controversial, like e.g. with instances *75c_d2v* or *75c_d4v*. Indeed, in the case of family *50c_d4v*, the decreasing behaviour of the gaps seems more likely to be due to the progressive clusterization and scattering of the customer nodes, as shown in figure 4.8. Analogously, figure 4.9 may explain the irregular variations of the gaps in the family *75c_d4v*.

The dependence on n_K is confirmed by table 4.3 which describes the tests on bigger TZK instances, although there is a small number of exceptions. However, these latter could also be caused by an inferior quality of the BKS. Still we have very good results, with the root node gap under 10% in 31 out of 36 cases, and under 7% in 20 of them, which are remarkable numbers if we consider that the root node time limit is not increased with the growth of the instance size and thus a worsening of the gap would be largely predictable. However, such good results could depend on the fact that TZK big-sized instances generally have very scattered customer nodes, as shown with some examples

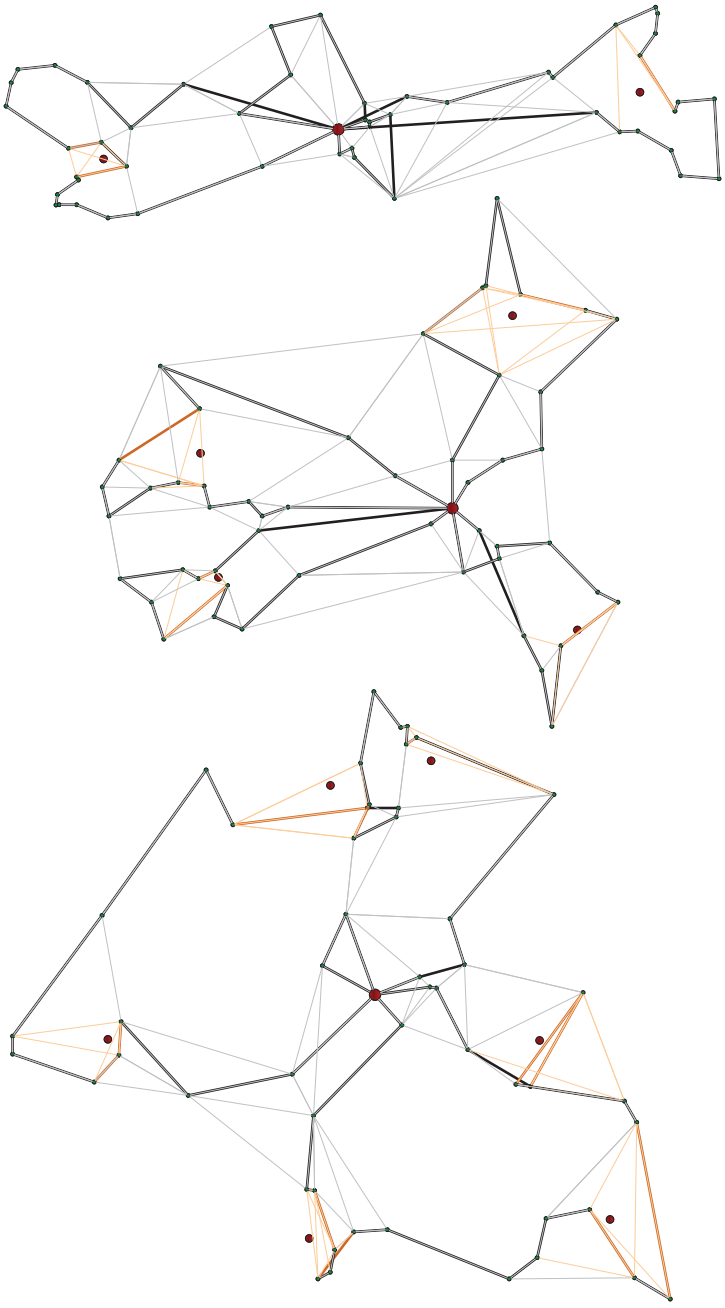


FIGURE 4.8: TZK instances *50c_d4v* (f increases from top to bottom), along with their BKS (black, orange) and the traces of the fractionary solutions (grey, light orange) at the end of phase 1. For ease of visualization, arcs do not have arrowheads. Red nodes are the facilities: the biggest one is the central depot.

by figures 4.10 and 4.11, resulting in a more simple decisional context from a combinatorial point of view. Note that on the big-sized TZK instances where the phase 2, i.e. the root refinement, is performed, the gap difference is always under 0.2%. However, differently from small-sized instances, the gap difference between the two gaps is due to the root refinement only.

Tables 4.4 and 4.5 show results on CCL small- and big-sized instances.

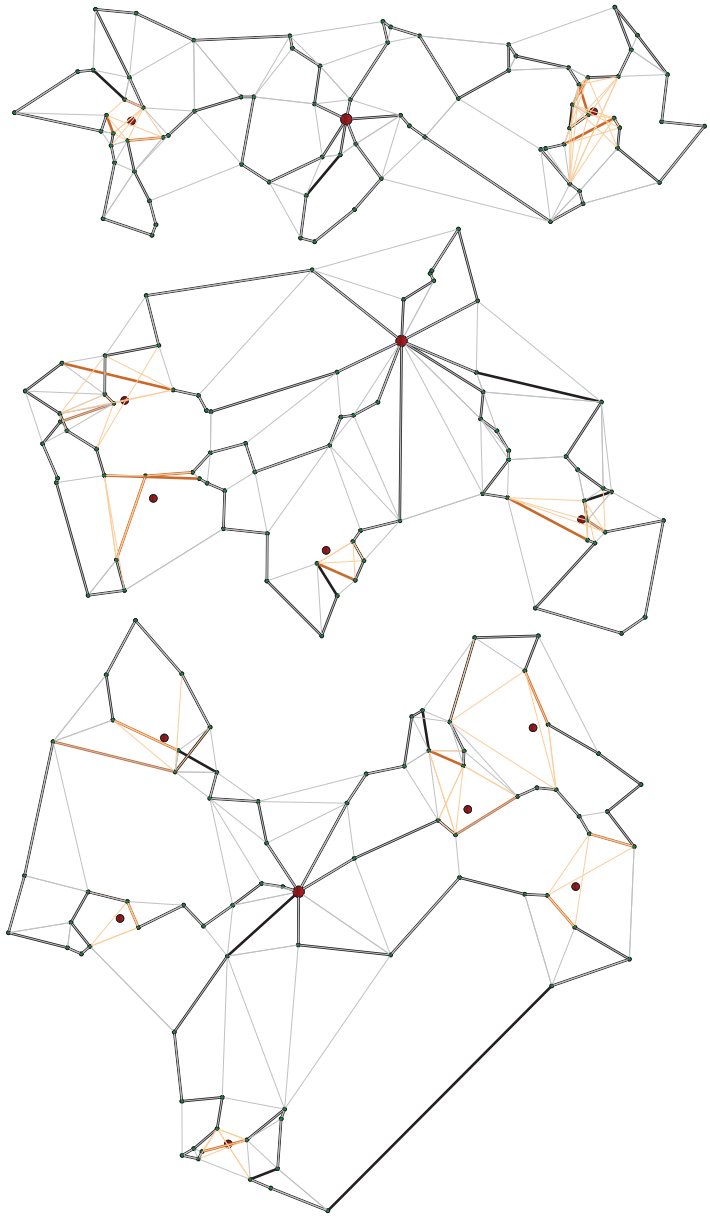


FIGURE 4.9: BKS and traces of the fractionary solutions for the TZK instances *75c_d4v* (top to bottom).

These results follow the same general trend for what concerns the growth of the size. Note that CCL instances have all different customer nodesets, nevertheless we can still infer a general dependence of the root node gaps on the number of vehicles n_K. On small-sized instances, the root node gap is rarely under 5% even with small instances, but if we consider that we never have less than 4 vehicles, then the results are similar to those on TZK instances.

If we compare a small CCL instance with a TZK instance with similar features, then the gaps are better in the first case, which is surprising given that small CCL instance generally have more dense customer nodesets. In this sense, figure 4.12 compares the

instance	n	f	n_K	T_{12}	t_1	t_2	BKS	$\%^1$	$\%^2$
100c3d3v	100	2	3	3600	170.76	34.84	3123.51	2.47	2.47
100c3d5v	100	2	5	3600	62.75	76.81	3548.44	13.58	13.58
100c3d7v	100	2	7	3600	372.76	2692.72	4235.30	9.40	9.32
100c5d3v	100	4	3	3600	244.81	84.93	4053.95	2.60	2.47
100c5d5v	100	4	5	3600	17.12	25.49	4413.16	9.36	9.23
100c5d7v	100	4	7	3600	84.26	49.56	5142.52	13.71	13.62
100c7d3v	100	6	3	3600	158.96	62.10	4207.79	5.41	5.41
100c7d5v	100	6	5	3600	54.55	54.67	4412.85	10.25	10.25
100c7d7v	100	6	7	3600	94.58	61.94	4869.65	10.46	10.46
125c4d3v	125	3	3	3600	465.86	158.00	3916.01	2.47	2.36
125c4d5v	125	3	5	3600	117.57	156.57	4308.44	9.36	9.36
125c4d7v	125	3	7	3600	289.77	162.71	4664.38	10.87	10.87
125c6d3v	125	5	3	3600	329.12	103.28	4063.25	2.55	2.55
125c6d5v	125	5	5	3600	791.67	1041.44	4760.46	6.25	6.15
125c6d7v	125	5	7	3600	265.55	148.37	5164.02	7.83	7.81
125c8d3v	125	7	3	3600	992.42	135.43	4534.14	3.97	3.97
125c8d5v	125	7	5	3600	1839.76	1765.75	4947.00	5.08	5.08
125c8d7v	125	7	7	3600	1184.34	2489.37	5334.91	6.94	6.94
150c4d3v	150	3	3	3600	2060.68	464.98	4049.47	2.09	2.09
150c4d5v	150	3	5	3600	1566.35	2933.70	4618.71	7.28	7.28
150c4d7v	150	3	7	3600	1311.58	3827.35	5118.40	9.82	9.82
150c6d3v	150	5	3	3600	940.12	138.51	4057.08	4.12	4.09
150c6d5v	150	5	5	3600	3620.32		4855.28	5.96	
150c6d7v	150	5	7	3600	3798.86		5695.25	7.95	
150c8d3v	150	7	3	3600	1749.28	247.99	4641.29	3.15	3.08
150c8d5v	150	7	5	3600	424.25	251.98	5065.10	6.41	6.41
150c8d7v	150	7	7	3600	444.10	327.18	5605.82	9.08	9.08
175c4d4v	175	3	4	3600	3652.25		4692.53	3.40	
175c4d6v	175	3	6	3600	3458.49	142.12	4816.54	4.13	4.13
175c4d8v	175	3	8	3600	3664.08		5830.62	9.87	
175c6d4v	175	5	4	3600	3813.19		5000.89	4.53	
175c6d6v	175	5	6	3600	2685.38	1304.21	5291.62	5.38	5.38
175c6d8v	175	5	8	3600	2708.57	1089.71	6034.21	9.94	9.94
175c8d4v	175	7	4	3600	3731.42		5747.72	5.29	
175c8d6v	175	7	6	3600	2836.71	961.12	5914.00	5.00	5.00
175c8d8v	175	7	8	3600	3764.96		6766.54	8.39	

TABLE 4.3: Results on TZK instances with 100 to 175 customers.

instance	n	f	n_K	$T_1 = T_{23}$	t_1	t_{23}	BKS	$\%^1$	$\%^{23}$
a1	48	2	6	3600	5.77	3600.22	1179.79	7.70	6.91
d1	48	3	5	3600	7.93	3600.02	1059.42	7.67	6.44
a2	48	4	4	3600	0.76	3600.04	997.94	6.96	5.71
g1	72	4	5	5400	60.28	5400.02	1181.13	4.77	4.59
j1	72	5	4	5400	33.40	5400.08	1115.77	5.46	4.46
g2	72	6	4	5400	7.85	5400.06	1152.92	5.76	4.65

TABLE 4.4: Results on CCL instances with 48 to 72 customers.

CCL case *a1* and the TZK case *50c3d6v*, while 4.13 compares instances *g1* and *75c5d4v*. Once more, the root separation procedure NODESEPARATION can be given credit for most of the final gap, as the gap difference is always less than 1.3%. This holds even more for the big-sized CCL instances, where the improvement due to the root refinement phase -the only responsible for the gap difference- is always under 0.5%. The results on

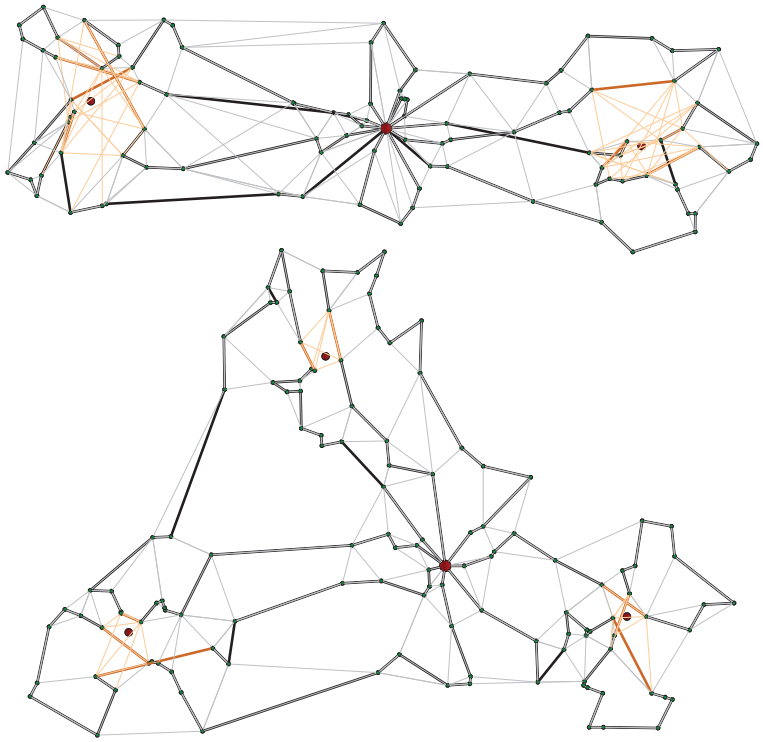


FIGURE 4.10: BKS to the TZK instances *100c3d7v* (top) and *125c4d5v*.

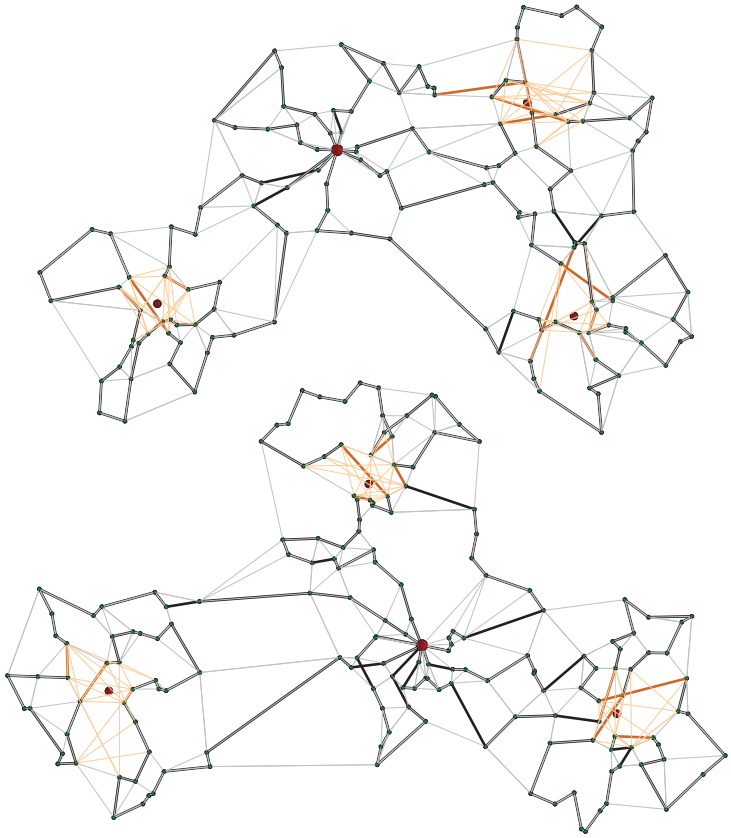


FIGURE 4.11: BKS to the TZK instances *150c4d5v* (top) and *175c4d6v*.

instance	n	f	n_K	T_{12}	t_1	t_2	BKS	$\%^1$	$\%^2$
b1	96	2	4	3600	115.95	54.56	1217.07	3.38	3.37
e1	96	3	5	3600	59.40	25.51	1309.12	2.48	2.05
b2	96	4	4	3600	56.38	28.79	1291.18	3.88	3.87
h1	144	4	4	3600	1487.26	588.42	1545.50	4.26	4.25
k1	144	5	4	3600	622.59	187.87	1573.20	3.89	3.89
c2	144	4	4	3600	1016.70	195.45	1715.59	3.86	3.86
h2	144	6	4	3600	1160.07	133.82	1575.27	4.34	4.34
c1	192	2	5	3600	2755.35	871.72	1866.75	3.61	3.55
f1	192	3	4	3600	3543.95	61.81	1570.40	2.36	2.36
d2	192	4	3	3600	3808.18		1854.03	3.92	
i1	216	4	4	7200	6090.99	1823.72	1922.17	2.85	2.85
l1	216	5	4	7200	7751.10		1863.27	3.23	
i2	216	6	3	7200	7341.02		1919.73	3.89	
e2	240	4	3	7200	7394.10		1916.67	4.43	
f2	288	4	3	7200	7201.17		2230.30	7.28	
j2	288	6	3	7200	8472.73		2247.68	3.13	

TABLE 4.5: Results on CCL instances with 96 to 288 customers.

instances	$\%^1$	$\underline{\%^1}$
50c3d_v	8.18	11.51
50c5d_v	7.63	9.94
50c7d_v	5.64	6.81
100c3d_v	8.48	9.75
100c5d_v	8.56	10.11
100c7d_v	8.71	16.26
125c4d_v	7.59	8.29
125c6d_v	5.54	5.82
125c8d_v	5.33	6.18
150c4d_v	6.40	14.90
150c6d_v	6.01	10.11
150c8d_v	6.21	6.55
CCL with n=48	7.44	13.37
CCL with n=72	5.33	5.63
CCL with n=96	3.25	10.32
CCL with n=144	4.09	19.02

TABLE 4.6: Comparison (average values on some sample instance families) of the gap after phase 1 with and without including the valid inequalities.

big-sized CCL instances are even better than those seen in table 4.3: the initial gap is under 5% in 15 out of 16 cases. However, given this fact and considering that n grows up to 288 (instances *f2* and *j2*), one may conclude that bigger CCL instances are quite easy ones. This is well-shown, for instance, by figure 4.14 for what concerns CCL case *j2*: apart from a dense core, the outer zones appears to be easy to deal with from a decisional point of view, and since their weight in terms of costs is considerable, they strongly contribute to tighten the root gap.

Finally, table 4.6 well shows how the insertion of the valid inequalities in the separation procedure NODESEPARATION is effective. As one can see, for some of the sample

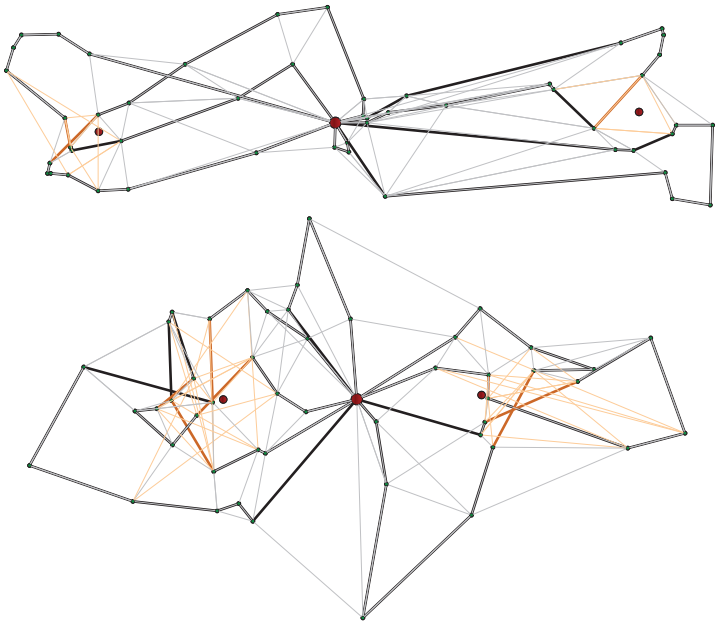


FIGURE 4.12: Comparison of the BKS to *50c3d6v* (top) and *a1*.

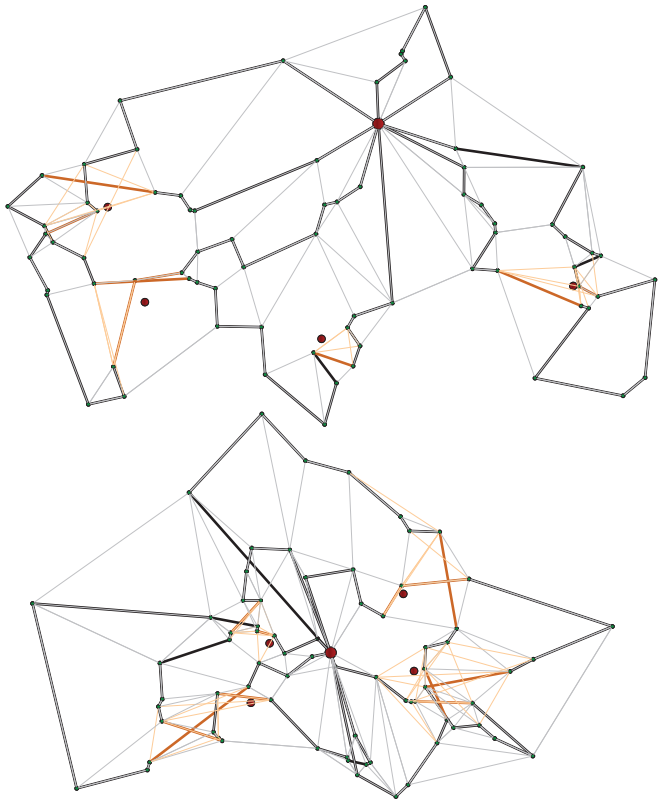


FIGURE 4.13: Comparison of the BKS to *75c5d4v* (top) and CCL instance *g1*.

instance family the gap worsens considerably.

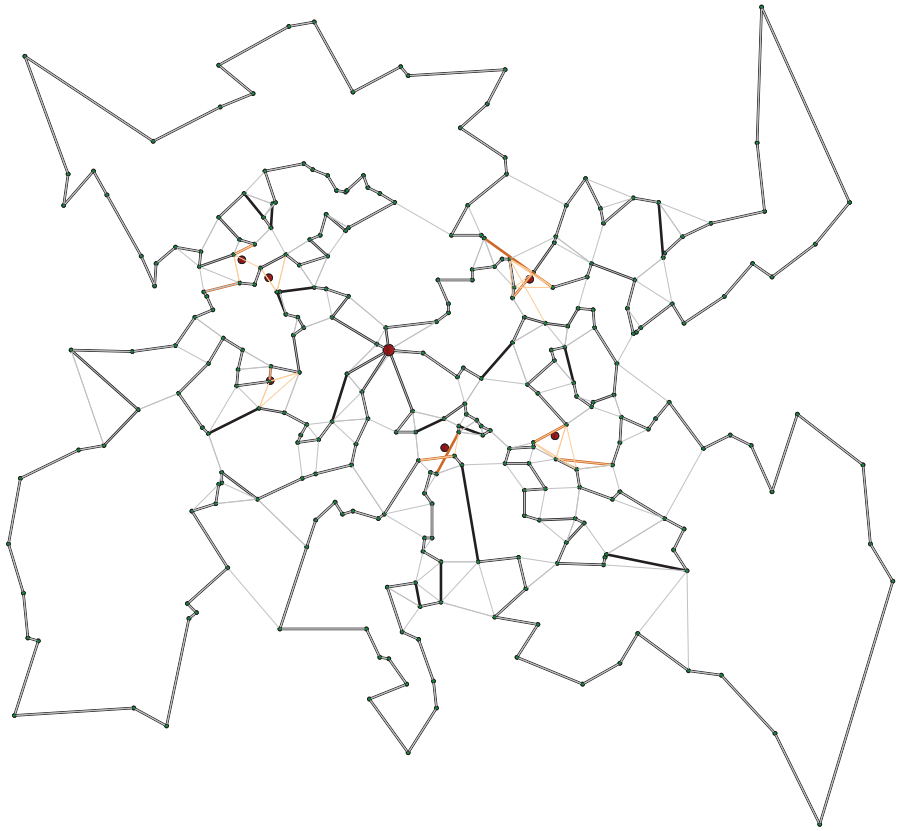


FIGURE 4.14: BKS to the CCL instance *j2*. The trace of the fractionary solution at the end of phase 1 well shows how variables concerning the outer zones practically have integer value even in the linear relaxation solution.

4.3 Conclusions and Perspectives

In this chapter, we have presented two Branch & Cut algorithms for the VRPIRF. However, the first one can be considered to all purposes preparatory to the second, which is far more clever in two respects: i.e. the MILP model and the separation techniques and strategy. The second one has shown an interesting behavior on most of the benchmark instances taken from the literature, and therefore it can be considered very promising. To improve the algorithm based on $\mathcal{M}_{RA}^{BC}$ and NODESEPARATION, two main ideas appear to be viable. The first one consists in enhancing the separation procedure with other cuts or valid inequalities taken from the literature. The second one is to study more specific, problem-tailored cuts that take into account, for instance, the arrival times. The latter stream could even lead to a more radical approach. Since many benchmark instances in the literature are symmetric, one could consider to reformulate the VRPIRF in a symmetric fashion. Arrival times imply the notion of predecessor and therefore could no longer be used, but it would be probably easier to design ad-hoc inequalities to prevent vehicle-related issues. Also, using the CVRPSEP routines would be more effective, since they are designed for a symmetric context: the graph transformations to

tackle this bias would not be required anymore, and their side-effects on the tightness of the lower bound would probably be overcome.

Branch & Cut approaches represent only one of the possible types of exact methods to deal with the VRPIRF – and in general with Combinatorial Optimization problems. Branch & Price algorithms are another possible solving strategy. Since they rely on extended MILP formulations, they often call for the design of a *Column Generation* framework. In order to broaden our study of exact approaches to the VRPIRF, we also designed and developped Branch & Price algorithms to solve it. They are presented and discussed in the next Chapter.

Chapter 5

Branch & Price algorithms for the Vehicle Routing Problem with Intermediate Replenishment Facilities

5.1 A Branch & Price Algorithm for the VRPIRF

In the first part of this chapter, we propose a third approach to VRPIRF, a Branch & Price algorithm based on an extended formulation $\mathcal{M}^{BP}$ which makes use of route variables instead of arc-flow ones. Section 5.1.1 is devoted to the description of the MILP model, while sections 5.1.2 and 5.1.3 go into the problem-specific details of the new algorithm, and section 5.1.4 shows the computational assessment. Later, section 5.2 introduces a second Branch & Price approach. Finally, 5.3 draws the conclusions of the chapter.

5.1.1 A Set-Partitioning Formulation

The support graph of model $\mathcal{M}^{BP}$ is the same seen in section 4.1.1: we reintroduce both the facility nodes' set, so as to have $V = C \cup \{\Delta\} \cup F$, and the index set K in order to distinguish the vehicles. The arc set $A = V \times C \cup C \times V$ composed of base arcs only is restored. In addition, we define the set $\mathscr{R}$ of all possible *feasible* (i.e. compliant to load and duration limit) service routes and we characterize its generic element $r \in \mathscr{R}$ by means of the following binary coefficients:

- $a_r^i = 1 \Leftrightarrow r$ visits client i;
- $e_r'^p = 1 \Leftrightarrow r$ starts at facility p, $e_r''^p = 1 \Leftrightarrow r$ ends at facility p;
- $e_r'^\Delta = 1 \Leftrightarrow r$ starts at Δ, $e_r''^\Delta = 1 \Leftrightarrow r$ ends at Δ;
- $b_r'^s = 1 \Leftrightarrow r$ has its endpoint in set $s \subseteq V\backslash\{\Delta\}$ and its starting point in $V\backslash s$;
- $b_r''^s = 1 \Leftrightarrow r$ has its starting point in $s \subseteq V\backslash\{\Delta\}$ and its endpoint in $V\backslash s$;
- $b_r^{ij} = 1 \Leftrightarrow r$ visits i and j in sequence, i.e. $ij \in r$

We can define the load, the cost and the duration of a route as follows:

$$q_r = \sum_{i \in C} a_r^i \cdot q_i \tag{5.1}$$

$$c_r = \sum_{ij \in A} b_r^{ij} \cdot d_{ij} \tag{5.2}$$

$$t_r = \sum_{ij \in A} b_r^{ij} \cdot \tau_{ij} + \sum_{i \in C} a_r^i \cdot \tau_i + \tfrac{1}{2}(e_r'^\Delta + e_r''^\Delta) \cdot \tau_\Delta + \sum_{p \in F} \tfrac{1}{2}(e_r'^p + e_r''^p) \cdot \tau_p \tag{5.3}$$

Note that (5.1) and (5.3) allow to define $\mathscr{R}$ as the set of all routes r s.t. $q_r \leq Q$ and $t_r \leq T$. The decision variables are:

- binary *route variables* x_r^k, $r \in \mathscr{R}$, $k \in K$, $x_r^k = 1 \Leftrightarrow$ route r is in solution and performed by vehicle k;
- binary *usage variables* $\widetilde{x}^k$, $k \in K$, $\widetilde{x}^k = 1 \Leftrightarrow$ vehicle k is used;
- binary *activity variables* y_p^k, $k \in K$, $p \in F$, defined like in 4.1.2.

Let us finally introduce symbol $\mathcal{S}_p = \{s \subseteq C \cup F : p \in s\}$ to denote the collection of subsets of nodes that contain the facility p but not the depot.
The first formulation of model $\mathcal{M}^{BP}$, which we denote by $\mathcal{M}^{BP'}$, is:

$(\mathcal{M}^{BP'})$

$$\min \quad \sum_{k \in K} \sum_{r \in \mathscr{R}} c_r x_r^k$$

$$\text{s.t.} \quad \sum_{r \in \mathscr{R}} \sum_{k \in K} a_r^i \; x_r^k = 1 \qquad \forall i \in C \tag{5.4}$$

$$e_r'^p \; x_r^k \leq y_p^k \qquad \forall k \in K, r \in \mathscr{R}, p \in F \tag{5.5}$$

$$\sum_{r \in \mathscr{R}} (e_r'^p - e_r''^p) \; x_r^k = 0 \qquad \forall k \in K, p \in F$$

$$\sum_{r \in \mathscr{R}} t_r \; x_r^k \leq T \; \widetilde{x}^k \qquad \forall k \in K \tag{5.6}$$

$$y_p^k \leq \widetilde{x}^k \qquad \forall k \in K, p \in F$$

$$\sum_{r \in \mathscr{R}} e_r'^\Delta \; x_r^k = \widetilde{x}^k \qquad \forall k \in K$$

$$\sum_{r\in\mathscr{R}} e'^{\Delta}_r \; x^k_r = \widetilde{x}^k \qquad \forall k \in K$$

$$\sum_{r\in\mathscr{R}} b'^{s}_r \; x^k_r \geq y^k_p \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p$$

$$\sum_{r\in\mathscr{R}} b''^{s}_r \; x^k_r \geq y^k_p \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p$$

$$\widetilde{x}^k, x^k_r, y^k_p \in \{0,1\} \qquad \forall k \in K, r \in \mathscr{R}, p \in F$$

Some simple considerations can help improving the model before describing it into details. Constraints (5.5) are used to activate variable y^k_p when vehicle k performs some route starting from p. As one can observe, they are in exponential number. In order to reduce their number, we first replace each of such constraints with the following set:

$$(\forall k \in K, r \in \mathscr{R}, p \in F)\ (\forall i \in C) \quad a^i_r \; e'^{p}_r \; x^k_r \leq y^k_p$$

which holds if $\mathscr{R}$ contains only non-empty routes, i.e. if $(\forall r \in \mathscr{R})\ (\exists i \in C)\ a^i_r = 1$. Then, by fixing i, k and p, we can replace the above constraints with their surrogates:

$$(\forall k \in K, p \in F)\ (\forall i \in C) \quad \sum_{r\in\mathscr{R}} a^i_r \; e'^{p}_r \; x^k_r \leq M_p \; y^k_p$$

where M_p must be large enough to be an upper bound on the left-hand side, i.e. theoretically $M_p = \mathcal{O}(|\{r \in \mathscr{R} : e'^{p}_r = 1\}|)$. In order to define M_p, we first observe that due to constraints (5.4) (which concern the visits to customers) we have

$$\sum_{r\in\mathscr{R}} a^i_r \; e'^{p}_r \; x^k_r \leq \sum_{r\in\mathscr{R}} a^i_r \; x^k_r \leq 1$$

so $M_p = 1$ and therefore

$$\sum_{r\in\mathscr{R}} a^i_r \; e'^{p}_r \; x^k_r \leq y^k_p$$

Secondly, we can relax constraints (5.4) in a set-covering fashion:

$$\sum_{r\in\mathscr{R}} \sum_{k\in K} a^i_r \; x^k_r \geq 1$$

In fact, assuming that the triangle inequality holds in the distance matrix, there always exists an optimal solution where each client is visited by exactly one route. Finally, constraints (5.6), which enforce the maximum shift length, can be rewritten in a symmetry-breaking fashion. In order to do so, we add real positive *duration* variables $\upsilon^k \in \mathbb{R}_+$,

$k \in K$ and replace these constraints with:

$$\sum_{r \in \mathscr{R}} t_r \; x_r^k \leq \upsilon^k \qquad \forall k \in K$$
$$\upsilon^k \leq T \; \widetilde{x}^k \qquad \forall k \in K$$
$$\widetilde{x}^k \leq \widetilde{x}^{k+1} \qquad \forall k \in K \backslash \{n_K\}$$
$$\upsilon^k \leq \upsilon^{k+1} \qquad \forall k \in K \backslash \{n_K\}$$

Hence we obtain the model $\mathcal{M}^{BP}$, which follows:

$(\mathcal{M}^{BP})$

$$\min \quad \sum_{k \in K} \sum_{r \in \mathscr{R}} c_r x_r^k \tag{5.7}$$
$$\text{s.t.} \quad \sum_{r \in \mathscr{R}} \sum_{k \in K} a_r^i \; x_r^k \geq 1 \qquad \forall i \in C \tag{5.8}$$
$$\sum_{r \in \mathscr{R}} a_r^i \; e_r'^p \; x_r^k \leq y_p^k \qquad \forall k \in K, p \in F, i \in C \tag{5.9}$$
$$\sum_{r \in \mathscr{R}} (e_r'^p - e_r''^p) \; x_r^k = 0 \qquad \forall k \in K, p \in F \tag{5.10}$$
$$\sum_{r \in \mathscr{R}} t_r \; x_r^k \leq \upsilon^k \qquad \forall k \in K \tag{5.11}$$
$$\upsilon^k \leq T \; \widetilde{x}^k \qquad \forall k \in K \tag{5.12}$$
$$\widetilde{x}^k \leq \widetilde{x}^{k+1} \qquad \forall k \in K \backslash \{n_K\} \tag{5.13}$$
$$\upsilon^k \leq \upsilon^{k+1} \qquad \forall k \in K \backslash \{n_K\} \tag{5.14}$$
$$y_p^k \leq \widetilde{x}^k \qquad \forall k \in K, p \in F \tag{5.15}$$
$$\sum_{r \in \mathscr{R}} e_r'^{\Delta} \; x_r^k = \widetilde{x}^k \qquad \forall k \in K \tag{5.16}$$
$$\sum_{r \in \mathscr{R}} e_r''^{\Delta} \; x_r^k = \widetilde{x}^k \qquad \forall k \in K \tag{5.17}$$
$$\sum_{r \in \mathscr{R}} b_r'^s \; x_r^k \geq y_p^k \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p \tag{5.18}$$
$$\sum_{r \in \mathscr{R}} b_r''^s \; x_r^k \geq y_p^k \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p \tag{5.19}$$
$$\widetilde{x}^k, x_r^k, y_p^k \in \{0,1\} \qquad \forall k \in K, r \in \mathscr{R}, p \in F$$
$$\upsilon^k \geq 0 \qquad \forall k \in K$$

Constraints (5.8) ensure that each client is visited by at least one service route. Relations (5.9) activate variable y_p^k when vehicle k performs some route starting from p. Note that in spite of the set-covering form of constraints (5.8), relations (5.9) still forbid a client to be visited by two or more routes starting from the same facility. Constraints (5.10) impose a balance between routes of k that start at p, and those ending at it.

The maximum shift length constraint is imposed on every *used* vehicle by means of constraints (5.11)–(5.14), which also force vehicles with higher indices to be used first and be assigned to the longer rotations, in a symmetry-breaking spirit. If $\widetilde{x}^k = 0$, i.e. vehicle k is not used, then the activity of k on facilities must be forbidden: this is enforced by (5.15). Constraints (5.16) and (5.17) impose that either k is not used, or it must leave and enter the depot exactly once. Finally, relations (5.18) and (5.19) are connectivity constraints for vehicle k from depot to facility p: we call them *fore-* and *back-connectivity* constraints, respectively. They state that if k has some activity at p, there must be a continuous sequence of routes performed by k from Δ to p, and one back.
Fore- and back-connectivity constraints can be reformulated as follows:

$$\sum_{r\in\mathscr{R}} \Big(\sum_{\substack{i\in s \\ j\notin s}} (b_r^{ij} - b_r^{ji}) \Big)\ x_r^k \geq y_p^k \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p$$

$$\sum_{r\in\mathscr{R}} \Big(\sum_{\substack{i\in s \\ j\notin s}} (b_r^{ji} - b_r^{ij}) \Big)\ x_r^k \geq y_p^k \qquad \forall k \in K, p \in F, s \in \mathcal{S}_p$$

as $b_r'^s$ and $b_r''^s$ are easily seen to be equivalent to $\sum_{i\in s, j\notin s}(b_r^{ij} - b_r^{ji})$ and $\sum_{i\in s, j\notin s}(b_r^{ji} - b_r^{ij})$, respectively. An even better rewriting of the connectivity constraints can be achieved by redefining the collection $\mathcal{S}_p$ of subsets of nodes (along with its complement $\overline{\mathcal{S}_p}$) as follows:

$$\mathcal{S}_p = \{s : s \subseteq F, p \in s\} \quad ; \quad \overline{\mathcal{S}_p} = \{s : s \subset F, p \notin s\} = \mathcal{P}(F)\backslash\mathcal{S}_p \tag{5.20}$$

i.e. by redefining $\mathcal{S}_p$ on the *facility graph*. This makes the number of connectivity constraints become much lower, as it is now $\mathcal{O}(|K||F||\mathcal{P}(F)|)$ instead of $\mathcal{O}(|K||F||\mathcal{P}(V)|)$. This can have important consequences from an algorithmic point of view, since for *relatively small* values of $|K|$ and $|F|$ one can consider to introduce the whole set of connectivity constraints statically, instead of generating them dynamically.
Constraints (5.18) and (5.19) can also be written as follows:

$$\sum_{r\in\mathscr{R}} b_r'^s\ x_r^k \geq y_p^k \qquad \forall k \in K, s \subseteq F, p \in s \qquad \delta'_{kps} \tag{5.21}$$

$$\sum_{r\in\mathscr{R}} b_r''^s\ x_r^k \geq y_p^k \qquad \forall k \in K, s \subseteq F, p \in s \qquad \delta''_{kps} \tag{5.22}$$

This slight modification in the constraint indexing can help simplify the expression of the reduced cost of the route variables, as it will be shown in the following.

One last thing that is worth mentioning, we note that $b_r'^s$ and $b_r''^s$ can be expressed as:

$$b_r'^s = (e_r'^{\Delta} + \sum_{p \notin s} e_r'^p)(\sum_{p \in s} e_r''^p) \quad ; \quad b_r''^s = (\sum_{p \in s} e_r'^p)(e_r''^{\Delta} + \sum_{p \notin s} e_r''^p) \tag{5.23}$$

5.1.2 Outline of the Branch&Price Algorithm

In the following we present a solution approach based on the MILP model $\mathcal{M}^{BP}$. This model is in the form of Gilmore and Gomory (see [Gilmore and Gomory, 1963]), as it contains a class of integer variables, the route variables $x_r^k \in \mathscr{R}$, whose number $|\mathscr{R}| = \mathcal{O}(|V|!)$ is exponential in the size of the problem instance. In these conditions, it is hard even to find a solution to the LP relaxation of the integer problem – which is a crucial point, since computing a lower bound for each node of the Branch&Bound tree is required to decide whether it must be pruned or branched on. *Column Generation (CG)* approaches are useful to compute the value of such LP relaxations.

5.1.2.1 Column Generation

The general Column Generation scheme to find the optimal solution to large-scale LP problems consists of the following steps. Although the described procedure has a more general validity, we explicitly refer to our case for the sake of simplicity:

1. initialize the problem with a small subset of route variables $x_r, r \in \overline{\mathscr{R}} \subset \mathscr{R}$, so as to obtain the so-called *Restricted Master Problem (RMP)*, to distinguish it from the *Master Problem (MP)*, i.e. the problem with the complete set of route variables. Set $\overline{\mathscr{R}}$ is commonly initialized with artificial variables, or trivial route variables (like e.g. one-customer routes), or a set of routes variables determined heuristically;
2. find the optimal solution $\mathbf{s}^\star$ of the RMP. Let $\overline{\mathscr{R}^\star} \subset \overline{\mathscr{R}}$ be the set of routes having $x_r > 0$ in this solution. $\mathbf{s}^\star$ is feasible w.r.t. the MP (unless it makes use of some artificial variables) as it is equivalent to picking the same routes $\overline{\mathscr{R}^\star}$ over the whole set $\mathscr{R}$ and setting $x_r = 0$ for every $r \in \mathscr{R} \backslash \overline{\mathscr{R}^\star}$. The difference holds in that if we had the whole route variable set $\mathscr{R}$, the optimal solution of the RMP would also be globally optimal, since no more negative reduced cost columns could be inserted in the current base, whereas the fact that $|\overline{\mathscr{R}}| < |\mathscr{R}|$ (and, in general, $|\overline{\mathscr{R}}| \ll |\mathscr{R}|$) implies that there might be further negative reduced cost variables in $\mathscr{R} \backslash \overline{\mathscr{R}}$ (we suppose to be dealing with a minimization problem);
3. recover the current dual solution $\boldsymbol{\pi}^\star$ of the RMP that allows to compute the reduced cost $\overline{c_r}$ of route variables x_r;

4. determine the existence of possible negative reduced cost variables in $\mathscr{R}\backslash\overline{\mathscr{R}}$. This is done *implicitly* by solving a secondary minimization problem, called the *Slave Problem* or *Subproblem* since it is subordinate to the Master Problem. Each solution of the Slave Problem corresponds to a route in $\mathscr{R}$: the cost parameters of the problem are chosen according to $\boldsymbol{\pi}^\star$ in such a way to assign each solution of the Slave Problem a value which equals the reduced cost of the matching MP variable. The Slave Problem is often referred to also as the *Oracle* or the *Pricing Problem* (which we will denote by PP) for its specific function. The search of routes $r \in \mathscr{R}\backslash\overline{\mathscr{R}}$ with $\overline{c_r} < 0$ is implicit in the sense that if the optimal solution of the Slave Problem has nonnegative value, then no such route exists, hence the optimality of the current (R)MP solution is proven and the procedure can end. Otherwise, one or more MP negative reduced cost variables exist.
5. decide how many and which of these variables are to be added to the restricted set $x_r : r \in \overline{\mathscr{R}}$, and repeat from point 2.

If the solution of the PP has a negative value, in most of the cases there are many negative reduced cost variables in $\mathscr{R}\backslash\overline{\mathscr{R}}$, but not all of them are worth being added to the RMP. Hence, point 5 actually requires the tuning of a variable selection strategy.
Column Generation represents one of the most investigated techniques of the last decades. The reader can refer to excellent introductory works like [Vanderbeck and Wolsey, 2009], [Lübbecke and Desrosiers, 2002] or [Feillet, 2010] to have an exhaustive insight to the subject, or to [du Merle et al., 1997] to be introduced to some known CG issues, or [Moungla et al., 2010] for an example of variable selection strategy.
A Column Generation tool to compute the lower bound at each node of the Branch& Bound tree, along with a *Branching policy* to choose the fractional-value variables to branch on in the same node, are the key elements of a *Branch&Price exact algorithm* to solve large-scale MILP problems.

5.1.2.2 Reduced costs

The first step is to determine the reduced cost of route variables, before deciding how to solve the PP. From the Linear Programming theory, we know that the reduced cost $\overline{c_j}$ of a variable x_j is equal to $c_j - \boldsymbol{\pi}^T A_j$, where A_j and $\boldsymbol{\pi}$ are the j-th column of the constraints matrix and the dual variables vector, respectively. Hence, the expression of the reduced cost of variable x_r^k is given by the cost c_r minus the sum, over all constraints, of the dual variable associated with each constraint, weighted by the coefficient of x_r^k in the constraint itself. Let us name the involved dual variables as follows:

► $\alpha_i \geq 0$ the variable associated with the constraint (5.8) related to $i \in C$,

- $\varphi_{kpi} \leq 0$ that associated with the constraint (5.9) related to $k \in K$, $p \in F$ and $i \in C$,
- $\theta_{kp} \geqslant 0$ the variables related constraint (5.10) of $k \in K$, $p \in F$,
- $\beta_k \leq 0$ that related to constraint (5.11) associated with vehicle k,
- $\mu'_k \geqslant 0$, $\mu''_k \geqslant 0$ the dual variables associated with constraints (5.16) (5.17) connected to vehicle k, and finally
- $\delta'_{kps} \geq 0$, $\delta''_{kps} \geq 0$ those related to constraints (5.18), (5.19), $k \in K$, $p \in F$ and $i \in C$.

The reduced cost of variable x_r^k is:

$$\begin{aligned}\overline{c}_r^k = c_r &- \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \sum_{p \in F} e_r'^p \cdot \varphi_{kpi}^\star) - \sum_{p \in F} (e_r'^p - e_r''^p) \cdot \theta_{kp}^\star - t_r \cdot \beta_k^\star \\ &- e_r'^\Delta \cdot \mu_k'^\star - e_r''^\Delta \cdot \mu_k''^\star - \sum_{s \subseteq F} \sum_{p \in s} (b_r'^s \cdot \delta_{kps}'^\star + b_r''^s \cdot \delta_{kps}''^\star)\end{aligned}$$

where $(\alpha_i^\star,\ \varphi_{kpi}^\star,\ \theta_{kp}^\star,\ \beta_k^\star,\ \gamma_k^\star,\ \varsigma_k'^\star,\ \varsigma_k''^\star,\ \sigma_{kp}^\star,\ \mu_k'^\star,\ \mu_k''^\star,\ \delta_{kps}'^\star,\ \delta_{kps}''^\star)$ is the dual solution corresponding to the current solution of the RMP. The above expression of $\overline{c}_r^k$ makes use of the indexing used in (5.21) and (5.22) instead of that of (5.18) and (5.19). It is important to notice that since route variables are indexed by vehicle, we cannot avoid to solve a distinct PP for each vehicle $k \in K$. The expression of $\overline{c}_r^k$ becomes:

$$\begin{aligned}\overline{c}_r^k = \quad & \sum_{ij \in A} b_r^{ij} \cdot d_{ij} - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \sum_{p \in F} e_r'^p \cdot \varphi_{kpi}^\star) \\ & - \left(\sum_{ij \in A} b_r^{ij} \cdot \tau_{ij} + \sum_{i \in C} a_r^i \cdot \tau_i + \sum_{p \in F} \tfrac{1}{2} (e_r'^p + e_r''^p) \cdot \tau_p \right) \cdot \beta_k^\star \\ & - e_r'^\Delta \cdot \mu_k'^\star - e_r''^\Delta \cdot \mu_k''^\star - \sum_{p \in F} (e_r'^p - e_r''^p) \cdot \theta_{kp}^\star \\ & - \sum_{s \subseteq F} \sum_{p \in s} \left((e_r'^\Delta + \sum_{q \notin s} e_r'^q)(\sum_{q \in s} e_r''^q) \cdot \delta_{kps}'^\star + (\sum_{q \in s} e_r'^q)(e_r''^\Delta + \sum_{q \notin s} e_r''^q) \cdot \delta_{kps}''^\star \right) \qquad (5.24) \\ = \quad & \sum_{ij \in A} b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star) - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star + \sum_{p \in F} e_r'^p \cdot \varphi_{kpi}^\star) \\ & - e_r'^\Delta \cdot \mu_k'^\star - e_r''^\Delta \cdot \mu_k''^\star - \sum_{p \in F} \tfrac{1}{2} (e_r'^p + e_r''^p) \cdot \tau_p \cdot \beta_k^\star - \sum_{p \in F} (e_r'^p - e_r''^p) \cdot \theta_{kp}^\star \\ & - \sum_{s \subseteq F} \sum_{p \in s} \left((e_r'^\Delta + \sum_{q \notin s} e_r'^q)(\sum_{q \in s} e_r''^q) \cdot \delta_{kps}'^\star + (\sum_{q \in s} e_r'^q)(e_r''^\Delta + \sum_{q \notin s} e_r''^q) \cdot \delta_{kps}''^\star \right)\end{aligned}$$

which can be seen as the cost of route r on a graph which is identical to the initial support graph (see again 4.1.1) but with altered costs on arcs and additional costs on nodes: the new costs depend on the dual variables associated with the constraints that involve route variables, and can be seen as *prizes* or *penalties* according to their contribution to the reduced cost $\overline{c}_r^k$. More in detail, such a weight turns out to be a prize or a penalty

depending on whether the insertion of r in the MP simplifies the satisfiability of the constraints in which it is involved, or pushes them closer to their boundaries. Here are some examples:

- term $b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star)$ is equivalent to setting a modified cost $d_{ij} - \tau_{ij} \cdot \beta_k^\star$ on arc ij;
- term $\alpha_i^\star + \tau_i \cdot \beta_k^\star + \sum_{p \in F} e_r'^p \cdot \varphi_{kpi}^\star$ is the weight assigned to client i;
- every term which is multiplied by $e_r'^p$ or $e_r''^p$ must be added when p is the starting or ending point of r, respectively. The same applies to terms with $e_r'^\Delta$ and $e_r''^\Delta$ w.r.t. Δ.

The weight of a node, however, can be conveniently added to the weight of each of its ingoing or outgoing arc, in order to have again a problem with costs on arcs only.
If we do so, solving the PP will amount to finding a shortest (i.e. with least $\overline{c}_r^k$ value) route r on the modified support graph. Since the solution routes must respect both a load and a maximum length constraint, this calls for solving an *Elementary Shortest Path Problem with Resource Constraints (ESPPRC)*. We will have a brief recall of ESPPRC later in section 5.1.2.3.
As one can observe, the weight of client node $i \in C$ in (5.24), $-a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star + \sum_{p \in F} e_r'^p \cdot \varphi_{kpi}^\star)$, depends on the starting point of the route r, which must be imposed. As a consequence, at each CG iteration we have to solve up to $|K|(|F|+1)$ PPs, i.e. one per vehicle and per starting facility/depot. On the other hand, the expression of $\overline{c}_r^k$ can be simplified, as it depends on whether the starting point is Δ or a facility $g \in F$.
In the first case, $e_r'^\Delta = 1$ and $e_r'^p = 0 \ \forall p \in F$, therefore we have:

$$\begin{aligned}
\overline{c}_r^k = \quad & \sum_{ij \in A} b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star) - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star) - \mu_k'^\star - e_r''^\Delta \cdot \mu_k''^\star \\
& - \sum_{p \in F} e_r''^p \cdot (\tfrac{1}{2} \cdot \tau_p \cdot \beta_k^\star - \theta_{kp}^\star) - \sum_{s \subseteq F} \sum_{p \in s} \delta_{kps}'^\star \cdot \sum_{q \in s} e_r''^q \\
= \quad & - \mu_k'^\star + \sum_{ij \in A} b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star) - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star) - e_r''^\Delta \cdot \mu_k''^\star \\
& - \sum_{p \in F} e_r''^p \cdot (\tfrac{1}{2} \cdot \tau_p \cdot \beta_k^\star - \theta_{kp}^\star) - \sum_{p \in F} e_r''^p \cdot \sum_{s \in \mathcal{S}_p} \sum_{q \in s} \delta_{kqs}'^\star
\end{aligned} \tag{5.25}$$

To verify that $\sum_{s \subseteq F} \sum_{p \in s} \delta_{kps}'^\star \cdot \sum_{q \in s} e_r''^q = \sum_{p \in F} e_r''^p \cdot \sum_{s \in \mathcal{S}_p} \sum_{q \in s} \delta_{kqs}'^\star$, one can take any $s \subseteq F$ and $p \in s$ and put a weight $\delta_{kps}'^\star$ on each $q \in s$; then, the sum on each $s \subseteq F$ and $p \in s$ leads to assign each $p \in F$ the sum of all the $\delta_{kps}'^\star$ of each $s \subseteq F$ it belongs to:

$$\sum_{\substack{s \subseteq F \\ p \in s}} \delta_{kps}'^\star \cdot \sum_{q \in s} e_r''^q = \sum_{\substack{s \subseteq F \\ p \in s \\ q \in s}} \delta_{kps}'^\star \cdot e_r''^q = \sum_{\substack{s \subseteq F \\ p,q \in s}} \delta_{kps}'^\star \cdot e_r''^q = \sum_{\substack{q \in F \\ s \in \mathcal{S}_q}} e_r''^q \cdot \sum_{p \in s} \delta_{kps}'^\star = \sum_{q \in F} e_r''^q \cdot \sum_{\substack{s \in \mathcal{S}_q \\ p \in s}} \delta_{kps}'^\star$$

We can now construct the modified support graph $G' = (V', A')$ for the PP:

1. the node set $V' = V \cup \{\Delta', N\}$ includes a copy Δ' of Δ and a dummy arrival node N;
2. the arc set $A' = A \cup \bigcup_{i \in C} \{(\Delta', i)\} \cup \left((\Delta, N) \cup \bigcup_{p \in F} \{(p, N)\} \right)$ is enhanced with arcs from Δ' to each client, and from each possible *real* arrival point to N;
3. the weights on nodes are the following:

$$w_v^\star = \begin{cases} -\alpha_i^\star - \tau_i \cdot \beta_k^\star & (v \equiv i \in C) \\ -\frac{1}{2} \cdot \tau_p \cdot \beta_k^\star + \theta_{kp}^\star - \sum_{s \in \mathcal{S}_p} \sum_{q \in s} \delta_{kqs}'^\star & (v \equiv p \in F) \\ -\mu_k''^\star & (v \equiv \Delta) \end{cases}$$

4. the weights of the arcs $ij \in A'$ are:

$$w_{ij}^\star = \begin{cases} d_{\Delta j} - \tau_{\Delta j} \cdot \beta_k^\star + \frac{1}{2} \cdot w_j^\star - \mu_k'^\star & (i \equiv \Delta') \\ d_{ij} - \tau_{ij} \cdot \beta_k^\star + \frac{1}{2} \cdot (w_i^\star + w_j^\star) & (ij \in C \times C) \\ d_{ip} - \tau_{ip} \cdot \beta_k^\star + \frac{1}{2} \cdot w_i^\star + w_p^\star & (i \in C, j \equiv p \in F) \\ d_{i\Delta} - \tau_{i\Delta} \cdot \beta_k^\star + \frac{1}{2} \cdot w_i^\star + w_\Delta^\star & (i \in C, j \equiv \Delta) \\ 0 & (j \equiv N) \end{cases}$$

The PP is then solved as an ESPPRC on (V', A') from Δ' to N. For each possible ending point v, the least reduced cost path from Δ' to v is added to the RMP.
When the starting point is a facility $g \in F$, we have $e_r'^g = 1$ and $(\forall q \in F \backslash \{g\})$ $e_r'^q = e_r'^\Delta = 0$, therefore:

$$\begin{aligned} \overline{c}_r^k = & \sum_{ij \in A} b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star) - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star + \varphi_{kgi}^\star) - \tfrac{1}{2} \cdot \tau_g \cdot \beta_k^\star \\ & - \sum_{p \in F} e_r''^p \cdot \tfrac{1}{2} \cdot \tau_p \cdot \beta_k^\star - e_r''^\Delta \cdot \mu_k''^\star - \theta_{kg}^\star + \sum_{p \in F} e_r''^p \cdot \theta_{kp}^\star \\ & - \sum_{\substack{s \subseteq F: \\ g \notin s}} \sum_{p \in s} \left((\sum_{q \in s} e_r''^q) \cdot \delta_{kps}'^\star \right) - \sum_{\substack{s \subseteq F: \\ g \in s}} \sum_{p \in s} \left((e_r''^\Delta + \sum_{q \notin s} e_r''^q) \cdot \delta_{kps}''^\star \right) \\ = & - (\theta_{kg}^\star + \tfrac{1}{2} \cdot \tau_g \cdot \beta_k^\star) + \sum_{ij \in A} b_r^{ij} \cdot (d_{ij} - \tau_{ij} \cdot \beta_k^\star) - \sum_{i \in C} a_r^i \cdot (\alpha_i^\star + \tau_i \cdot \beta_k^\star + \varphi_{kgi}^\star) \\ & - e_r''^\Delta \cdot \mu_k''^\star + \sum_{p \in F} e_r''^p \cdot (\theta_{kp}^\star - \tfrac{1}{2} \cdot \tau_p \cdot \beta_k^\star) \\ & - \sum_{s \in \overline{\mathcal{S}_g}} \sum_{p \in s} \delta_{kps}'^\star \cdot (\sum_{q \in s} e_r''^q) - \sum_{s \in \mathcal{S}_g} \sum_{p \in s} \delta_{kps}''^\star \cdot (e_r''^\Delta + \sum_{q \notin s} e_r''^q) \end{aligned} \tag{5.28}$$

The term $-\sum_{s \in \overline{\mathcal{S}_g}} \sum_{p \in s} \delta_{kps}'^\star \cdot (\sum_{q \in s} e_r''^q)$ can be rewritten according to what follows. If one considers any $s \in \overline{\mathcal{S}_g}$, the term is equivalent to putting $-\sum_{p \in s} \delta_{kps}'^\star$ on each element $q \in s$, therefore a sum over $\overline{\mathcal{S}_g}$ causes each $q \in F \backslash \{g\}$ to be charged of a term $-\sum_{p \in s} \delta_{kps}'^\star$

for each $s \in \overline{\mathcal{S}_g} : q \in s$, i.e. $\forall s \in \overline{\mathcal{S}_g} \cap \mathcal{S}_q$.
In a similar way we can rewrite term $-\sum_{s\in\mathcal{S}_g}\sum_{p\in s}\delta''^{\star}_{kps}\cdot(e''^{\Delta}_r+\sum_{q\notin s}e''^{q}_r)$: if one considers any $s \in \mathcal{S}_g$, the term is equivalent to putting $-\sum_{p\in s}\delta''^{\star}_{kps}$ on both Δ and each $q \notin s$, therefore each $q \in F\backslash\{g\}$ is charged of a further term $-\sum_{p\in s}\delta''^{\star}_{kps}$ for each $s \in \mathcal{S}_g : q \notin s$, i.e. $\forall s \in \mathcal{S}_g \cap \overline{\mathcal{S}_q}$. As a consequence we reformulate (5.28) as:

$$\begin{aligned}\bar{c}^k_r = \quad & -(\theta^{\star}_{kg}+\tfrac{1}{2}\cdot\tau_g\cdot\beta^{\star}_k)+\sum_{ij\in A}b^{ij}_r\cdot(d_{ij}-\tau_{ij}\cdot\beta^{\star}_k)-\sum_{i\in C}a^{i}_r\cdot(\alpha^{\star}_i+\tau_i\cdot\beta^{\star}_k+\varphi^{\star}_{kgi})\\ & -e''^{\Delta}_r\cdot(\mu''^{\star}_k+\sum_{s\in\mathcal{S}_g}\sum_{p\in s}\delta''^{\star}_{kps})+e''^{g}_r\cdot(\theta^{\star}_{kg}-\tfrac{1}{2}\cdot\tau_g\cdot\beta^{\star}_k)\\ & +\sum_{q\in F\backslash\{g\}}e''^{q}_r\cdot\left(\theta^{\star}_{kq}-\tfrac{1}{2}\cdot\tau_q\cdot\beta^{\star}_k-\sum_{s\in\overline{\mathcal{S}_g}\cap\mathcal{S}_q}\sum_{p\in s}\delta'^{\star}_{kps}-\sum_{s\in\mathcal{S}_g\cap\overline{\mathcal{S}_q}}\sum_{p\in s}\delta''^{\star}_{kps}\right)\end{aligned} \tag{5.29}$$

The modified support graph $G' = (V', A')$ to solve the PP is built as follows:

1. the node set $V' = V \cup \{g', N\}$ includes a copy g' of g and the dummy arrival node N;
2. the arc set $A' = A \cup \bigcup_{i\in C}\{(g', i)\} \cup \left((\Delta, N) \cup \bigcup_{p\in F}\{(p, N)\}\right)$ is enhanced with arcs from g' to each client and from each possible real arrival point to N;
3. the weights on nodes are the following:

$$w^{\star}_v = \begin{cases} -\alpha^{\star}_i-\tau_i\cdot\beta^{\star}_k-\varphi^{\star}_{kgi} & (v\equiv i\in C)\\ \theta^{\star}_{kq}-\tfrac{1}{2}\cdot\tau_q\cdot\beta^{\star}_k-\sum\limits_{\substack{s\in\overline{\mathcal{S}_g}\\ s\in\mathcal{S}_q}}\sum\limits_{p\in s}\delta'^{\star}_{kps}-\sum\limits_{\substack{s\in\mathcal{S}_g\\ s\in\overline{\mathcal{S}_q}}}\sum\limits_{p\in s}\delta''^{\star}_{kps} & (v\equiv q\in F\backslash\{g\})\\ \theta^{\star}_{kg}-\tfrac{1}{2}\cdot\tau_g\cdot\beta^{\star}_k & (v\equiv g)\\ -\mu''^{\star}_k-\sum\limits_{s\in\mathcal{S}_g}\sum\limits_{p\in s}\delta''^{\star}_{kps} & (v\equiv\Delta)\end{cases}$$

4. the weights of the arcs $ij \in A'$ are:

$$w^{\star}_{ij} = \begin{cases} d_{gj}-\tau_{gj}\cdot\beta^{\star}_k+\tfrac{1}{2}\cdot w^{\star}_j-(\theta^{\star}_{kg}+\tfrac{1}{2}\cdot\tau_g\cdot\beta^{\star}_k) & (i\equiv g')\\ d_{ij}-\tau_{ij}\cdot\beta^{\star}_k+\tfrac{1}{2}\cdot(w^{\star}_i+w^{\star}_j) & (ij\in C\times C)\\ d_{ip}-\tau_{ip}\cdot\beta^{\star}_k+\tfrac{1}{2}\cdot w^{\star}_i+w^{\star}_p & (i\in C, j\equiv p\in F)\\ d_{i\Delta}-\tau_{i\Delta}\cdot\beta^{\star}_k+\tfrac{1}{2}\cdot w^{\star}_i+w^{\star}_{\Delta} & (i\in C, j\equiv\Delta)\\ 0 & (j\equiv N)\end{cases}$$

This time, the PP is then solved as an ESPPRC on (V', A') from g' to N. Again, we add to the RMP the least reduced cost path from g' to each possible ending point.

5.1.2.3 Solving the Pricing Problem as an ESPPRC

The ESPPRC on an oriented graph $G = (V, A)$ consists in finding the shortest path from a source node to a destination node while taking into account the consumption of a set $\{1,\ldots,T\}$ of resources. With each resource $t \in \{1,\ldots,T\}$ are associated both a consumption c_{ij}^t for each arc ij and an interval $[a_i^t, b_i^t]$ of allowed values for the consumption level at each node v_i. Triangle inequality is assumed to hold for consumption terms c_{ij}^t, as it holds in the case of MRLRP (see 2.3). A path from the source node to the destination node is feasible if the level of each resource fits the corresponding interval for each of the visited nodes, or can be adapted to it. The time resource is the easiest to understand, the interval and the consumption level being, respectively, a time window and an arrival moment, which can be delayed up to the beginning of the window if needed.
In spite of being a generalization of the well-known *Shortest Path Problem (SPP)*, which is known to be solvable in polynomial time, the ESPPRC is NP-hard in the strong sense (see [Dror, 1994]). The best-known (and probably the first to have ever appeared) algorithm to solve ESPPRC on instances with positive or negative arc costs –as is our case– is the one presented in [Feillet et al., 2004]. This algorithm extends the one proposed in [Desrochers, 1988], which in turn is a generalization of the well-known Bellman-Ford algorithm to solve the SPP on graphs with negative arc weights. It implicitly enumerates all the possible paths from the source to each node v_i and associates a *state*, most often represented by means of a *label* l, with each of such paths, in order to keep track of:

- the consumption vector $\mathbf{D}_l = (D_l^1,\ldots,D_l^t)$, with D_l^t the consumption of resource t on l;
- the cost $C(\mathbf{D}_l)$ of the path;
- the vector $\mathbf{U}_l = (U_l^1,\ldots,U_l^n)$ and the number $\psi_l \in \{1,\ldots,n\}$ of *unreachable nodes*, $n \equiv |V|$. A node j is said to be unreachable for a label l associated with a path from the source node to a node i when either it has already been visited by the path, or it cannot be the next node in the path because at least one of the resource intervals would be violated on j, i.e. $(\exists t \in \{1,\ldots,T\})\ D_l^t + c_{ij}^t > b_j^t$. $U_l^j = 1$ denotes that j is unreachable for label l.

The principle of the algorithm is simple and follows the general guidelines of *Dynamic Programming (DP)*: an empty label is put on source node p, corresponding to the empty path. This label is then *extended* to each possible successor node i, generating a new label associated with the new partial path from p to i, inserting it in a set L_i of labels of node i and marking it as *nonextended*. At each step, a nonextended label l is picked from one of the nodes and extended towards each node which is reachable according to the vector $\mathbf{U}_l$. The new label is obtained by the extended one l by updating in a

straightforward way its state variables $\mathbf{D}_l$, C, $\mathbf{U}_l$ and ψ_l. As to l, it is kept in its list and marked as *extended*. The algorithm terminates when there is no label left to extend (i.e. $(\forall i)\ \{l \in L_i : l \text{ nonextended}\} = \varnothing$), yielding all Pareto-optimal paths. The way set L_i is organized and the criteria to select the next label l to extend can deeply affect the overall efficiency of the algorithm and must therefore be carefully chosen in order to enhance it.

The aim of the unreachable nodes vector $\mathbf{U}$ is twofold: on one hand it ensures to generate only elementary paths, while on the other hand it helps preventing combinatorial explosion. To achieve this second purpose, the introduction of a *dominance rule* is needed. Given two labels l, l' of a same node i, l is said to *dominate* l' if:

1. $(\forall t \in \{1,\dots,T\})\ D_l^t \leq D_{l'}^t$;
2. $C(\mathbf{D}_l) \leq C(\mathbf{D}_{l'})$;
3. $(\forall j \in V)\ U_l^j \leq U_{l'}^j$.
4. $l \neq l'$.

Condition 4 means that not only all of the state variables of l must be less or equal than those of l' (conditions 1-3), but at least one must be strictly less. Note that condition 3 implies $\psi_l = \sum_{i \in V} U_l^i \leq \psi_{l'} = \sum_{i \in V} U_{l'}^i$: this explains the use of redundant label variable ψ_l, which can help speed up the check of this dominance condition.

Dominated labels are discarded, since only nondominated partial paths can lead to optimal solutions, as the authors of [Feillet et al., 2004] have shown.

The above dominance check can be *relaxed* by simplifying or removing some of the conditions 1-3, for instance:

- ▶ by removing condition 3, or
- ▶ by replacing condition 3 with $\psi_l \leq \psi_{l'}$

in order to further speed up the DP algorithm. However, the minor accuracy in the check leads to dismiss labels which would not be dominated by a full check, i.e. to possibly have subpaths of optimal paths discarded. Hence, relaxing the dominance rule results in a *heuristic* DP-based algorithm for ESPPRC. In a Column Generation framework where the PP is an ESPPRC, using such a relaxed dominance rule can prevent some negative reduced cost variable to be found, ultimately leading to nonoptimal solutions of the (R)MP.

5.1.2.4 Branching Strategy

In order to completely define an exact algorithm, it is necessary to adapt the branching policy to the CG framework, as the branching decisions cannot concern route variables. To understand this, let r be a previously generated route s.t. $0 < x_r^k < 1$ for a given k in the fractional solution of a node n of the Branch&Bound tree. The condition $x_r^k = 1$ will cause no problem, whereas the imposition of $x_r^k = 0$ in a child node of n, say n', is likely to be uneffective: indeed, the reduced cost $\overline{c}_r^k$ of variable x_r^k will be negative in the subproblem associated with n' (since it was in the solution of n), and route r may be generated and added again to the RMP by the corresponding PP. This point is a sensitive issue. A common choice to deal with it consists in *branching on edge variables*: in the case e.g. of a (fractional) solution of $\mathcal{M}^{BP}$, we transform it into an equivalent solution of a corresponding two-index formulation, i.e. with edge variables x_{ij}^k, and we branch on these latter. Given the vector $\mathbf{x}^\star$ of the route variables in a solution of $\mathcal{M}^{BP}$, the transformation is:

$$x_{ij}^{k\,\star} = \sum_{r \in \mathscr{R}} b_r^{ij} \cdot x_r^{k\star}$$

An edge variable is suitable to be branched on if it has a fractional value, i.e. if $0 < x_{ij}^{k\,\star} < 1$. Section A.4.3 proposes an example of a plain branching rule on edge variables. For what concerns the branching strategy in the $\mathcal{M}^{BP}$-based Branch&Price algorithm, the following rules have been applied. For each branching decision, an explanation is given of how the MILP and the PPs associated with the descendents of the current Branch&Bound node are affected. The order in which the branching rules are explained reproduces the priority among them:

1. *branching on the number of used vehicles*:
   similarly to what is done in [Muter et al., 2014] when the LP relaxation of the node has been solved to optimality with CG, we compute the term $\sum_{k \in K} \widetilde{x}^k$: if it is fractional, we impose:
   - ▶ to use no more than $\overline{n_K} = \lfloor \sum_{k \in K} \widetilde{x}^k \rfloor$ vehicles in the left child. This is obtained by setting $\widetilde{x}^{|K|-\overline{n_K}-1} = 0$ and produces $(\forall k < |K| - \overline{n_K} - 1)\ \widetilde{x}^k = 0$ due to (5.13);
   - ▶ to use at least $\overline{n_K} + 1$ vehicles in the right child, by setting $\widetilde{x}^{|K|-\overline{n_K}-1} = 1$, which also produces $(\forall k > |K| - \overline{n_K} - 1)\ \widetilde{x}^k = 1$.

   Moreover, we force the right child to be solved first.
   Another attempt that has been made was to have $\overline{n_K} = \sum_{k \in K} \lceil \widetilde{x}^k \rceil$ and to impose at most $\overline{n_K} - 1$ vehicles used in the left child, and at least $\overline{n_K}$ in the right child, but this choice has not proved effective.
   Constraints (5.16), (5.17), (5.15), (5.9) and (5.10) forbid to use any route variable

associated with a vehicle k, or impose to use the vehicle, depending on whether $\widetilde{x}^k = 0$ or $\widetilde{x}^k = 1$. However, in the former case we must also prevent the Pricing phase to generate routes of the vehicle in question. Since a different PP is run for each vehicle and starting point (cf 5.1.2.2), this is done by simply *not running* the PPs related to the vehicles we do not want to use;

2. *branching on activity variables*:
   when branching on $\widetilde{x}^k$ variables does not take place, we branch on y variables. To choose the variable to branch on, we consider the least-index vehicle that has at least one fractional activity variable, and we take the most fractional among those of such vehicle.
   Constraints (5.9) and (5.10) together prevent a vehicle k to use routes that end or start at facility $p \in F$. However, if the branching decision is $y_p^k = 0$, in order to prevent the Pricing module to generate such routes we:
   - ▶ do not run the PP related to k and with p as starting point;
   - ▶ remove p as ending point from the PP graph in the PPs related to k and with a starting point different from p.

   On the other hand, the branching decision $y_p^k = 1$ does not affect the PPs, but requires to change to 1 the left-hand side of the constraints (5.18) and (5.19) related to k and p. Once this has been done, the dual variables associated with the same constraints will lead the PPs to yield routes for vehicle k that visit facility p.
3. *branching on arc variables*: this branching rules requires to transform back a solution of the LP relaxation of $\mathcal{M}^{BP}$, i.e. with routes variables x_r^k, into an equivalent solution for the three-index MILP model $\mathcal{M}^{BC}$ (cf 4.1.2), and then sum over the vehicles. Given the vector $\overline{\mathbf{x}}$ of the route variables in a solution of $\mathcal{M}^{BP}$, the transformation is:

$$\overline{x}_{ij} = \sum_{k \in K} \sum_{r \in \mathscr{R}} b_r^{ij} \cdot \overline{x}_r^k$$

   Once the solution $\overline{\mathbf{x}}$ has been transformed this way, we select the customer node i with the highest number of outgoing fractional arcs. Let n_i be this number. In the left child we impose the value 0 to:
   - ▶ the aggregate variables x_{ij} associated with the $\lceil \frac{n_i}{2} \rceil$ arcs of the set $\{ij \in \delta^-(i) : \overline{x}_{ij} > 0\}$ with lowest j indices, and to
   - ▶ those associated with the $\lfloor \frac{|\delta^-(i)| - n_i}{2} \rfloor$ arcs of the set $\{ij \in \delta^-(i) : \overline{x}_{ij} = 0\}$ with lowest j indices,

   whereas in the right child we impose the value 0 to the x_{ij} associated with all the other arcs of $\delta^-(i)$.
   When an arc ij has its x_{ij} set to 0, then in every node of the offspring of the current

Branch&Bound node we must:

- forbid every previously generated route, of no matter what vehicle, that traverses ij;
- remove the arc ij from the ESPPRC graph when running any of the PPs.

Section A.4.3 provides further details on how the ESPPRC graph should be modified in order to take the branching decisions on x_{ij} variables into account.

5.1.3 A more in-depth view of the Dynamic Programming Algorithm to solve the Pricing Problem

As said, the PP is solved as an ESPPRC by means of a Dynamic Programming algorithm. An important design decision is how the label set L_i of node i should be organized to enhance the overall efficiency of the algorithm. In our case, L_i is a list of labels sorted by increasing reduced cost firstly, then by increasing load. This helps when the label l associated with a new path from the source (be it the copy Δ' of the depot or of one of the facilities) to client i is created. A simple iteration through the whole list will allow not only to determine whether l is dominated by other labels (either already extended or not) of i, but, if this is not the case, to discard all the labels previously stored in L_i and dominated by l itself. The dominance check algorithm is shown in Figure 5.1. An important parameter of function isLabelDominated is the *dominance level*, which can be *weak* or *strong*. In the latter case, a full dominance check is performed, whereas in the former case the condition 3 of the dominance check is replaced by the condition on the ψ_l variables only, thus leading to a heuristic Pricing algorithm (cf 5.1.2.3). This makes the resulting Column Generation algorithm more flexible: in the first iterations, the PP is called with the weak dominance level, which allows a faster solving of the subproblem; when no more negative reduced cost columns can be found, we switch to the strong level to finish the computation of the LP relaxation of the current Branch&Bound node.
Finally, a *best reduced cost first* rule is used to determine the next label to be extended: the task is simplified by the aforementioned ordering of L_i lists.

5.1.3.1 Completion bound

In order to further restrain the combinatorial explosion, a *completion bound* method is used. When a new label is created, a lower bound on the cost to arrive to any of the possible ending points is added to its reduced cost: if the result is nonnegative, the label is discarded. The lower bound is given by a so-called *q-path* and is computed in a preprocessing phase by means of Dynamic Programming. q-paths, which are based on a *state-space relaxation*, have been widely used in the literature to compute dual bounds

```
isLabelDominated(L, *l, domLev)
argtype  L: labelList;  *l: label;  domLev: char;
argcond  domLev ∈ {'w', 's'};
returns  bool;
declare  l: label;
1   l ← firstLabel(L)
2   while(l ≠ null && l.rcost < *l.rcost)
3      if(l.load ≤ *l.load && uSetDom(l.U, *l.U, domLev) = 1) return true
4      l ← nextLabel(L, l)
5   while(l ≠ null && l.rcost = *l.rcost && l.load < *l.load)
6      if(uSetDom(l.U, *l.U, domLev) = 1) return true
7      l ← nextLabel(L, l)
8   while(l ≠ null && l.rcost = *l.rcost && l.load = *l.load)
9      if(uSetDom(l.U, *l.U, domLev) = 1) return true
10     if(uSetDom(l.U, *l.U, domLev) = 2) l ← discardAndNextLabel(L, l)
11     else l ← nextLabel(L, l)
12  if(l = null) insertLast(L, *l) else insertBefore(L, *l, l)
13  while(l ≠ null)
14     if(l.load ≥ *l.load && uSetDom(l.U, *l.U, domLev) = 2)
15        l ← discardAndNextLabel(L, l)
16     else
17        l ← nextLabel(L, l)
18  return false
```

```
uSetDom(U1, U2, domLev)
argtype  U1, U2: unreachableNodeVector;  domLev: char;
argcond  domLev ∈ {'w', 's'};
returns  int;
1   if(domLev = 'w')
2      if(|U1| ≤ |U2|) return 1 else return 2
3   if(domLev = 's')
4      if(U1 ⊆ U2) return 1 elif(U1 ⊃ U2) return 2 else return 3
```

FIGURE 5.1: Dominance check on a newly generated label during the DP algorithm.

in the context of exact algorithms for the CVRP, like for instance in [Christofides et al., 1981], which offers a detailed discussion on the subject.

A q-path for the CVRP is a least cost path $(\Delta, i_0,\dots,i_k, i)$, not necessarily simple, from the central depot to a customer i, with a given total load $q(w) \in W = \{q \leq Q : (\exists c \subseteq C)\ \sum_{i \in c} q_i = q\}$. W is the ordered set of all possible load values for a vehicle, and $w \in \{1,\dots,|W|\}$. Note that if customer demands are integer, then $|W| \leq Q$ and we have no more than $Q\,|C|$ q-paths to keep track of. Another possible implementation of q-paths defines $W = \{1,\dots,\overline{n}\}$, with $\overline{n}$ the maximum number of customers that can be visited in the same route, and $q(w) \equiv w$ the number of clients visited so far in the route. We used the latter implementation, but we will refer to the former to explain how do

q-paths work.

Let $f_w(i)$ denote the least cost q-path with load $q(w)$ from the depot to customer i. Let also $d_w(i)$ and $e_w(i) \in C\backslash\{i\}$ respectively be the cost of $f_w(i)$ and the predecessor of i along it. Moreover, let $w' \in W, w' < w$ the index s.t. $q(w') = q(w) - q_i$. We can write:

$$e_w(i) = \arg \min_{j \in C\backslash\{i\}} \left(d_{w'}(j) + d_{ji}\right) \qquad d_w(i) = d_{w'}(e_w(i)) + d_{e_w(i),i} \tag{5.32}$$

Equations (5.32), when applied recursively, would allow to define a q-path $f_w(i)$ for each customer i and each (feasible) load value $q(w)$, $w \in W$, even though the resulting paths would presumably be not simple. However, the above expression of $e_w(i)$ can be rewritten as in (5.33) in order to at least avoid q-paths with *two-loops*, i.e. with one customer visited twice and only one other customer visited in between:

$$e_w(i) = \arg \min_{\substack{j \in C\backslash\{i\} \\ e_{w'}(j) \neq i}} (d_{w'}(j) + d_{ji}) \tag{5.33}$$

On the other hand, the determination of $f_w(i)$ for each $i \in C$ and $w \in W$ becomes now more subtle, as we are forced to keep track of both the best q-path that delivers $q(w)$ at i, and the best one among those with a different predecessor. Let us denote by $\phi_w(i)$ such q-path, and by $\delta_w(i)$ and $\epsilon_w(i)$ the cost and the second-to-last customer of $\phi_w(i)$. Moreover, let us define function $g_w(j,i)$ as:

$$g_w(j,i) = \begin{cases} d_{w'}(j) + d_{ji} & \text{, if } e_{w'}(j) \neq i \\ \delta_{w'}(j) + d_{ji} & \text{, if } e_{w'}(j) = i \end{cases}$$

We can now give the recursive definition of functions $d_w(i)$, $e_w(i)$, $\delta_w(i)$ and $\epsilon_w(i)$:

$$d_w(i) = \min_{j \in C\backslash\{i\}} g_w(j,i) \qquad e_w(i) = \arg \min_{j \in C\backslash\{i\}} g_w(j,i) \tag{5.35}$$

$$\delta_w(i) = \min_{\substack{j \in C\backslash\{i\} \\ j \neq e_w(i)}} g_w(j,i) \qquad \epsilon_w(i) = \arg \min_{\substack{j \in C\backslash\{i\} \\ j \neq e_w(i)}} g_w(j,i) \tag{5.36}$$

The above functions can be computed recursively by initializing $d_w(i) = d_{\Delta i}$, $e_w(i) = \Delta$ for $w \in W : q(w) = q_i$; $d_w(i) = +\infty$ for any other $w \in W$; $(\forall w \in W)\ \delta_w(i) = +\infty$.

The authors of [Christofides et al., 1981] make use of q-paths to build *through-q-routes*. A through-q-route $\varrho_w(i)$ is the least cost route with no two-loops that starts and comes back at Δ, has a total load $q(w)$ and visits i. Route $\varrho_w(i)$ can be obtained by:

- taking all the couples $w', w'' \in W$ s.t $q(w') + q(w'') = q(w) + q_i$;
- for each such couple, combining
  - $f_{w'}(i)$ and $f_{w''}(i)$, if $e_{w'}(i) \neq e_{w''}(i)$, or

- $f_{w'}(i)$ and $\phi_{w''}(i)$, if $d_{w'}(i) + \delta_{w''}(i) < d_{w''}(i) + \delta_{w'}(i)$, or
- $\phi_{w'}(i)$ and $f_{w''}(i)$, otherwise

and choosing the overall least cost combination.
Through-q-routes suggest a way to compute a completion bound. Suppose we have computed in a preprocessing phase the q-paths associated with each customer i and each possible (feasible) load $q(w)$, $w \in W$. Each such q-path will have a best starting point, which is then implicit. Given a label l associated with a partial path, let i be its last visited customer, and d and q its cost and load. A lower bound to the cost to complete the path associated with l is given by the least cost q-path $f_w(i)$ s.t. $q(w) \leq Q - q + q_i$. Let:

$$w' = \arg \min_{\substack{w \in W: \\ q(w) \leq Q - q + q_i}} d_w(i)$$

if $d + d_{w'}(i) \geq 0$, label l can be discarded, as it will never result in a negative reduced cost path.
The q-paths completion bound method is used only when the Pricing algorithm is invoked with a *strong* dominance level (see 5.1.3).

5.1.3.2 *ng*-paths

Another path relaxation which has been used to compute lower bounds in exact algorithms for several VRPs is represented by *ng*-paths, which generalize q-paths in that their definition, as we will see, allows to forbid n-loops, with $n \geq 2$. They have been introduced in [Baldacci et al., 2011a] in the context of an exact method to solve both the CVRP and the *VRP with Time Windows (VRPTW)*.
An *ng*-path is defined recursively as follows. Let us define a *neighborhood* $N_i \subseteq C$ for each customer $i \in C$. Moreover, given a path ϱ starting from Δ and not necessarily simple, let:

- $(\varrho, 1),\ldots,(\varrho, \sigma(\varrho))$ index the customers it visits, i.e. $\varrho = \left(\Delta, i_{(\varrho,1)},\ldots,i_{(\varrho,\sigma(\varrho)-1)}, i_{(\varrho,\sigma(\varrho))}\right)$;
- $V(\varrho) \subseteq C$ be the set of such customers: we have $|V(\varrho)| \leq \sigma(\varrho)$ as ϱ may be not simple;
- $\pi(\varrho) = \left(\Delta, i_{(\varrho,1)},\ldots,i_{(\varrho,\sigma(\varrho)-1)}\right)$ denote the subpath of ϱ up to its second-to-last customer.

Besides, let $\Pi(\varrho)$ be the set of all the customers in ϱ (with the exception of the last one, $i_{(\varrho,\sigma(\varrho))}$) that appear in the neighborhood of all the following customers:

$$\Pi(\varrho) = \{i_{(\varrho,r)} \in V(\pi(\varrho)) : i_{(\varrho,r)} \in \textstyle\bigcap_{s=r+1}^{\sigma(\varrho)} N_{i_{(\varrho,s)}}\} \tag{5.37}$$

Note that $\Pi(\varrho)$ can contain only nodes in ϱ – more precisely, in $\pi(\varrho)$.
A path ϱ is said to be a *ng*-path if:

- $\pi(\varrho)$ is a *ng*-path and
- $i_{(\varrho,\sigma(\varrho))} \notin \Pi(\pi(\varrho))$.

Some important observations. It should be clear from the above definition that given a *ng*-path ϱ, $\Pi(\varrho)$ is the set of nodes that would make ϱ lose its property when added to it as next visited customer. It is therefore, in a sense, a set of *forbidden* nodes. An even more important thing is the fact that since $\Pi(\varrho) \subset V(\varrho)$, an elementary path is a *ng*-path by definition.
It is easy from (5.37) to infer the expression of how to update $\Pi(\varrho)$ as long as ϱ grows:

$$\Pi(\varrho) = \big(\Pi(\pi(\varrho)) \cup \{i_{(\varrho,\sigma(\varrho)-1)}\}\big) \cap N_{i_{(\varrho,\sigma(\varrho))}} \tag{5.38}$$

A small example can help understand how do *ng*-paths work. Let $\varrho = \big(\Delta, 1, 2, 3, 4, 1\big)$: we have $V(\varrho) = \{1, 2, 3, 4\}$, $\sigma(\varrho) = 5$ and $\pi(\varrho) = \big(\Delta, 1, 2, 3, 4\big)$. Suppose that $N_1 = \{3, 4\}$, $N_2 = \{1, 5\}$, $N_3 = \{1, 4\}$ and $N_4 = \{2, 3\}$. We will have:

$$1 \notin N_2 \cap N_3 \cap N_4 \cap N_1, \quad 2 \notin N_3 \cap N_4 \cap N_1, \quad 3 \in N_4 \cap N_1, \quad 4 \in N_1 \;\Rightarrow\; \Pi(\pi(\varrho)) = \{3, 4\}$$

and since $\pi(\varrho)$ is a *ng*-path due to its elementariness, and $1 \notin \Pi(\pi(\varrho))$, ϱ is a *ng*-path. The example also shows that although a simple path is always an *ng*-path, the opposite is generally not true, unless $(\forall i \in C)\ N_i = C$. However, it is easy to see that by suitably choosing the node neighborhoods N_i, *ng*-paths offer a very good approximation of elementariness. This can be accomplished even with very simple neighborhoods, like e.g. defining $N_i \equiv N_i^n$, with N_i^n the set of the n nearest nodes to i, and n a convenient compromise value.
We have used *ng*-paths precisely to exploit this feature. In the DP algorithm to solve the ESPPRC, the state of a path ϱ associated with a label l is given by the load q, the cost d, and the full vector of unreachable nodes: this means that for each couple (q, d) we can theoretically have $\mathcal{O}(2^{|C|})$ labels on a node. By replacing $\mathbf{U}_l$ with $\Pi(\varrho) \subseteq N_{i_{(\varrho,\sigma(\varrho))}}$, the same DP algorithm generates *ng*-paths, hence yielding Pareto-optimal *ng*-paths. The combinatorics is significantly reduced, as for each couple (q, d) we now have no more than $2^{|N_{i_{(\varrho,\sigma(\varrho))}}|}$ labels. Of course, the algorithm is prone to generate nonelementary *ng*-paths and is therefore no more valid for the ESPPRC, but a fine tuning of the neighborhood sets during its execution can hopefully make it converge to a set of optimal elementary paths. Such tuning can be made with a little effort. Let us initialize the neighborhood

sets as $N_i = N_i^n$, with n s.t. 2^n is a *small* value, and suppose that a path with a t-loop:

$$\varrho = \left(\Delta, i_{(\varrho,1)},\dots,i_{(\varrho,s-1)}, i_{(\varrho,s)}, i_{(\varrho,s+1)},\dots,i_{(\varrho,s+t)} \equiv i_{(\varrho,s)}, i_{(\varrho,s+t+1)},\dots,i_{(\varrho,\sigma(\varrho))}\right)$$

has been generated and inserted in the set P of optimal *ng*-paths by the DP algorithm. Once the loop $\left(i_{(\varrho,s)}, i_{(\varrho,s+1)},\dots,i_{(\varrho,s+t-1)}, i_{(\varrho,s)}\right)$ has been detected, two possible options are:

1) add $i_{(\varrho,s)}$ to $N_{i_{(\varrho,s+1)}},\dots,N_{i_{(\varrho,s+t-1)}}$, do the same for every loop in every nonelementary path in P and launch again the DP algorithm;
2) end the PP and:
   a) return the elementary paths in P;
   b) *correct* the nonelementary paths of P so as to make them simple, and return those which still have a negative reduced cost. The correction can be done e.g. by shortcutting loops: $\varrho \to \varrho' = \left(\Delta, i_{(\varrho,1)},\dots,i_{(\varrho,s-1)}, i_{(\varrho,s+1)},\dots,i_{(\varrho,s+t)} \equiv i_{(\varrho,s)}, i_{(\varrho,s+t+1)},\dots,i_{(\varrho,\sigma(\varrho))}\right)$.

We have chosen option 1): experimental sessions have shown that even when nonelementary paths have been found, a few more iterations have been sufficient to converge to a set of only elementary paths.
One could wonder whether this strategy leads to a heuristic PP, which actually is not the case. Indeed, by using the neighborhood sets and *ng*-paths instead of the unreachable customers vector we are not relaxing the dominance check: we are performing a full check on the paths of a *state-space relaxation*. Hence, the resulting paths in P are not suboptimal: they are optimal w.r.t. a relaxed problem. Therefore, when the set P is made of elementary paths only (possibly after correcting neighborhood and rerunning the algorithm), we have the optimal solution of the ESPPRC.

5.1.4 Preliminary Computational Results

A session of preliminary tests has been conducted to assess the effectiveness of the presented Branch&Price algorithm. The tests consist of a performance comparison with the Branch&Cut algorithm based on model $\mathcal{M}_{RA}^{BC}$ (see section 4.2) on a sample of small-sized TZK and CCL instances. The two algorithms are requested to perform a complete computation of the root node, i.e. to terminate just before branching on it. The results are shown in table 5.1.
As one can see, the gap achieved by the Branch&Price is generally much worse of that of the Branch&Cut, except for a few cases where it is comparable or slightly better. Further, the time is in most of the cases much higher. For most of the instances with 72 or 75 customers (see tables 4.2 and 4.4 for the detail of instance features) it is impossible

instance	$\mathcal{M}^{BP}$		$\mathcal{M}^{BC}_{RA}$	
	t_r	$\%_r$	t_r	$\%_r$
50c3d2v	182	4.87	9	1.65
50c3d4v	655	17.83	9	10.72
50c3d6v	386	39.40	14	11.32
50c5d2v	138	5.10	10	1.81
50c5d4v	146	16.39	12	7.61
50c5d6v	216	31.86	15	10.41
50c7d2v	36	3.48	15	1.92
50c7d4v	66	4.31	11	2.38
50c7d6v	111	24.25	15	11.08
75c3d2v		$+\infty$	54	1.62
75c3d4v		$+\infty$	50	2.84
75c3d6v		$+\infty$	178	7.78
75c5d2v	866	1.45	64	3.39
75c5d4v		$+\infty$	40	6.01
75c5d6v	708	13.10	58	8.13
75c7d2v		$+\infty$	57	1.73
75c7d4v		$+\infty$	36	4.88
75c7d6v		$+\infty$	34	7.14
a1	55	7.07	41	7.59
d1	89	6.12	39	7.62
a2	1144	11.33	6	6.68
g1	916	4.83	136	4.76
j1		$+\infty$	57	5.40
g2		$+\infty$	30	5.26

TABLE 5.1: Comparison of the behavior of the $\mathcal{M}^{BP}$-based Branch&Price algorithm to that of the Branch&Cut algorithm based on model $\mathcal{M}^{BC}_{RA}$. Both are requested to perform a complete computation of the root node on a sample of small-sized instances, under a 1200s time limit. The computation time (s) and the best achieved gap are reported, unless the time limit is exceeded without finding a lower bound.

to even find a lower bound within the time limit of 1200s, which on the other hand is always largely sufficient for the Branch&Cut algorithm to accomplish the task.

5.1.5 Issues of Model $\mathcal{M}^{BP}$

However, when looking at the Branch&Price results only, one can see a clear dependence on the number n_K of vehicles. This can partly explain the weakness of the lower bound yielded by the continuous relaxation of the model $\mathcal{M}^{BP}$ (and therefore the poor performances of the Branch&Price algorithm) since this model still has a vehicle index k – which is not the case for the model $\mathcal{M}^{BC}_{RA}$. By the way, this also implicitly suggests that a removal of such vehicle index may help to improve the algorithm efficiency.
A further investigation allows to detect another important weakness of the model $\mathcal{M}^{BP}$. Suppose to have a small instance with $n = 3$ and $f = 1$ like the one shown in figure 5.2. Further, suppose cost and time matrices to be symmetric, and $Q \geq \sum_{i \in C} q_i$. The two leftmost subfigures represents two solutions. In both of them, the routes are performed by only one vehicle – otherwise they would not respect the connection requirements.

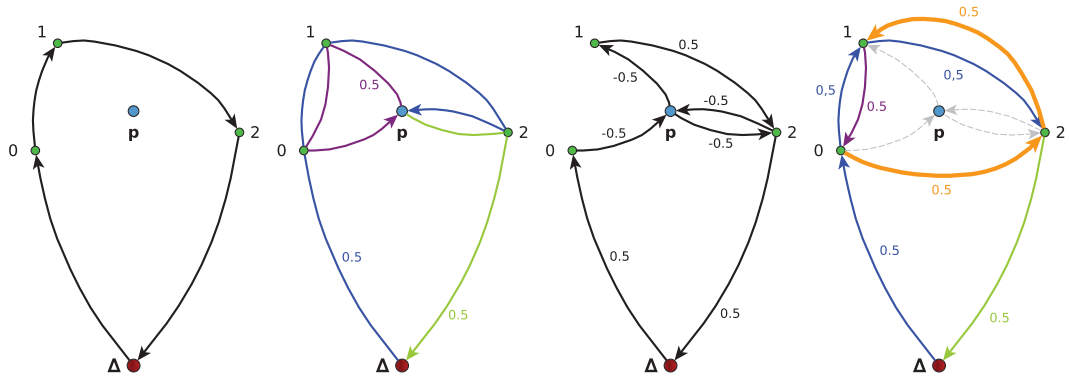


FIGURE 5.2: Study of a small instance with $n = 3$ and $f = 1$. From the leftmost to the rightmost subfigure we have: the integer optimal solution; a fractional solution that uses three routes, each represented by a color; a *difference graph*, useful to calculate the cost difference between the two solutions; finally, the same fractional solution expressed in terms of *base* and *replenishment arcs* (cf 4.2.1). The fractional solution turns out to be more convenient and feasible w.r.t. the linear relaxation of model $\mathcal{M}^{BP}$, whereas the connectivity valid inequality of model $\mathcal{M}^{BC}_{RA}$ upon set $S = \{0, 1, 2\}$ would cut it.

Therefore we can suppose without loss of generality that $n_K = 1$. This will also allow us to show that the issue we are talking about is not vehicle-related. Let $k = 0$ denote the only vehicle. The first subfigure represents the minimum cost integer solution of model $\mathcal{M}^{BP}$, which we refer to as $\overline{\mathbf{x}}$, that makes use of only one route. The second-from-left image represents a fractional solution, $\mathbf{x}^\star$, which uses three routes, each with $x_r^{0\star} = 0.5$. The fractional solution $\mathbf{x}^\star$ is feasible w.r.t. model $\mathcal{M}^{BP}$:

- each client is visited by a set of routes whose route variables sum up to 1 (constraints (5.8));
- the collection $\mathcal{S}_p$ has only one element, i.e. $s = \{p\}$. For s we have $\sum_{r\in\mathscr{R}} b_r'^s x_r^{0\star} = 0.5$, since there is only one route s.t. $b_r'^s = 1$ (in the example, the blue one). Since $|\mathcal{S}_p| = |K| = |F| = 1$, the family (5.18) has only one constraint, that on s, i.e. $\sum_{r\in\mathscr{R}} b_r'^s x_r^{0\star} \geq y_p^{0\star}$, which holds: this is because the lower bound imposed on $y_p^{0\star}$ by constraints (5.9) is exactly 0.5, as no route leaving p has $x_r^{0\star} > 0.5$. With a similar reasoning on the only constraint (5.19), one concludes that $\mathbf{x}^\star$ is compliant to connection requirements. This is a crucial point;
- as to constraints (5.11) and (5.12), we can suppose without loss of generality that they are satisfied by $\mathbf{x}^\star$;
- finally, it is straightforward to see that $\mathbf{x}^\star$ also respects the other constraints, notably (5.10) and (5.15)–(5.17).

Moreover, the third subfigure shows that the saving achieved by $\mathbf{x}^\star$ w.r.t. $\overline{\mathbf{x}}$ is:

$$\sum_{r\in\mathscr{R}} c_r(\overline{x}_r^0 - x_r^{0\star}) = 0.5\ (d_{\Delta 0} + d_{12} + d_{2\Delta} - d_{0p} - d_{p1} - d_{2p} - d_{p2})$$

hence, by suitably locating the customers, it is easy to make it greater than 0 and thus $\mathbf{x}^\star$ more convenient that $\overline{\mathbf{x}}$. Therefore, when relaxing the integrality constraints, the solving of model $\mathcal{M}^{BP}$ may yield a solution like $\mathbf{x}^\star$, as it would be fully compliant to its constraints.
On the other hand, a solution like $\mathbf{x}^\star$ would be cut by the separation procedure of the $\mathcal{M}^{BC}_{RA}$-based Branch&Cut, and precisely by the separation of the *connectivity valid inequalities* (4.40). This is shown by the rightmost subfigure, from which it is clear that the inequality (4.40) on set $S = \{0, 1, 2\}$ is violated:

$$x(A_0(S)) + w(A_P(S)) = 2.5 > 2 = |S| - 1$$

This is a major issue in the Branch&Price algorithm proposed so far. Connectivity constraints (5.18) and (5.19), in spite of being sufficient for the integer problem, are not as strong as constraints (4.40) when integrality is relaxed. Note that similar considerations apply for the Branch&Cut algorithm: as it has been shown in table 4.6 (section 4.2.5), although relations (4.40) are not problem-defining (i.e. their insertion in the model $\mathcal{M}^{BC}_{RA}$ is not necessary), removing them from the separation procedure NODE-SEPARATION (section 4.2.4.4) considerably worsens the performances of the Branch&Cut algorithm, which can become as bad as the performances of the Branch&Price algorithm.

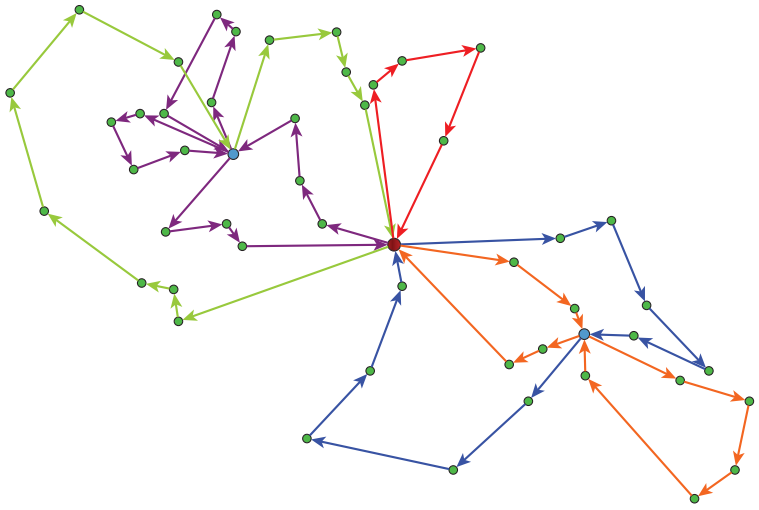
5.2 A new Branch&Price Algorithm for the VRPIRF

In this section we present another Branch&Price algorithm that relies on a new MILP formulation for VRPIRF. The aim is to tackle symmetry issues by dropping out the vehicle index, as it has been done to upgrade $\mathcal{M}^{BC}$ to $\mathcal{M}^{BC}_{RA}$. The new formulation, $\mathcal{M}^{BP}_{RA}$, will merge elements of the $\mathcal{M}^{BP}$ model with some of the $\mathcal{M}^{BC}_{RA}$ (see section 4.2.3), notably *replenishment arcs* and *arrival times* that have been introduced precisely for this purpose. The insertion in the model of these tools will hopefully help reducing the connection issues shown in section 5.1.5. The new formulation is explained in section 5.2.1, while section 5.2.2 introduces the algorithm.

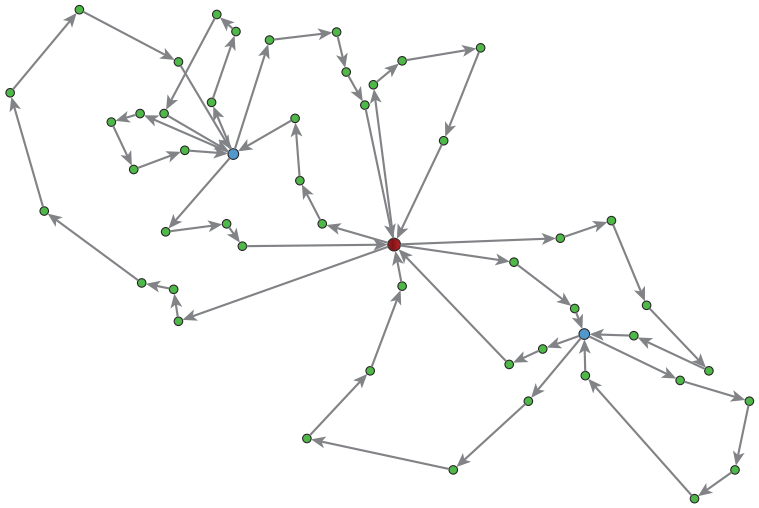
5.2.1 A new Set-Partitioning formulation without the vehicle index

A solution to overcome vehicle-related symmetry issues consists in using *arrival times* and *replenishment arcs* as it has been done in the compact formulation $\mathcal{M}^{BC}_{RA}$. Arrival times (cf 4.2.2) enable to measure the consumption of the time resource along a rotation: the association between a vehicle and the routes it performs to compute its total service

time can be disregarded, and the vehicle index k removed. Further, arrival times assure the connection of a solution as a *side-effect*. However, in order to do so, a rotation must be represented as a sequence of arcs in which each intermediate node must have indegree and outdegree equal to 1. This need is made evident by figure 5.3.



(a)



(b)

FIGURE 5.3: Dropping the vehicle index is likely to considerably reduce the drawbacks of model $\mathcal{M}^{BP}$. However, this imposes to change the way rotations are represented, in order to be able to distinguish them and overcome the (possibly) multiple indegree and outdegree of the facility nodes.

This representation change is precisely what *replenishment arcs* (4.2.1) allow to do. The introduction of these two tools requires the support graph of model $\mathcal{M}^{BP}_{RA}$ to merge the support graph of model $\mathcal{M}^{BP}$ with that of model $\mathcal{M}^{BC}_{RA}$:

▶ the node set is $V = C \cup \{\Delta\} \cup F$, i.e. the same of $\mathcal{M}^{BC}$ (cf 4.1.1) and $\mathcal{M}^{BP}$;

- the arc set $A = A'_0 \cup A_P$, with $A'_0 = V \times C \cup C \times V$, incorporates both the arc set of $\mathcal{M}^{BP}$ and the *replenishment arcs* seen in $\mathcal{M}^{BC}_{RA}$. For ease of language, and to explicitly refer to the features of model $\mathcal{M}^{BC}_{RA}$, both the name *base arcs* and the symbol A_0 will denote the arcs in $\{ij \mid i, j \in C \cup \{\Delta\} : i \neq j\}$ only. We will explain this shortly.

Moreover, three sets of variables must be added to represent the usage of base or replenishment arcs and the measure of arrival times. Lastly, new constraints to link arc and route variables are required.

5.2.1.1 Decision Variables

The overall set of decision variables of model $\mathcal{M}^{BP}_{RA}$ is composed by:

- binary *route variables* x_r, $r \in \mathscr{R}$, $x_r = 1 \Leftrightarrow$ route r is in solution;
- binary *activity variables* y_p, $p \in F$, $y_p = 1 \Leftrightarrow$ at least one recharge occurs at $p \in F$;
- binary *base arc variables* $x_{ij}, ij \in A_0, x_{ij} = 1 \Leftrightarrow$ node j follows node i in the same route;
- binary *replenishment arc variables* w_{ij}, $ij \in A_P$, $w_{ij} = 1 \Leftrightarrow$ a replenishment (at the most convenient facility) takes place in between customers i and j;
- real nonnegative *arrival time variables* z_{ij}, $i \in V$, $j \in V \backslash \{i\}$, the arrival time at node j if its predecessor is i (i.e. if $x_{ij} + w_{ij} = 1$)

As one can see, no arc variable is explicitly associated with arcs in $A'_0 \backslash A_0 = C \times F \cup F \times C$, i.e. with arcs connecting customer nodes and facility nodes. Of course, they are in the graph and are necessary to form a route that has a recharge either at the beginning or at the end; moreover, they are needed when computing the cost $c_r = \sum_{ij \in A'_0} b^{ij}_r \cdot d_{ij}$ of a route, and essential in the graph of the PP. Nevertheless, to use the symbol A_0 and the phrase *base arcs* in the same sense of section 4.2, they are not a part of set A_0. The example of figure 5.4 depicts the different components of A.

5.2.1.2 The MILP Formulation

The model $\mathcal{M}^{BP}_{RA}$ is given in the following. Once again, dual variables have been written alongside the constraint concerning route variables x_r as they will be useful later to determine their reduced costs.

$$(\mathcal{M}^{BP}_{RA})$$
$$\min \sum_{r \in \mathscr{R}} c_r x_r \tag{5.39}$$

$$\text{s.t.} \sum_{r\in\mathscr{R}} a_r^i\ e_r'^p\ x_r \le y_p \qquad \forall p\in F, i\in C \qquad \varphi_{pi} \qquad (5.40)$$

$$\sum_{r\in\mathscr{R}} (e_r'^p - e_r''^p)\ x_r = 0 \qquad \forall p\in F \qquad \theta_p \qquad (5.41)$$

$$\sum_{r\in\mathscr{R}} (e_r'^\Delta - e_r''^\Delta)\ x_r = 0 \qquad \theta_\Delta \qquad (5.42)$$

$$\sum_{r\in\mathscr{R}} e_r'^\Delta x_r \le n_K \qquad \xi \qquad (5.43)$$

$$\sum_{r\in\mathscr{R}} b_r^{ij} x_r = x_{ij} \qquad \forall ij\in A_0 \qquad \eta_{ij} \qquad (5.44)$$

$$\sum_{ji\in A_0} x_{ji} + \sum_{ji\in A_P} w_{ji} = \sum_{ij\in A_0} x_{ij} + \sum_{ij\in A_P} w_{ij} \qquad \forall i\in C \qquad (5.45)$$

$$\sum_{ji\in A_0} x_{ji} + \sum_{ji\in A_P} w_{ji} = 1 \qquad \forall i\in C \qquad (5.46)$$

$$z_{\Delta i} = t_{\Delta i} x_{\Delta i} \qquad \forall i\in C \qquad (5.47)$$

$$(t_{\Delta i} + t_{ij}) x_{ij} + (t_{\Delta i} + u_{ij}) w_{ij} \le z_{ij} \qquad \forall i\in C, j\in C\backslash\{i\} \qquad (5.48)$$

$$z_{ij} \le (T - t_{j\Delta})(x_{ij} + w_{ij}) \qquad \forall i\in C, j\in C\backslash\{i\} \qquad (5.49)$$

$$(t_{\Delta i} + t_{i\Delta}) x_{i\Delta} \le z_{i\Delta} \qquad \forall i\in C \qquad (5.50)$$

$$z_{i\Delta} \le T x_{i\Delta} \qquad \forall i\in C \qquad (5.51)$$

$$\sum_{\substack{j\in V\\ j\neq i}} z_{ij} = \sum_{\substack{j\in V\\ j\neq i}} z_{ji} + \sum_{\substack{j\in V\\ j\neq i}} t_{ij} x_{ij} + \sum_{\substack{j\in C\\ j\neq i}} u_{ij} w_{ij} \qquad \forall i\in C \qquad (5.52)$$

$$\sum_{r\in\mathscr{R}} b_r'^s\ x_r \ge y_p \qquad \forall s\subseteq F, p\in s \qquad \delta'_{ps} \qquad (5.53)$$

$$\sum_{r\in\mathscr{R}} b_r''^s\ x_r \ge y_p \qquad \forall s\subseteq F, p\in s \qquad \delta''_{ps} \qquad (5.54)$$

$$x_r \in \{0,1\} \qquad \forall r\in\mathscr{R}$$

$$y_p \in \{0,1\} \qquad \forall p\in F$$

$$x_{ij} \in \{0,1\} \qquad \forall ij\in A_0$$

$$w_{ij} \in \{0,1\} \qquad \forall ij\in A_P$$

$$z_{ij} \ge 0 \qquad \forall i\in V, j\in V\backslash\{i\}$$

Relations (5.41) impose a null balance of routes starting and arriving at facility $p\in F$. The same does constraint (5.42) w.r.t. Δ, while (5.43) bounds the number of routes leaving Δ to the number of vehicles, n_K.

Constraints (5.44) link route variables and base arcs, while (5.45) assert that each customer node must have equal indegree and outdegree: combined with (5.46), this amounts to have exactly one incoming arc and one outgoing arc, be they base or replenishment arcs. The main purpose of (5.45) is certainly to concatenate routes of a rotation by means of replenishment arcs, since when a route r is chosen, i.e. $x_r = 1$, constraints (5.44) set to 1 all the x_{ij} variables related to the base arcs of r. Therefore, the nodes associated with the last customer before a replenishment, and the first one after, can

only have their arc flow satisfied by replenishment arcs, as no base arc is associated with facilities. This is shown by figure 5.4.

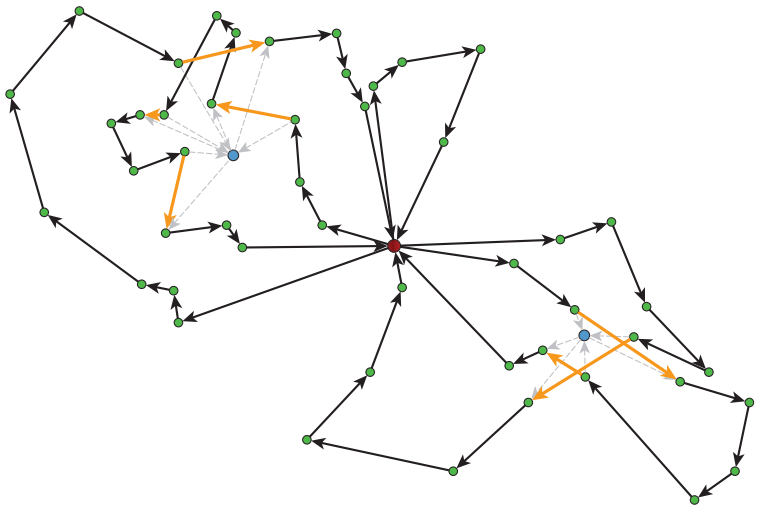


FIGURE 5.4: A possible way to concatenate the routes of figure 5.3 (b) by means of replenishment arcs in order to obtain a solution equivalent to 5.3 (a). Base arcs are drawn in black; replenishment arcs are thicker and drawn in orange; lastly, dashed grey arcs represents the elements of $A'_0 \backslash A_0 = C \times F \cup F \times C$.

Constraints (5.47)-(5.52) are identical to (4.33)-(4.38): they determine the arrival times along a rotation in order to enforce the maximum shift length. As a recall, u_{ij} is the extended travel time associated with the replenishment arc w_{ij}. Therefore, (5.47)–(5.52) replace (5.11)–(5.12).

Finally, constraints (5.40) are activity detection constraints to support connectivity constraints (5.53) and (5.54). Therefore, the whole of (5.40), (5.53) and (5.54) have the same meaning of (5.9), (5.18) and (5.19). However, the former are weaker than the latter, as variable y_p detects whether there is *some* activity at facility p, regardless of the involved vehicles. Analogously, terms $\sum_{r\in\mathscr{R}} b'^s_r\ x_r$ and $\sum_{r\in\mathscr{R}} b''^s_r\ x_r$ account for all the routes that either have the starting point in $\{\Delta\} \cup F \backslash s$, $s \subseteq F$ and the endpoint in s, or viceversa, no matter which rotation they belong to. Moreover, (5.40) and (5.53)–(5.54) are unnecessary, as connection is assured by (5.47)-(5.52). We keep them in the model as they can hopefully help to raise the lower bound.

One could wonder why the constraint related to the service of customers has been imposed by means of (5.46) instead of a relation of the form of (5.8). The reason is that the second option would have raised issues in the representation of rotations in the case of single-client routes, as shown in figure 5.5. Single-client routes are the only case of route with no base arcs: without constraints (5.46), in such case relations (5.45) do not allow a correct modelization of rotations. Consider for instance customer 2 in the figure. If constraints (5.46) were not in the model, relation (5.45) for customer 2 would have

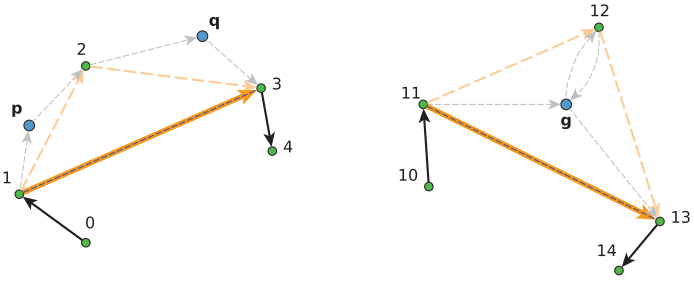


FIGURE 5.5: If the service to a customer node was imposed by assuring that at least one route visits it, rotations with single-client routes would look like this. Base arcs are drawn in black, while grey dashed arcs are the elements of $A'_0 \backslash A_0$. Dashed, light orange arcs represent the *correct* way to concatenate the routes by means of replenishment arcs, whereas blue/orange ones are the replenishment arcs that would be improperly used instead.

both sides null for cost reasons. At the same time, customer 1 would need an outgoing replenishment arc, and customer 3 an incoming one: this would produce the modelization error $w_{13} = 1$, and therefore the erroneous representation of the left subfigure.
An interesting last observation can be made by comparing models $\mathcal{M}_{RA}^{BP}$ and $\mathcal{M}_{RA}^{BC}$. Suppose to remove unnecessary constraints (5.40), (5.53) and (5.54) from $\mathcal{M}_{RA}^{BP}$, and (4.27) from $\mathcal{M}_{RA}^{BC}$. It is easy to see that:

- ▶ constraints (5.45)–(5.46) are identical to (4.28)–(4.29);
- ▶ the same can be said for relations (5.47)–(5.52) w.r.t. (4.33)–(4.38);
- ▶ constraints (5.42)–(5.43), concerning the degree of Δ, are equivalent to (4.30)–(4.31).

As a consequence, model $\mathcal{M}_{RA}^{BP}$ can be considered as a *Dantzig-Wolfe Decomposition (DWD)* of model $\mathcal{M}_{RA}^{BC}$. DWD is a widely used technique to deal with Linear Problems (LPs) such as $\min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq 0\}$. It consists of dividing the constraints $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ of the original LP in two subfamilies $\mathbf{A}'\mathbf{x} \geq \mathbf{b}'$ and $\mathbf{A}''\mathbf{x} \geq \mathbf{b}''$. This defines two subproblems, commonly called *master problem* and *subproblem*, each associated with a polyhedron that includes all of its solutions. Let us consider the problem $\min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}'\mathbf{x} \geq \mathbf{b}', \mathbf{x} \geq 0\}$ as the master problem. As it is stated by the *Minkowski-Weyl theorem*, each point enclosed by a polyhedron can be expressed as a convex combination of its extreme points, plus a nonnegative combination of its extreme rays. Therefore, by taking e.g. the polyhedron $H'' = \{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}''\mathbf{x} \geq \mathbf{b}'', \mathbf{x} \geq 0\}$ defined by the subproblem, one can reformulate the original LP as:

$$\min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}'\mathbf{x} \geq \mathbf{b}', \mathbf{x} = \sum_{p \in P''} \lambda_p \mathbf{x}''_p + \sum_{r \in R''} \lambda_r \mathbf{x}''_r, \sum_{p \in P''} \lambda_p = 1, \boldsymbol{\lambda} \geq 0\}$$

where $\mathbf{x}''_p, p \in P''$ and $\mathbf{x}''_r, r \in R''$ are the sets of extreme points and rays of H'', and the relations $\mathbf{x} = \sum_{p \in P''} \lambda_p \mathbf{x}''_p + \sum_{r \in R''} \lambda_r \mathbf{x}''_r, \sum_{p \in P''} \lambda_p = 1$ allow to link the original

variables $\mathbf{x}$ to the subproblem variables.

DWD is intimately related to *Column Generation (CG)* as most of the times the extreme points of the subproblem's polyhedron are progressively determined by CG. We refer the reader to well-known works as [Barnhart et al., 1996, Vanderbeck and Savelsbergh, 2006], and again to [du Merle et al., 1997, Feillet, 2010, Lübbecke and Desrosiers, 2002, Vanderbeck and Wolsey, 2009], to have an exhaustive view on these subjects.

When applied to Mixed-Integer Linear Program, DWD can be used to compute the Linear Relaxation of the integer problem, although it does not yield a tighter bound, unless a *partial convexification* is performed. This latter consists in including the integrality constraint in the subproblem. In this case, the linear relaxation of the original problem is reformulated as:

$$\min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}'\mathbf{x} \geq \mathbf{b}', \mathbf{x} \in \text{conv}(\{\mathbf{x} \geq 0 \mid \mathbf{A}''\mathbf{x} \geq \mathbf{b}''\})\}$$

This can help considerably when tackling huge Mixed-Integer Problems, provided that the integer subproblem has not the integrality property and is nevertheless easily solvable.

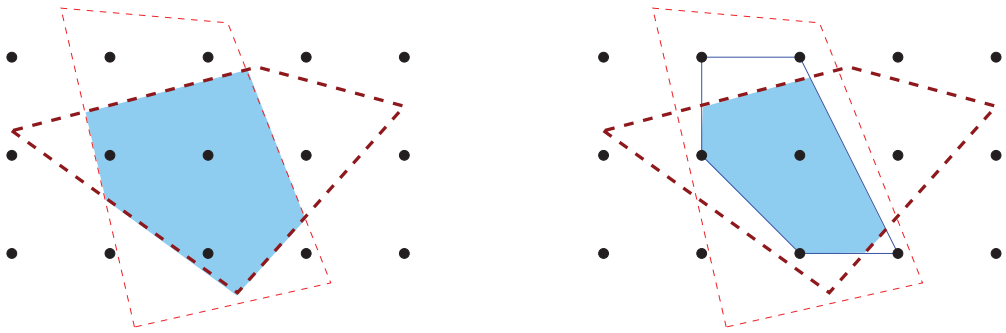


FIGURE 5.6: When including the integrality contraint in the subproblem, the feasible region is actually restrained and it is possible to achieve tighter bounds.

In our case, the subproblem of $\mathcal{M}_{RA}^{BC}$ consists of the capacity inequality (4.32) only. By solving it as an ESPPRC with Dynamic Programming, we convexify it.

Relations allows to link (5.44) route variables and base arc variables. However, in order for the models $\mathcal{M}_{RA}^{BC}$ and $\mathcal{M}_{RA}^{BP}$ to be equivalent, relations (5.41) are necessary. The reason for this is subtle. Relations (5.45) and (5.46) state that each client will have indegree and outdegree equal to 1, but (5.44) assert that a base arc variable will be activated only for route arcs that do not concern a facility. Hence, in model $\mathcal{M}_{RA}^{BP}$, the last customer of every route that ends at a facility and the first of each route that starts at a facility will need a replenishment arc to satisfy its arc flow (5.45). At this point, relations (5.41) are essential to make a proper use of replenishment arcs in model $\mathcal{M}_{RA}^{BP}$. Indeed, in $\mathcal{M}_{RA}^{BC}$, the use of a replenishment arc $w_{ij} = 1$ *implies by definition* that the

same vehicle visits customer i, then replenishes at facility p, then visits j, p being the most convenient facility (cf 4.2.1). On the contrary, in model $\mathcal{M}_{RA}^{BP}$ this is not implicit and the omission of (5.41) could cause an improper use of replenishment arcs, as shown in the leftmost case of figure 5.7. In theory this can also take place with (5.41), i.e. when the null balance of incoming and outgoing routes of each facility is imposed, as in the rightmost case of the same figure. However, this is unlikely to happen due to the costs and the triangular inequality.

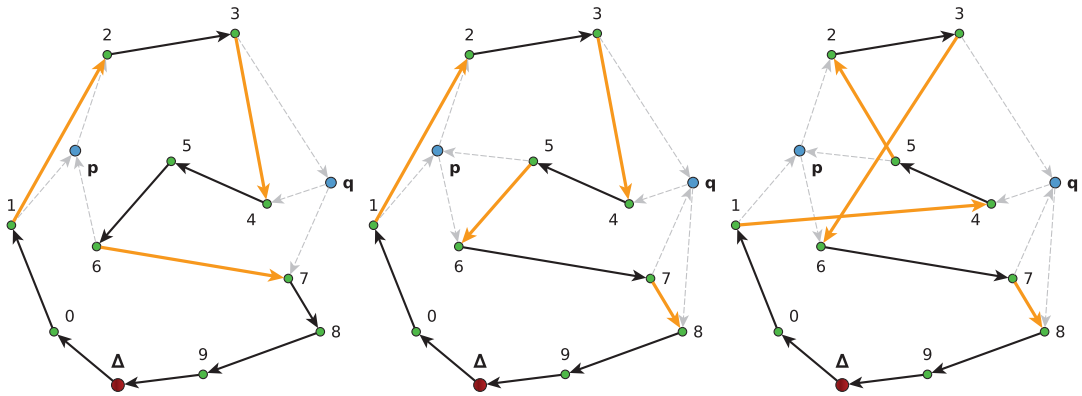


FIGURE 5.7: Constraints (5.41) are essential to the correctness of model $\mathcal{M}_{RA}^{BP}$, as if we discard them, the need to fulfill the arc flow at each client can lead to an improper use of replenishment arcs variables. In the case on the left, the facility visited after node 6 and the one visited before node 7 are not the same, therefore $w_{67} = 1$ represents a modelization error. The imposition of a null balance of incoming and outgoing routes of each facility (center and right case), along with the costs and the triangular inequality, allows to overcome this. The correct values of the w variables according to the chosen routes are those in the center figure: the saving w.r.t. the rightmost figure, i.e. $f_{36} + f_{52} + f_{14} - (f_{12} + f_{34} + f_{56}) = \sum_{ij \in \{36,52,14\}} \min(d_{ip} + d_{pj}, d_{iq} + d_{qj}) - \sum_{ij \in \{12,34,56\}} \min(d_{ip} + d_{pj}, d_{iq} + d_{qj}) = d_{3p} + d_{p4} - (d_{3q} + d_{q4}) > 0$, will lead the model to do the right choice.

5.2.2 The new Branch&Price Algorithm

The general guidelines of the new Branch&Price algorithm based on $\mathcal{M}_{RA}^{BP}$ are the same of the one presented in section 5.1.2. A Column Generation-based framework is considered, where the PP consists of an ESPPRC to determine new negative reduced cost route variables. The ESPPRC is solved by means of a DP algorithm inspired by the one presented in [Feillet et al., 2004] and enhanced with *ng*-paths and a *q*-paths-based completion bound method to restrain the combinatorial explosion. Then, the exploration of the Branch&Bound tree is guided by problem-tailored branching rules. In the following, we present the determination of the reduced costs of route variables x_r and the setting of the PP, in section 5.2.2.1, while section 5.2.2.2 is devoted to the branching rules that have been designed to generate the child nodes of the generic Branch&Bound node.

5.2.2.1 Reduced Costs

The reduced cost of variable x_r is:

$$\overline{c_r} = c_r - \sum_{i\in C}\sum_{p\in F} a_r^i \cdot e_r'^p \cdot \varphi_{pi}^\star - \sum_{p\in F}(e_r'^p - e_r''^p)\cdot\theta_p^\star - (e_r'^\Delta - e_r''^\Delta)\cdot\theta_\Delta^\star - e_r'^\Delta\cdot\xi^\star$$
$$- \sum_{s\subseteq F}\sum_{p\in s}(b_r'^s\cdot\delta_{ps}'^\star + b_r''^s\cdot\delta_{ps}''^\star) - \sum_{ij\in A_0} b_r^{ij}\cdot\eta_{ij}^\star$$

where $\varphi_{pi}^\star \le 0$, $\theta_p^\star \gtrless 0$, $\theta_\Delta^\star \gtrless 0$, $\xi^\star \le 0$, $\eta_{ij}^\star \gtrless 0$, $\delta_{ps}'^\star \ge 0$, $\delta_{ps}''^\star \ge 0$ are the values in the current dual solution (i.e. corresponding to the current solution of the RMP) of the dual variables associated with the constraints involving x_r.
The above expression can be rewritten as:

$$\begin{aligned}\overline{c_r} = \quad & \sum_{ij\in A_0} b_r^{ij}\cdot(d_{ij} - \eta_{ij}^\star) + \sum_{ij\in A_0'\setminus A_0} b_r^{ij}\cdot d_{ij} - \sum_{i\in C}\sum_{p\in F} a_r^i\cdot e_r'^p\cdot\varphi_{pi}^\star \\ & - \sum_{p\in F}(e_r'^p - e_r''^p)\cdot\theta_p^\star - (e_r'^\Delta - e_r''^\Delta)\cdot\theta_\Delta^\star - e_r'^\Delta\cdot\xi^\star \\ & - \sum_{s\subseteq F}\sum_{p\in s}\left((e_r'^\Delta + \sum_{q\notin s} e_r'^q)(\sum_{q\in s} e_r''^q)\cdot\delta_{ps}'^\star \;+\; (\sum_{q\in s} e_r'^q)(e_r''^\Delta + \sum_{q\notin s} e_r''^q)\cdot\delta_{ps}''^\star\right)\end{aligned} \tag{5.55}$$

Again, the weight of customer node $i \in C$, $-\sum_{p\in F} a_r^i\cdot e_r'^p\cdot\varphi_{pi}^\star$, requires to know, i.e. to impose, the starting point in the PP. Reduced cost $\overline{c_r}$ can therefore be simplified, and its expression depends on whether the starting point is Δ or a facility $g \in F$.
In the former case we have $e_r'^\Delta = 1$ and $e_r'^p = 0\ \forall p \in F \Rightarrow (\forall p \in F, i \in C)\ b_r^{pi} = 0$; hence, (5.55) becomes:

$$\begin{aligned}\overline{c_r} = \quad & \sum_{ij\in A_0} b_r^{ij}\cdot(d_{ij} - \eta_{ij}^\star) + \sum_{i\in C}\sum_{p\in F} b_r^{ip}\cdot d_{ip} \\ & + \sum_{p\in F} e_r''^p\cdot\theta_p^\star - (1 - e_r''^\Delta)\cdot\theta_\Delta^\star - \xi^\star - \sum_{s\subseteq F}\sum_{p\in s}\delta_{ps}'^\star\cdot\sum_{q\in s} e_r''^q\end{aligned}$$

where $\sum_{s\subseteq F}\sum_{p\in s}\delta_{ps}'^\star\cdot\sum_{q\in s} e_r''^q = \sum_{p\in F} e_r''^p\cdot\sum_{s\in\mathcal{S}_p}\sum_{q\in s}\delta_{qs}'^\star$ (cf (5.25)).
Terms $\sum_{p\in F} e_r''^p\cdot\theta_p^\star - (1 - e_r''^\Delta)\cdot\theta_\Delta^\star$ mean a weight $-\theta_\Delta^\star$ on Δ as starting point, $\theta_\Delta^\star$ on Δ as ending point and $\theta_p^\star$ on $p \in F$ as ending point. Therefore one can put 0 on Δ as both starting and ending point, and $\theta_p^\star - \theta_\Delta^\star$ on each $p \in F$ and write:

$$\begin{aligned}\overline{c_r} = \quad & -\xi^\star + \sum_{ij\in A_0} b_r^{ij}\cdot(d_{ij} - \eta_{ij}^\star) + \sum_{i\in C}\sum_{p\in F} b_r^{ip}\cdot d_{ip} \\ & + \sum_{p\in F} e_r''^p\cdot(\theta_p^\star - \theta_\Delta^\star) - \sum_{p\in F} e_r''^p\cdot\sum_{s\in\mathcal{S}_p}\sum_{q\in s}\delta_{qs}'^\star\end{aligned} \tag{5.56}$$

The modified support graph $G' = (V', A')$ for the PP is:

1. the node set $V' = V \cup \{\Delta', N\}$ has a copy of the depot, Δ', and a dummy end node N;
2. the arc set $A' = A'_0 \cup \bigcup_{i \in C} \{(\Delta', i)\} \cup \left((\Delta, N) \cup \bigcup_{p \in F} \{(p, N)\} \right)$ comprises A'_0 and additional arcs from Δ' to each client, and from each possible real ending point to N;
3. the weights on nodes are the following:

$$w_v^\star = \begin{cases} 0 & (v \equiv i \in C) \\ \theta_p^\star - \theta_\Delta^\star - \sum_{s \in \mathcal{S}_p} \sum_{q \in s} \delta'^\star_{qs} & (v \equiv p \in F) \\ 0 & (v \equiv \Delta) \end{cases}$$

4. the weights of the arcs $ij \in A'$ are:

$$w_{ij}^\star = \begin{cases} d_{\Delta j} - \eta^\star_{\Delta j} + \frac{1}{2} \cdot w_j^\star - \xi^\star & (i \equiv \Delta') \\ d_{ij} - \eta^\star_{ij} + \frac{1}{2} \cdot (w_i^\star + w_j^\star) & (ij \in C \times C) \\ d_{ip} + \frac{1}{2} \cdot w_i^\star + w_p^\star & (i \in C, j \equiv p \in F) \\ d_{i\Delta} - \eta^\star_{i\Delta} + \frac{1}{2} \cdot w_i^\star + w_\Delta^\star & (i \in C, j \equiv \Delta) \\ 0 & (j \equiv N) \end{cases}$$

The PP is solved as an ESPPRC on (V', A') from Δ' to N. One possible strategy consists in taking for each possible *real* ending point $v \in F \cup \{\Delta\}$, the least reduced cost path from the start to v, and adding it to the RMP.
When the starting point of the PP is a facility $g \in F$, we have $e'^g_r = 1$ and $e'^\Delta_r = 0$; moreover, $(\forall p \in F \backslash \{g\})\ e'^p_r = 0 \Rightarrow (\forall p \in F \backslash \{g\}, i \in C)\ b^{pi}_r = 0$. Hence $\overline{c_r}$ is:

$$\begin{aligned} \overline{c_r} = \quad & \sum_{ij \in A_0} b^{ij}_r \cdot (d_{ij} - \eta^\star_{ij}) + \sum_{i \in C} (b^{gi}_r \cdot d_{gi} + \sum_{p \in F} b^{ip}_r \cdot d_{ip}) - \sum_{i \in C} a^i_r \cdot \varphi^\star_{gi} \\ & - (1 - e''^g_r) \cdot \theta^\star_g + \sum_{q \in F \backslash \{g\}} e''^q_r \cdot \theta^\star_q + e''^\Delta_r \cdot \theta^\star_\Delta \\ & - \sum_{\substack{s \subseteq F: \\ g \notin s}} \sum_{p \in s} \left((\sum_{q \in s} e''^q_r) \cdot \delta'^\star_{ps} \right) - \sum_{\substack{s \subseteq F: \\ g \in s}} \sum_{p \in s} \left((e''^\Delta_r + \sum_{q \notin s} e''^q_r) \cdot \delta''^\star_{ps} \right) \end{aligned}$$

where $s \subseteq F : g \in s$ and $s \subseteq F : g \notin s$ imply $s \in \mathcal{S}_g$ and $s \in \overline{\mathcal{S}_g}$, respectively, and θ weights can be corrected as before, so as to obtain:

$$\begin{aligned} \overline{c_r} = \quad & \sum_{ij \in A_0} b^{ij}_r \cdot (d_{ij} - \eta^\star_{ij}) + \sum_{i \in C} (b^{gi}_r \cdot d_{gi} + \sum_{p \in F} b^{ip}_r \cdot d_{ip}) - \sum_{i \in C} a^i_r \cdot \varphi^\star_{gi} \\ & + \sum_{q \in F \backslash \{g\}} e''^q_r \cdot (\theta^\star_q - \theta^\star_g) + e''^\Delta_r \cdot (\theta^\star_\Delta - \theta^\star_g) \\ & - \sum_{s \in \overline{\mathcal{S}_g}} \sum_{p \in s} \delta'^\star_{ps} \cdot \sum_{q \in s} e''^q_r - \sum_{s \in \mathcal{S}_g} \sum_{p \in s} \delta''^\star_{ps} \cdot (e''^\Delta_r + \sum_{q \notin s} e''^q_r) \end{aligned}$$

The last term can be developed as follows (similarly to (5.29)):

$$\begin{aligned} & - \sum_{s\in\overline{\mathcal{S}_g}} \sum_{p\in s} \delta'^{\star}_{ps} \cdot \sum_{q\in s} e''^{q}_{r} - \sum_{s\in\mathcal{S}_g} \sum_{p\in s} \delta''^{\star}_{ps} \cdot (e''^{\Delta}_{r} + \sum_{q\notin s} e''^{q}_{r}) \\ = \quad & - e''^{\Delta}_{r} \sum_{s\in\mathcal{S}_g} \sum_{p\in s} \delta''^{\star}_{ps} + \sum_{q\in F\setminus\{g\}} e''^{q}_{r} \Bigg(- \sum_{\substack{s\notin\mathcal{S}_g \\ s\in\mathcal{S}_q}} \sum_{p\in s} \delta'^{\star}_{ps} - \sum_{\substack{s\in\mathcal{S}_g \\ s\notin\mathcal{S}_q}} \sum_{p\in s} \delta''^{\star}_{ps} \Bigg) \end{aligned}$$

so as to have:

$$\begin{aligned} \overline{c_r} = \quad & \sum_{ij\in A_0} b^{ij}_{r} \cdot (d_{ij} - \eta^{\star}_{ij}) + \sum_{i\in C} (b^{gi}_{r} \cdot d_{gi} + \sum_{p\in F} b^{ip}_{r} \cdot d_{ip}) - \sum_{i\in C} a^{i}_{r} \cdot \varphi^{\star}_{gi} \\ & + e''^{\Delta}_{r} \cdot (\theta^{\star}_{\Delta} - \theta^{\star}_{g} - \sum_{s\in\mathcal{S}_g} \sum_{p\in s} \delta''^{\star}_{ps}) \\ & + \sum_{q\in F\setminus\{g\}} e''^{q}_{r} \cdot (\theta^{\star}_{q} - \theta^{\star}_{g} - \sum_{\substack{s\notin\mathcal{S}_g \\ s\in\mathcal{S}_q}} \sum_{p\in s} \delta'^{\star}_{ps} - \sum_{\substack{s\in\mathcal{S}_g \\ s\notin\mathcal{S}_q}} \sum_{p\in s} \delta''^{\star}_{ps}) \end{aligned} \tag{5.59}$$

The modified support graph $G' = (V', A')$ to solve the PP is built as follows:

1. the node set $V' = V \cup \{g', N\}$ includes a copy g' of g and the dummy arrival node N;
2. the arc set $A' = A'_0 \cup \bigcup_{i\in C} \{(g', i)\} \cup \Big((\Delta, N) \cup \bigcup_{p\in F} \{(p, N)\} \Big)$ is basically A'_0 enhanced with arcs from g' to each client and from each possible real arrival point to N;
3. the weights on the nodes are the following:

$$w^{\star}_{v} = \begin{cases} -\varphi^{\star}_{gi} & (v \equiv i \in C) \\ \theta^{\star}_{q} - \theta^{\star}_{g} - \sum\limits_{\substack{s\notin\mathcal{S}_g \\ s\in\mathcal{S}_q}} \sum\limits_{p\in s} \delta'^{\star}_{ps} - \sum\limits_{\substack{s\in\mathcal{S}_g \\ s\notin\mathcal{S}_q}} \sum\limits_{p\in s} \delta''^{\star}_{ps} & (v \equiv q \in F\setminus\{g\}) \\ 0 & (v \equiv g) \\ \theta^{\star}_{\Delta} - \theta^{\star}_{g} - \sum\limits_{s\in\mathcal{S}_g} \sum\limits_{p\in s} \delta''^{\star}_{ps} & (v \equiv \Delta) \end{cases}$$

4. the weights of the arcs $ij \in A'$ are:

$$w^{\star}_{ij} = \begin{cases} d_{gj} + \frac{1}{2} \cdot w^{\star}_{j} & (i \equiv g') \\ d_{ij} - \eta^{\star}_{ij} + \frac{1}{2} \cdot (w^{\star}_{i} + w^{\star}_{j}) & (ij \in C \times C) \\ d_{ip} + \frac{1}{2} \cdot w^{\star}_{i} + w^{\star}_{p} & (i \in C, j \equiv p \in F) \\ d_{i\Delta} - \eta^{\star}_{i\Delta} + \frac{1}{2} \cdot w^{\star}_{i} + w^{\star}_{\Delta} & (i \in C, j \equiv \Delta) \\ 0 & (j \equiv N) \end{cases}$$

The PP is solved as an ESPPRC on (V', A') from g' to N. Again, for each possible *real* ending point $v \in F \cup \{\Delta\}$, one possible strategy is to add the least reduced cost path from g' to v to the RMP.

5.2.2.2 Branching Rules

As to the branching rules to adopt in our CG-based framework, they can only concern base arc variables x_{ij}, $ij \in A_0$, or replenishment arc variables w_{ij}, $ij \in A_P$. Indeed, route variables x_r cannot be branched on, as it has been explained in section A.4.3. Nor we can branch on activity variables y_p, $p \in F$, as they appear only in redundant constraints (5.40), (5.53) and (5.54) and therefore do not have any actual decisional value.

On the other hand, branching on arc variables requires no transformation, since the link between route and arc variables is now explicit in the MILP model. The chosen branching rule is inherited from the first proposed Branch&Price algorithm for VRPIRF and consists in branching on the outgoing arcs of the customer node with the highest number of fractional outgoing arcs. The rule could be applied on both the base and the replenishment arcs. Moreover, the priority to the former or to the latter arc sets is a parameter of the Branch&Price algorithm that may be tuned by computational experience so as to find the most effective ordering.

The propagation of branching decisions on x_{ij} variables to the PP follows the general scheme given in A.4.3, whereas the branching decisions on w_{ij} variables have no effect on the PP as replenishment arcs do not appear in the graph of the PP.

5.3 Conclusions and Perspectives

In this chapter, two solution methods of type Branch&Price have been proposed for the VRPIRF. The aim was to improve the results achieved with the Branch&Cut algorithm presented in chapter 4.

The first relies on $\mathcal{M}^{BP}$, a Set-Partitioning based MILP formulation of the problem, with an index to distinguish the vehicles and thus enforce the maximum shift length constraint. This gives rise to a Branch&Price algorithm where the subproblem is solved as an ESPPRC using techniques such as *ng*-paths and a *q*-path-based completion bound to accelerate the convergence of the Column Generation performed at each node of the Branch&Bound tree. The algorithm has been fully implemented: however, preliminary results have shown that it behavior is far worse than that of the $\mathcal{M}^{BC}_{RA}$-based Branch&Cut algorithm. Further analysis allow to detect that in spite of adopting a more powerful algorithmic approach as Branch&Price, the method issues originate from some weaknesses of model $\mathcal{M}^{BP}$.

In order to deal with them, a new MILP model, $\mathcal{M}^{BP}_{RA}$, has been designed that merges elements of both models $\mathcal{M}^{BC}_{RA}$ and $\mathcal{M}^{BP}$, in order to remove the vehicle index as it had been done when upgrading model $\mathcal{M}^{BC}$ to $\mathcal{M}^{BC}_{RA}$. A new Branch&Price algorithm has

been designed, from the reduced costs of the new route variables to the definition of new branching rules.

The implementation and test of this latter algorithm, however, are left as next development steps. We believe that the integration of the strong points of model $\mathcal{M}_{RA}^{BC}$, which have give rise a successful Branch&Cut method, into a Branch&Price framework, may deliver even better results.

Apart from this latter task, the perspectives of future work comprise also further developments of the first proposed Branch&Price algorithm. Ad-hoc valid inequalities may be designed to deal with the issues discussed in section 5.1.5 and tighten the lower bound. This would yield a *Branch&Cut&Price* algorithm which appears to be a promising work.

Conclusions and Perspectives

Conclusions and Perspectives

This thesis originates from the *ANR* Project *MODUM* and the City Logistics system it proposes. City Logistics (cf Chapter 1) takes care of studying the dynamic management and operations of urban freight transport, with the aim of delivering distribution systems design solutions that may be suitable for both practitioners and the community, which typically have conflicting objectives. The stakes are considerable, as an improved management of the traffic related to the freight transport can have a positive impact in terms of security, noise and air pollution, congestion of the road network and costs. The achievement of an improved traffic management calls for the design of new models of integrated freight distribution networks, the study of the viability of such systems, the investigation of how to maximize the benefits for all the implied actors, and finally the evaluation of the system impact. Combinatorial Optimization is largely concerned, as the logistic problems that can arise in the urban context are numerous, and the criteria that the different involved actors may want to optimize are several.
The purpose of the MODUM project is the study of a freight distribution system for urban areas based on a *ring of Urban Distribution Centers(UDCs)* located in the outskirts of a city, and the design and implementation of a series of tools that would allow potentially interested subjects to consider the adoption of such a system. Among the different decision layers involved by the study of the proposed distribution system, the strategic level is the one that the first part of this PhD thesis is mostly concerned with.

The study of Combinatorial Optimization techniques to tackle the Location and Network design long-term decisions of the MODUM City Logistics system has given rise to the *Multicommodity-Ring Location Routing Problem (MRLRP)* studied in Chapter 2. In this problem, strategic decisions concerning the location and connection of a set of UDCs must be taken. A set of demands to be served is known. Goods to be delivered arrive to a first UDC from gates, are possibly transported along the ring, and finally shipped; the reverse process occurs for pick-up. No time dependence is considered. Goods are characterized by a quantity, a customer and a gate: the attribute *multicommodity* refers solely to the different gates. Retail shipments are performed by electric vans with both maximum route length and maximum load limits: the fleet is shared among UDC and

SPL, the latter having parking functions only. Service routes can be open, hence a rebalancing policy is imposed to simplify repositioning. The aim is to determine the subset of UDCs to open and the ring connections among them, and to ship every demand from its source to its destination, in such a way to minimize the overall installation, routing and flow transportation costs, while respecting UDC and ring arc capacities. The MRLRP is a rich and complex problem as it involves a number of different, strongly interconnected decision layers (location, network design, vehicle routing, flow transportation). A MILP Set-Partitioning formulation that makes use of route variables is proposed which can be solved efficiently by a Branch&Bound exact method for small-sized instances only.
Given the high difficulty of the problem, we designed *GALW*, a four-stage decomposition matheuristic. GALW tackles the different decisional components of the MRLRP sequentially and solves most of them to optimality. First, a set of near-optimal routes is heuristically generated. Then, an *assignment subproblem* is solved so as to determine the first and last UDC of each demand. A *Symmetric TSP* is then solved to link the chosen UDC, and finally a ring multiflow problem with multiple sources and sinks for each commodity is solved to find the optimal ring flows. GALW performances were assessed by means of an exact algorithm (a Branch&Bound algorithm based on the MILP formulation and solved by a commercial solver) and a hybrid method to evaluate the impact of the route generation. Among the other contributions, a new way to formulate Generalized Subtour Elimination Constraints when none of the nodes is mandatory has been presented. An exhaustive set of 200 instances has been created for this new problem. Instances vary both in terms of dimensional features (notably, the number of UDCs and demands) and in terms of strategic costs to allow a more effective evaluation of the three methods. Further, a set of *ecological instances*, i.e. with pollution indicators instead of economic costs, allow to determine the system configuration with the highest degree of environmental sustainability.
As to future work, on a short- to medium-term horizon, it could be interesting to enrich GALW with some feedback mechanism and hence further improve its performances. Another short-term development could be the design of a heuristic solver for the assignment subproblem, which is the most time-consuming. This could lead to a dramatic reduction of the total computational time of GALW and to an overall better ratio between solution quality and computational time.

The *Multicommodity-Ring Vehicle Routing Problem (MRVRP)* presented in Appendix A is a more tactical problem based on the same distribution system and can be considered to all purposes as one of the main future work perspectives in the same stream. For this problem, an exact algorithm is more likely to be considered. Therefore, after defining a new, more tailored Set-Partitioning formulation, a Branch&Price solution approach is proposed, where the *Pricing Problem (PP)* is solved as an *Elementary Shortest Path*

Problem with Resource Constraints (ESPPRC). Its implementation and test represent one of the most forthcoming development tasks after the thesis.
The MODUM ring-of-UDC-based urban freight distribution system can give rise to several other variants of strategic and/or tactical problems. For example, one could consider the intermediate problem in which the ring is somehow defined (because e.g. the city is already surrounded by a motorway that can be reused) but the UDC are to be located. Such problem would stand somehow in the middle between MRLRP and MRVRP as it features location decision but not network design elements, and would probably require to design a new matheuristic.

The second part of this thesis tackles the *Vehicle Routing Problem with Intermediate Replenishment Facilities (VRPIRF)*. The VRPIRF features a *central depot*, n *customers* and f *replenishment facilities*. The aim is to find a least cost set of routes that visits each client exactly once, the cost of a route being the sum of the costs of the visited arcs. Each client has a *demand* and can be served by one of the homogeneous, fixed capacity *vehicles* based at the depot. Furthermore, vehicles can recharge at replenishment facilities so as to perform not one but a sequence of routes called a *rotation*. However, the rotation of a vehicle must start and end at the depot and its total *duration* (the sum of the travel, service and recharge times associated with the visited arcs, clients, and depots, respectively) must not exceed a given shift length.
The study of this problem is suggested by some more operational decision issues thay may occur in a City Logistics context like the one of the MODUM project. More specifically, the VRPIRF arises when both the *Multi-Depot* and the *Multi-Trip* requirements are present, i.e. there is more than one facility that can act as a starting point for shipment trips, and vehicles can perform more than one trip a day. Therefore, the exhaustive literature review on the VRPIRF proposed in Chapter 3 starts from the *Multi-Depot VRP (MDVRP)* and the *Multi-Trip VRP (MTVRP)*, before presenting several research work inspired by numerous real-life applications in which the most prominent VRPIRF features are found. Most of these works result in heuristic approaches, also motivated by the complexity of the case studies. We mostly focused on the design of exact approaches to the VRPIRF, as –to the best of our knowledge– there exist only a few works that have investigated this path.

Chapter 4 revolves around two algorithms of type Branch & Cut, based on compact MILP formulations of the problem. The first one is based on a classical three-index formulation where vehicles are explicitly identified by means of a vehicle index and is capable to solve small-sized instances. In order to deal with bigger instances, a second Branch & Cut algorithm has been designed which is far more clever in two respects: the MILP model and the separation techniques and strategy. The MILP model on which the second algorithm is based, $\mathcal{M}_{RA}^{BC}$, incorporates two powerful modeling tools:

replenishment arcs and *arrival times*. Together, they allow to drop the vehicle index thanks to a substantial change in the representation of a rotation. Both have been previously used in the literature: the former for the MTVRP and a variant of the ESPPRC, the latter for the *Asymmetric Distance-Constrained VRP*. However, to the best of our knowledge, both of them are applied for the first time to the VRPIRF. The separation policy of the second Branch&Cut is simple, as the only classical *capacity* and *subtour elimination constraints* are separated. Yet, it proves effective, as the algorithm has shown an interesting behavior on most of the 76 benchmark instances taken from the literature, proving capable of achieving root gaps under 10% in 71 cases, and under 5% in 36 cases within reasonable time limits. It can therefore be considered very promising. To improve the algorithm based on $\mathcal{M}_{RA}^{BC}$ and the separation policy two main ideas appear to be viable. The first one consists in enhancing the separation procedure with other cuts or valid inequalities taken from the literature. The second one is to study more specific, problem-tailored cuts that take into account, for instance, the arrival times. Since many benchmark instances in the literature are symmetric, one could even consider to reformulate the VRPIRF in a symmetric fashion. The design of ad-hoc inequalities to prevent some problem issues would be probably easier then.

In order to broaden our study of exact approaches to the VRPIRF and to improve the results achieved with the Branch&Cut algorithms, two Branch&Price approaches are also designed for the same problem. They are introduced in Chapter 5. We rely on Set-Partitioning MILP formulations with route variables, as in recent years exact algorithms based on such formulations have proven to be the most effective ones for many Vehicle Routing Problems. The MILP model for the first Branch&Price algorithm, $\mathcal{M}^{BP}$, uses a vehicle index in order to enforce the maximum shift length constraint, and has ad-hoc connectivity constraints to assure rotations to be connected. These latter constraints are formulated on a *facility graph* so as to considerably reduce their number. This allows to generate them statically instead of adding them dynamically. The PP is solved as an ESPPRC by *Dynamic Programming*, which is enhanced with two very effective techniques such as *ng*-paths and a *q*-path-based completion bound. This allows to restrain combinatorial explosion and accelerate the convergence of the Column Generation performed at each node of the Branch&Bound tree. Although the method is very promising, the preliminary computational results have been worse than those of the $\mathcal{M}_{RA}^{BC}$-based Branch&Cut algorithm. Further analysis have shown a weakness in the connectivity constraints of $\mathcal{M}^{BP}$ that loosens the lower bound. Ad-hoc valid inequalities may be designed to deal with these issues. This would yield a *Branch&Cut&Price* algorithm which appears to be a very promising work.
The study of a second Branch&Price algorithm, which is based on a new extended MILP formulation, $\mathcal{M}_{RA}^{BP}$, is also undertook. $\mathcal{M}_{RA}^{BP}$ aims at tackling some issues of its predecessor

by removing the vehicle index and merging $\mathcal{M}^{BP}$ with replenishment arcs and arrival times of $\mathcal{M}^{BC}_{RA}$ as it has been done with the $\mathcal{M}^{BC}_{RA}$-based Branch&Cut method. A new Branch&Price algorithm is designed, from the reduced costs of the new route variables to the definition of new branching rules. However, the implementation and numerical assessment for this latter Branch&Price algorithm are left as a perspective for future work on a short-term horizon. We believe that this last approach is the most promising, as it integrates the strong points of model $\mathcal{M}^{BC}_{RA}$, which have give rise a successful Branch &Cut method, into a Branch&Price framework.

During this PhD thesis, other minor research works have been conducted. More specifically, they deal with some problems related to the assignment of computer tasks to machines. Therefore, they have been interesting opportunities to cope with Combinatorial Optimization applications in a rather different domain than the main one of the thesis.
Appendix B tackles the *Machine Reassignment Problem* which has been the object of the *Google ROADEF/EURO Challenge 2011–2012*. An initial allocation of processes to machines is given, and another one must be found that minimizes a global movement cost while respecting a series of machine resource constraints and dependency/compatibility constraints among processes. A hybrid method based on *local search*, *MILP driven search* and *superprocesses* is developed to deal with the small- to big-sized instances proposed for the Challenge.
Appendix C presents a work on some problems arising in *Grid* and *Volunteer Cloud Computing* about the optimization of Energy. Tasks must be allocated according to the capacity and availability of the machines. The allocation must cover a given time horizon and minimize the overall sum of *base*, *overhead* and *transfer* energy.

To conclude, we briefly outline what in our opinion are the most interesting longer-term perspectives.
The intractability of the MRLRP with medium- and large-sized instances naturally led to the development of a heuristic approach. As it has been seen in the parts of the thesis dedicated to the review of Vehicle and Location Routing Problems (see sections 2.2, A.2, 3.2), heuristic approaches to Combinatorial problems often rely on local search techniques (*Tabu Search (TS)*, *Variable Neighborhood Search (VNS)*, *Adaptive Variable Neighborhood Search (AVNS)*, *Adaptive Large Neighborhood Search (ALNS)*). However, it appeared extremely difficult to tackle the MRLRP by means of such techniques, as the multi-layered decision structure and the presence of both integer and continuous variables make it difficult to define suitable neighborhood structures. This opens some perspectives of investigation about the suitability of local search methods for rich, large-scale, multi-layered problems and whether it should be preferrable to adopt matheuristic

decomposition approaches that tackle the different decision layers in sequence and solve each to optimality.

The work on the VRPIRF has left several development tasks to be accomplished shortly after the end of this thesis work, but also some interesting, more general open research paths that will be investigated on a longer-term horizon.
The second Branch&Cut algorithm, as well as the first Branch&Price one, require the study of problem-tailored valid inequalities in order to improve the results achieved so far. This encourages an in-depth study of *valid inequalities* as a general modeling and algorithmic tool on a medium-term basis.
The same work also suggests another longer-term research path. According to what has been achieved by the most prominent exact methods of the recent literature on Vehicle Routing Problems, the first Branch&Price algorithm was expected to yield the best results. Surprisingly enough, the best results that we have obtained so far on the benchmark VRPIRF instances have been attained by what appeared to be the weakest method, i.e. a Branch&Cut algorithm based on a compact formulation. The reason resides in the major strength of compact MILP model $\mathcal{M}_{RA}^{BC}$ w.r.t. the extended MILP model $\mathcal{M}^{BP}$. Although we believe that the implementation and computational assessment of the $\mathcal{M}_{RA}^{BP}$-based Branch&Price algorithm will provide more problem-specific insights, this consideration lead us, in a more general sense, to further investigate the trade-off between formulation and method as the *key factor* to deliver the most effective exact approach.

Appendix A

The Multicommodity-Ring Vehicle Routing Problem

A.1 Introduction

The MRLRP, which has been extensively presented in chapter 2, is a problem with a high degree of difficulty, due to its several decision levels and the deep relations between them. Therefore, in spite of being a strategic problem, where operational details are either aggregated or simplified, the MRLRP can hardly be solved, unless the instances being dealt with are small-sized, or the chosen approach is heuristic.
However, a family of related problems can be derived from the MRLRP when considering only a part of the decisions involved by the discussed urban distribution system. The *Multicommodity-Ring Vehicle Routing Problem (MRVRP)* that we present in this chapter is the problem obtained when the UDCs have already been built and connected in a ring. Hence, the analysis is no more concerned by strategic decisions and the related costs. The scenario remains the same in every other respect, though. Goods to be delivered arrive to a first UDC from gates, are possibly transported along the ring, and finally shipped; the reverse process occurs for pick-up. No time dependence is considered. Goods are characterized by a quantity, a customer and a gate: once more, the attribute *multicommodity* refers solely to the different gates. Retail shipments are performed by electric vans with both maximum route length and maximum load limits: the fleet is shared among UDC and SPL, the latter having parking functions only. Service routes can be open, hence a rebalancing policy is imposed to simplify repositioning. The objective is to ship every demand, be it a delivery or a pick-up one, from its source to its destination, in such a way to minimize the overall routing and flow transportation

costs, while respecting UDC and ring arc capacities.
The purpose of this chapter is to tackle a derived problem of MRLRP, the MRVRP, with the aim to define the guidelines of an exact method for it. The rest of the chapter is therefore structured as follows. Section A.2 gives some more literature insights for what concerns problems that are more related to the MRVRP. Section A.3 defines additional notations and a MILP model for the MRVRP. Section A.4 outlines a Column-Generation-based approach to the MRVRP and a Branch&Price algorithm derived from it. Finally, section A.5 draws the conclusions of the chapter.

A.2 Positioning in the Literature

The MRVRP can be considered to belong to the family of the *Multi-level VRPs*. Differently from canonical VRPs, where the vehicles that visit the final customers start from a central depot, in Multi-level VRPs goods are dispatched to intermediate depots before reaching their final destination. The motivation to study such more complex distribution systems comes mainly from real-world traffic restrictions that prevent big trucks to enter the city center. This imposes transshipment or cross-docking operations outside the urban area, in order to load smaller vehicles to perform shipments to retailers.
The most studied problem is the *Two-Echelon Vehicle Routing Problem (2E-VRP)*, in which goods are initially stored at a central *warehouse*, from where they are delivered to secondary-level *logistic platforms* or *satellites*, which we will refer to as UDCs (i.e. Urban Distribution Center) as we have done in chapter 2. After being consolidated in second-level vehicles, products can finally be shipped to customers. Split delivery are forbidden at second-level, but allowed at first-level, therefore a UDC can receive the merchandise it has to deliver from many first-level vehicles. UDCs have a capacity that bounds the first-level deliveries. Second-level vehicles can only perform one service route and must return at the depot from where they started. In more general versions, other features can be taken into account, like e.g.:

- more than one warehouse;
- the possibility to deliver customers via first-level vehicles, to represent situations in which the restrictions about the circulation of big trucks are weaker;
- second level vehicles can perform more than one service trip (multi-trip feature) which can have different endpoints;
- time dependent travel times;
- customer time windows;

- synchronization between first- and second-level, whereas in the classical 2E-VRP the only interaction is the capacity of the UDCs.

Several heuristic approaches to the 2E-VRP can be found in the literature. In [Crainic et al., 2008], a set of two-phase heuristics are proposed. The second-level subproblem is solved as a *Multi-Depot VRP (MDVRP)*, or alternatively as a set of small VRPs after a clustering of the customers in order to assign them to UDCs. Then, the first level subproblem is solved as a common CVRP. In the second phase, a series of heuristics are used to improve the solution. In [Crainic et al., 2011b], the problem is approached in a similar way. A greedy initial clustering heuristic is used to decompose the problem in as many CVRPs as the number of UDCs plus one, i.e. the first-level problem. Then, a local search step changes the customer-UDC assignment so as to improve the solution. Finally, a multi-start phase is applied for a given number of iterations: the current best solution is perturbed according to *savings*-inspired criteria, yielding either an infeasible solution, which is then repaired, or a feasible one. In the latter case, if the quality is promising, the solution in further improved by means of the customer-UDC assignment improvement LS tool. To mention other heuristic algorithms, we refer the reader for instance to the GRASP with path-relinking of [Crainic et al., 2011a] or the ALNS proposed in [Cordeau et al., 2011]. The literature of exact method is more scarce. Among the most recent works, the exact algorithm presented in [Baldacci et al., 2013] decomposes the 2E-VRP into a set of MDVRP with additional constraints. Valid lower bounds are provided which allow to restrict the search. The algorithm achieves the best computational results to date. We also mention the Branch&Cut approach of [Jepsen et al., 2013], the flow model and valid inequalities for the 2E-VRP presented in [Gonzalez-Feliu et al., 2008], and the study on valid inequalities for the 2E-VRP of [Masoero et al., 2009]. Finally, we refer the reader to [Gonzalez-Feliu, 2011] for a recent survey on two-level distribution systems.

A.3 A Mixed-Integer Linear Programming (MILP) formulation for the MRVRP

Compared to MRLRP, the MRVRP is a more tactical problem where location and network aspects are known a priori. Therefore, the MILP model that we propose for the MRVRP will keep only some of the elements of the one that has been presented for the MRLRP (see section 2.4).

A.3.1 Additional Notation

Most of the notations defined for the MRLRP will be used again in the following description of MRVRP and with the same meaning. Hence, we refer the reader to section 2.3 or to appendix D.1 and restrict ourselves to highlight the differences and the new notations. The node set $U = \{u_1,\dots,u_{|U|}\}$, represents now a set of *existing* UDCs instead of a set of *potential locations*. Each $u \in U$ is therefore characterized by its capacity Q_u only. Moreover, for ease of notation, we index the UDCs according to their order along the ring. We introduce the symbols $s(u)$ and $p(u)$, $u \in U$, to denote, respectively, the *successor* of u, i.e. the UDC that follows u in the sense of the increasing UDC indices, and the *predecessor* of u, i.e. the UDC $v \in U$ s.t. $s(v) = u$. The oriented subgraph (U, A_U) is no longer complete, as now A_U contains only $2|U|$ *existing* ring arcs:

$$A_U = \bigcup_{u \in U} \{(u, p(u)), (u, s(u))\}$$

Finally, we introduce some binary coefficients to characterize a service route $r \in \mathscr{R}$:

- $d_r = 1 \Leftrightarrow r \in \mathscr{R}^d$, i.e. r is a delivery route, $d_r = 0 \Leftrightarrow r \in \mathscr{R}^p$, i.e. r is a pick-up route;
- $a_r^i = 1 \Leftrightarrow i \in R_r$, defined for $i \in D$ and $d_r = 1$, or $i \in P$ and $d_r = 0$;
- $e_r^{+h} = 1 \Leftrightarrow r$ starts at $h \in U \cup L$ (i.e. h UDC or SPL), not defined for $h \in L$ if $d_r = 1$;
- $e_r^{-h} = 1 \Leftrightarrow r$ ends at $h \in U \cup L$, not defined for $h \in L$ if $d_r = 0$;
- $b_r^{ij} = 1 \Leftrightarrow (i,j) \in E_r$, defined for $(i,j) \in E^d$ and $d_r = 1$, or $(i,j) \in E^p$ and $d_r = 0$.

A.3.2 Decision variables

The decision variables of the MRVRP are a subset of those of the MRLRP. We discard ring variables, which are related to strategic decisions, and keep the other three families of variables, which we recall here briefly:

1. binary *service variables* $\chi_{ku} = 1$ if gate k exchanges goods with UDC u, 0 otherwise;
2. binary second level *routing variables* $x_r = 1$ if route $r \in \mathscr{R}$ is selected, 0 otherwise;
3. first level *flow variables*, which are all nonnegative and continuous:
   - φ_{ku} = flow from gate k to site u (k-outflow);
   - φ_{uk} = flow from site u to gate k (k-inflow);
   - φ_{uv}^{dk} = k-outflow on the ring arc $(u,v) \in A_U$;
   - φ_{uv}^{pk} = k-inflow on the ring arc $(u,v) \in A_U$;
   - ϕ_{ku} = upper bound on the capacity occupied at $u \in U$ due to deliveries of $k \in K$;

► ϕ_{uk} = upper bound on the capacity occupied at $u \in U$ due to pick-ups of $k \in K$.

As a reminder, *k-outflows* and *k-inflows* are flows of commodity k for, respectively, delivery and pick-up purposes. They are identified by the d and p superscripts.

A.3.3 MILP formulation

The proposed model for the MRVRP is:

$$(\mathcal{M}^{MRVRP})$$

$$\min \sum_{r\in\mathscr{R}} c(r)x_r + \sum_{\substack{k\in K\\ u\in U}} (c_{ku}\varphi_{ku} + c_{uk}\varphi_{uk}) + \sum_{\substack{(u,v)\in A_U\\ k\in K}} c_{uv}(\varphi^{pk}_{uv} + \varphi^{dk}_{uv}) \tag{A.1}$$

$$\text{s.t.} \sum_{u\in U} \varphi_{ku} = \sum_{i\in D_k} q_i \qquad \forall k \in K \tag{A.2}$$

$$\sum_{u\in U} \varphi_{uk} = \sum_{i\in P_k} q_i \qquad \forall k \in K \tag{A.3}$$

$$\varphi_{uk} + \varphi_{ku} \le \chi_{ku} \sum_{i\in P_k\cup D_k} q_i \qquad \forall k \in K, u \in U \tag{A.4}$$

$$\sum_{u\in U} \chi_{ku} \le B \qquad \forall k \in K \tag{A.5}$$

$$\varphi_{ku} + \varphi^{dk}_{p(u)u} + \varphi^{dk}_{s(u)u} = \varphi^{dk}_{up(u)} + \varphi^{dk}_{us(u)} + \sum_{r\in\mathscr{R}} d_r e^{+u}_r q_k(r) x_r \qquad \forall k \in K, u \in U \tag{A.6}$$

$$\sum_{r\in\mathscr{R}} (1-d_r) e^{-u}_r q_k(r) x_r + \varphi^{pk}_{p(u)u} + \varphi^{pk}_{s(u)u} = \varphi^{pk}_{up(u)} + \varphi^{pk}_{us(u)} + \varphi_{uk} \qquad \forall k \in K, u \in U \tag{A.7}$$

$$\sum_{r\in\mathscr{R}} a^i_r x_r = 1 \qquad \forall i \in P \cup D \tag{A.8}$$

$$-\delta^-_h \le \sum_{r\in\mathscr{R}} (e^{-h}_r - e^{+h}_r) x_r \le \delta^+_h \qquad \forall h \in U \cup L \tag{A.9}$$

$$\sum_{k\in K} (\varphi^{dk}_{uv} + \varphi^{pk}_{uv}) \le q_{uv} \qquad \forall (u,v) \in A_U \tag{A.10}$$

$$\sum_{k\in K} (\phi_{ku} + \phi_{uk}) \le Q_u \qquad \forall u \in U \tag{A.11}$$

$$\phi_{ku} \ge \varphi_{ku} \qquad \forall k \in K, u \in U \tag{A.12}$$

$$\phi_{ku} \ge \sum_{r\in\mathscr{R}} d_r e^{+u}_r q_k(r) x_r \qquad \forall k \in K, u \in U \tag{A.13}$$

$$\phi_{uk} \ge \varphi_{uk} \qquad \forall k \in K, u \in U \tag{A.14}$$

$$\phi_{uk} \ge \sum_{r\in\mathscr{R}} (1-d_r) e^{-u}_r q_k(r) x_r \qquad \forall k \in K, u \in U \tag{A.15}$$

$$\chi_{ku} \in \{0,1\} \qquad \forall k \in K, u \in U$$

$$x_r \in \{0,1\} \qquad \forall r \in \mathscr{R}$$

$$\varphi_{ku}, \varphi_{uk}, \phi_{ku}, \phi_{uk} \ge 0 \qquad \forall k \in K, u \in U$$

$$\varphi^{dk}_{uv}, \varphi^{pk}_{uv} \ge 0 \qquad \forall k \in K, (u,v) \in A_U$$

The objective function (A.1) takes into account the routing costs and the costs to transport flows of goods between gates and UDC and along the ring. Constraints (A.2) and (A.3) ensure that for each $k \in K$ the total k-outflow and k-inflow sum up to the overall delivery and pick-up demands in which k is involved. However, the flows that k exchanges with each $u \in U$ is null if $\chi_{ku} = 0$, as imposed by (A.4), and the number of UDCs a gate k can address is bounded to B by (A.5). Constraints (A.6) and (A.7) are flow balance equations on each $u \in U$ and for the k-inflows and k-outflows of each commodity. Relations (A.8) assure that each demand is served by exactly one route $r \in \mathscr{R}$. Constraints (A.9) are needed for rebalancing purposes. Constraints (A.10) impose the capacity bound of ring arcs, and the same do relations (A.11)–(A.15) w.r.t. the capacity of UDCs, as it has been explained in section 2.4.2.

A.4 Column Generation-based approaches for the MRVRP

In this section we present two solution approaches based on the MILP model $\mathcal{M}^{MRVRP}$. $\mathcal{M}^{MRVRP}$, like $\mathcal{M}^{BP}$ (see 5.1.2), is a model that contains a class of integer variables, the route variables $x_r \in \mathscr{R}$, whose number $|\mathscr{R}| = \mathcal{O}(|P|! + |D|!)$ is exponential in the size of the problem instance. Once more, the most suitable approach is based on Column Generation (CG). In the following, we outline the basic components of our CG scheme.

A.4.1 Reduced costs

The first step is to determine the reduced cost of route variables, before deciding how to solve the PP. The reduced cost of x_r is:

$$\begin{aligned}\overline{c_r} = c(r) &- \sum_{\substack{k\in K\\ u\in U}} \left(d_r \cdot e_r^{+u} \cdot q_k(r) \cdot \alpha_{ku}^{d\star} \;+\; (1-d_r) \cdot e_r^{-u} \cdot q_k(r) \cdot \alpha_{ku}^{p\star} \right) \\ &- \sum_{i\in P\cup D} a_r^i \cdot \beta_i^\star - \sum_{h\in U\cup L} (e_r^{-h} - e_r^{+h})(\theta_h'^\star + \theta_h''^\star) \\ &- \sum_{\substack{k\in K\\ u\in U}} \left(d_r \cdot e_r^{+u} \cdot q_k(r) \cdot \gamma_{ku}^{d\star} \;+\; (1-d_r) \cdot e_r^{-u} \cdot q_k(r) \cdot \gamma_{ku}^{p\star} \right)\end{aligned}$$

where $\beta_i^\star$, $\gamma_{ku}^{d\star}$, $\gamma_{ku}^{p\star}$, $\alpha_{ku}^{d\star}$, $\alpha_{ku}^{p\star}$, $\theta_h'^\star$ and $\theta_h''^\star$, are the values of the dual variables associated with the current solution of the *Restricted Master Problem* (*RMP*, see section 5.1.2.1) and, respectively, to constraints (A.8), (A.13), (A.15), (A.6), (A.7) and the two families

of inequalities of (A.9). The expression of $\overline{c_r}$ can be rewritten as:

$$\begin{aligned}\overline{c_r} = \sum_{(i,j)\in E} b_r^{ij}\cdot c_{ij} - \sum_{i\in P\cup D} a_r^i\cdot\beta_i^\star \\ - d_r\Big(\sum_{\substack{k\in K\\ u\in U}}(\alpha_{ku}^{d\star}+\gamma_{ku}^{d\star})\cdot q_k(r)\cdot e_r^{+u}\Big) \\ - (1-d_r)\Big(\sum_{\substack{k\in K\\ u\in U}}(\alpha_{ku}^{p\star}+\gamma_{ku}^{p\star})\cdot q_k(r)\cdot e_r^{-u}\Big) \\ - \sum_{h\in U\cup L}(e_r^{-h}-e_r^{+h})(\theta_h'^\star+\theta_h''^\star)\end{aligned} \tag{A.16}$$

The above expression can be seen as the cost of route r on a graph which is identical to the initial support graph but with additional weights on nodes. These weights depend on the dual variables associated with the constraints in which r is involved, and can be seen as *prizes* or *penalties* according to their contribution to the reduced cost $\overline{c_r}$. Here are some examples:

- term $a_r^i\cdot\beta_i^\star$ means that a weight $\beta_i^\star$ must be put on client $i\in P\cup D$;
- term $\sum_{\substack{k\in K\\ u\in U}}(\alpha_{ku}^{d\star}+\gamma_{ku}^{d\star})\cdot q_k(r)$ is multiplied by $d_r\cdot e_r^{+u}$ and must therefore be taken into account only in case r is a delivery route that starts from u;
- a term $\theta_h'^\star+\theta_h''^\star$ must be added when $h\in U\cup L$ is the starting point of r (i.e. when $e_r^{+h}=1$), and inversely a term $-(\theta_h'^\star+\theta_h''^\star)$ must be added when h is the ending point of r (i.e. $e_r^{-h}=1$). Note that one between this two terms will be a prize and the other a penalty, according to which one between $\theta_h'^\star\geq 0$ and $\theta_h''^\star\leq 0$ has the greatest absolute value.

The weight of a node, however, can be conveniently added to the weight c_{ij} of each of its ingoing or outgoing arc, in order to have again a problem with costs on arcs only. Therefore, solving the PP calls for finding a route r with least $\overline{c_r}$ value. Since the solution routes must respect a load constraint, this amounts to solving an *Elementary Shortest Path Problem with Resource Constraints (ESPPRC)* on the aforementioned modified support graph.

As one can observe, some parts of relation (A.16) depend on the terminal points of route r, i.e. the solution of the PP. Let us denote by h_1 and h_2 the starting and ending point of r. Since both of them can be either a UDC or a SPL, but not both a SPL, we can distinguish four cases:

1. $h_1\equiv u\in U, h_2\equiv l\in L$: r can only be a delivery route (see section 2.1). Therefore, the only clients it can visit are delivery clients, and we consider a *delivery subgraph*

where pick-up clients and edges are discarded. The reduced cost becomes:

$$\overline{c_r}^d = \sum_{(i,j)\in E^d} b_r^{ij}\cdot c_{ij} - \sum_{k\in K}(\alpha_{ku}^{d\star}+\gamma_{ku}^{d\star})\cdot q_k(r) - \sum_{i\in D}\beta_i^\star\cdot a_r^i + (\theta_u'^\star+\theta_u''^\star) - (\theta_l'^\star+\theta_l''^\star) \quad \text{(A.17)}$$

Since $q_k(r)$ is the sum of demands of clients of k visited by r, term $\sum_{k\in K}(\alpha_{ku}^{d\star}+\gamma_{ku}^{d\star})\cdot q_k(r)$ is equivalent to putting on each customer node $i\in D$ of the delivery subgraph a weight equal to $-(\alpha_{k_iu}^{d\star}+\gamma_{k_iu}^{d\star})\cdot q_i-\beta_i^\star$, k_i being the gate of which i is client.
Terms $(\theta_u'^\star+\theta_u''^\star)$ and $-(\theta_l'^\star+\theta_l''^\star)$ must be put respectively on u and l, but since these are the designated starting and ending point, they turn out to be constant terms of the PP;

2. $h_1\equiv l\in L, h_2\equiv u\in U$: analogously, r can only be a pick-up route. We can consider a *pick-up subgraph* and the reduced cost becomes:

$$\overline{c_r}^p = \sum_{(i,j)\in E^p} b_r^{ij}\cdot c_{ij} - \sum_{k\in K}(\alpha_{ku}^{p\star}+\gamma_{ku}^{p\star})\cdot q_k(r) - \sum_{i\in P}\beta_i^\star\cdot a_r^i + (\theta_l'^\star+\theta_l''^\star) - (\theta_u'^\star+\theta_u''^\star) \quad \text{(A.18)}$$

This time the weight to add to each customer node of the pick-up subgraph is $(\alpha_{k_iu}^{p\star}+\gamma_{k_iu}^{p\star})\cdot q_i-\beta_i^\star$, while the constant of the PP is $\theta_l'^\star+\theta_l''^\star-\theta_u'^\star-\theta_u''^\star$;

3. $h_1\equiv u\in U, h_2\equiv v\in U$, r delivery route ($r\in\mathscr{R}^d$): the expression of the reduced cost is similar to (A.17), the only difference being the constant term:

$$\overline{c_r}^d = \sum_{(i,j)\in E^d} b_r^{ij}\cdot c_{ij} - \sum_{k\in K}(\alpha_{ku}^{d\star}+\gamma_{ku}^{d\star})\cdot q_k(r) - \sum_{i\in D}\beta_i^\star\cdot a_r^i + (\theta_u'^\star+\theta_u''^\star) - (\theta_v'^\star+\theta_v''^\star)$$

4. $h_1\equiv v\in U, h_2\equiv u\in U$, r pick-up route ($r\in\mathscr{R}^p$): the reduced cost expression is similar to (A.18), with a different constant term:

$$\overline{c_r}^p = \sum_{(i,j)\in E^p} b_r^{ij}\cdot c_{ij} - \sum_{k\in K}(\alpha_{ku}^{p\star}+\gamma_{ku}^{p\star})\cdot q_k(r) - \sum_{i\in P}\beta_i^\star\cdot a_r^i + (\theta_v'^\star+\theta_v''^\star) - (\theta_u'^\star+\theta_u''^\star)$$

A comparison of the above cases 1 and 3 shows that, whenever the route r to be built is a delivery one, the only dependence from the ending point h_2 is in the term $-(\theta_{h_2}'^\star+\theta_{h_2}''^\star)$. This reflects a structural property of the problem: the ending point of a delivery route is only affected in terms of rebalancing (constraints (A.9)), as it has been explained in section 2.5.1. Therefore cases 1 and 3 can be unified, and analogous conclusions can be drawn by comparing cases 2 and 4.
However, the dependence of the reduced cost $\overline{c_r}^d$ of a delivery route from its starting UDC cannot be disregarded, and imposes to solve $|U|$ *delivery PPs*, i.e. one per UDC, when seeking for negative reduced cost delivery routes. Similarly, to determine the existence of pick-up routes with $\overline{c_r}^p<0$, we have to solve $|U|$ *pick-up PPs*. As a consequence, at

each CG iteration we have to solve $2|U|$ PPs.
The modified non-oriented support graph G_u^d for the delivery PP with $u \in U$ as starting point is defined as follows:

1. the node set $U \cup L \cup D \cup \{\hat{u}, 0\}$ includes a copy $\hat{u}$ of u and a dummy arrival node 0;
2. the edge set $E_u^d = \bigcup_{i \in D} \{(\hat{u}, i)\} \cup E^d \cup \bigcup_{h \in U \cup L} \{(h, 0)\}$ is enhanced with arcs from $\hat{u}$ to each delivery client, and from each possible *real* arrival point to 0;
3. the weights to put on nodes are the following:

$$w_v^\star = \begin{cases} -(\alpha_{k_i u}^{d\star} + \gamma_{k_i u}^{d\star}) \cdot q_i - \beta_i^\star & (v \equiv i \in D) \quad \text{(A.19a)} \\ -\theta_h'^\star - \theta_h''^\star & (v \equiv h \in U \cup L) \quad \text{(A.19b)} \\ \theta_u'^\star + \theta_u''^\star & (v \equiv \hat{u}) \quad \text{(A.19c)} \end{cases}$$

This is equivalent to put the following weights on the arcs $(i, j) \in E_u^d$:

$$w_{ij}^\star = \begin{cases} c_{uj} + w_{\hat{u}}^\star + \frac{1}{2} \cdot w_j^\star & (i \equiv \hat{u},\ j \in D) \quad \text{(A.20a)} \\ c_{ij} + \frac{1}{2} \cdot w_i^\star + \frac{1}{2} \cdot w_j^\star & (i, j \in D) \quad \text{(A.20b)} \\ c_{ih} + \frac{1}{2} \cdot w_i^\star + w_h^\star & (i \in D,\ h \in U \cup L) \quad \text{(A.20c)} \\ 0 & (j \equiv 0) \quad \text{(A.20d)} \end{cases}$$

The PP is then solved as an ESPPRC on G_u^d from $\hat{u}$ to 0. For each possible ending point $h \in U \cup L$, the least reduced cost path from $\hat{u}$ to h (provided that it has $\overline{c_r}^d < 0$), or a subset of negative reduced cost paths, is added to the RMP.
The pick-up PP is approached in a slightly different manner. Solution routes are sought for *backward*, i.e. from the fixed endpoint u towards the possible start points in $U \cup L$, in order to possibly generate a path for each possible starting point. The modified support graph G_u^p is constructed as follows:

1. the node set $U \cup L \cup P \cup \{\hat{u}, 0\}$ includes a dummy starting node 0 and a copy $\hat{u}$ of u;
2. the edge set $E_u^p = \bigcup_{h \in U \cup L} \{(0, h)\} \cup E^p \cup \bigcup_{i \in P} \{(i, \hat{u})\}$ is enhanced with arcs from 0 to each possible *real* starting point, and from each pick-up client to $\hat{u}$;
3. the weights on nodes are the following:

$$w_v^\star = \begin{cases} -(\alpha_{k_i u}^{p\star} + \gamma_{k_i u}^{p\star}) \cdot q_i - \beta_i^\star & (v \equiv i \in P) \quad \text{(A.21a)} \\ \theta_h'^\star + \theta_h''^\star & (v \equiv h \in U \cup L) \quad \text{(A.21b)} \\ -\theta_u'^\star - \theta_u''^\star & (v \equiv \hat{u}) \quad \text{(A.21c)} \end{cases}$$

again, this is equivalent to put the following weights on the arcs $(i,j) \in E_u^p$:

$$w_{ij}^\star = \begin{cases} 0 & (i \equiv 0) & \text{(A.22a)} \\ c_{hi} + w_h^\star + \frac{1}{2}\cdot w_i^\star & (i \in D,\, h \in U \cup L) & \text{(A.22b)} \\ c_{ij} + \frac{1}{2}\cdot w_i^\star + \frac{1}{2}\cdot w_j^\star & (i,j \in D) & \text{(A.22c)} \\ c_{iu} + \frac{1}{2}\cdot w_i^\star + w_{\hat{u}}^\star & (i \in P,\, j \equiv \hat{u}) & \text{(A.22d)} \end{cases}$$

The PP is solved backward as an ESPPRC on G_u^p from $\hat{u}$ to 0. Again, the most negative reduced cost path, or a subset of negative reduced cost paths, is added to the RMP for each possible starting point.
Hence, at each CG iteration we add up to $2|U|(|U|+|L|)$ new route variables to the MP.

A.4.2 A Column Generation-based heuristic algorithm

The CG-based procedure LPcgLowerBound in figure A.1 computes the optimal value of the LP relaxation of model $\mathcal{M}$ with a starting set $\overline{\mathscr{R}_0}$ of columns (route variables). $\mathcal{M}$ is the model associated with a node of the Branch&Bound tree of a MRVRP instance, i.e. the initial model possibly enhanced with the constraints due to the branching decisions. As pointed out before, the specific variable selection strategy of LPcgLowerBound chooses both a delivery and a pick-up negative reduced cost route for any possible pair of starting and ending points, and can be refined. LPcgLowerBound can be used, for example, to solve the root node of the Branch&Bound tree. In this case, $\mathcal{M}$ is the initial $\mathcal{M}^{MRVRP}$ model, while $\overline{\mathscr{R}_0}$ is an initial set of dummy routes, which must be provided in order to avoid the infeasibility of the LP optimization (line 3). A reasonable choice is given by the set of all feasible one-customer routes:

$$\overline{\mathscr{R}_{DR}} = \bigcup_{\substack{u \in U \\ h \in U \cup L}} \Big(\bigcup_{i \in D} \{r \in \mathscr{R}^d : E_r = \{(u,i),(i,h)\}\} \cup \bigcup_{i \in P} \{r \in \mathscr{R}^p : E_r = \{(h,i),(i,u)\}\} \Big) \tag{A.23}$$

This gives rise to the CG-based heuristic algorithm cgHeur (figure A.1). Basically, it consists of feeding model $\mathcal{M}^{MRVRP}$ with the column set $\overline{\mathscr{R}_{DR}} \cup \overline{\mathscr{R}}$, $\overline{\mathscr{R}}$ being the columns returned by LPcgLowerBound($\mathcal{M}^{MRVRP}$, $\overline{\mathscr{R}_{DR}}$). Column Generation is performed at the root node only, and the MILP is solved via Branch&Bound, regardless of the implementation of any particular branching policy.

LPcgLowerBound($\mathcal{M}$, $\overline{\mathscr{R}_0}$)

argtype $\mathcal{M}$: MILPmodel; $\overline{\mathscr{R}_0}$: routeSet;
returns MRVRPfractSol; routeSet;
declare $\mathbf{x}$: MRVRPfractSol; $\overline{\mathscr{R}}$, $\overline{\mathscr{R}^\star}$: routeSet; r_h: route; u,h: node; $\boldsymbol{\pi}$: dualvarVector; n: int;

1 $\overline{\mathscr{R}} \leftarrow \overline{\mathscr{R}_0}$
2 **while(true)**
3 $\quad \mathbf{x} \leftarrow$ *LPoptimizeRMP*($\mathcal{M}$, $\overline{\mathscr{R}}$)
4 $\quad \boldsymbol{\pi} \leftarrow$ *retrieveDuals*()
5 $\quad n \leftarrow |\overline{\mathscr{R}}|$
6 $\quad$ **foreach**($u \in U$)
7 $\quad\quad G_u^d \leftarrow$ *deliverySubgraph*($\boldsymbol{\pi}$, u)
8 $\quad\quad \overline{\mathscr{R}^\star} \leftarrow$ *solveESPPRC*(G_u^d)
9 $\quad\quad$ **foreach**($h \in U \cup L$)
10 $\quad\quad\quad r_h \leftarrow \arg\min_{r \in \overline{\mathscr{R}^\star} : e_r^h = 1} \overline{c_r}^d$
11 $\quad\quad\quad$ **if**($\overline{c_{r_h}} < 0$) $\overline{\mathscr{R}} \leftarrow \overline{\mathscr{R}} \cup \{r_h\}$
12 $\quad\quad G_u^p \leftarrow$ *pickupSubgraph*($\boldsymbol{\pi}$, u)
13 $\quad\quad \overline{\mathscr{R}^\star} \leftarrow$ *solveESPPRC*(G_u^p)
14 $\quad\quad$ **foreach**($h \in U \cup L$)
15 $\quad\quad\quad r_h \leftarrow \arg\min_{r \in \overline{\mathscr{R}^\star} : e_r^h = 1} \overline{c_r}^p$
16 $\quad\quad\quad$ **if**($\overline{c_{r_h}} < 0$) $\overline{\mathscr{R}} \leftarrow \overline{\mathscr{R}} \cup \{r_h\}$
17 $\quad$ **if**($|\overline{\mathscr{R}}| = n$) **break**
18 **return** ($\mathbf{x}$, $\overline{\mathscr{R}}$)

cgHeur()

returns MRVRPintSol;
declare $\mathbf{x}$: MRVRPintSol; $\overline{\mathscr{R}}$: routeSet; $\mathbf{x}_f$: MRVRPfractSol;

1 ($\mathbf{x}_f$, $\overline{\mathscr{R}}$) $\leftarrow$ LPcgLowerBound($\mathcal{M}^{MRVRP}$, $\overline{\mathscr{R}_{DR}}$)
2 $\overline{\mathscr{R}} \leftarrow \overline{\mathscr{R}} \cup \overline{\mathscr{R}_{DR}}$
3 $\mathbf{x} \leftarrow$ *MILPoptimize*($\mathcal{M}^{MRVRP}$, $\overline{\mathscr{R}}$)
4 **return** $\mathbf{x}$

FIGURE A.1: A CG-based procedure to compute the optimal value of the LP relaxation of a node of the MRVRP Branch&Bound tree with a given initial set of column variables, and a heuristic algorithm based on it.

A.4.3 Towards a Branch&Price Algorithm. Branching Strategy

The algorithm cgHeur is clearly heuristic. Indeed, running a plain Branch&Bound on a subset of routes cannot yield an optimal solution, unless it is proved that such subset contains the routes of the optimal solution. That is the case, for example, of a family of approaches to the CVRP that can be found in the literature, like the one in [Baldacci et al., 2011a]. The authors apply a series of bounding procedures in order to raise the dual bound and reduce the optimality gap w.r.t. a previously determined feasible solution. Then, the exhaustive generation of the set $\mathscr{R}_g$ of all the routes whose

reduced cost is less than the optimality gap assures that $\mathscr{R}_g$ contains all the routes of any optimal solution. In general, the set $\overline{\mathscr{R}_{DR}} \cup \overline{\mathscr{R}}$ does not have this property; moreover, not only the provided solution is not optimal, but it can be arbitrarily bad. One can expect the nontrivial routes $\overline{\mathscr{R}}$ returned by LPcgLowerBound($\overline{\mathscr{R}_{DR}}$) be enough to generate at least a feasible solution, but unfortunately this is not always the case. This is the reason to start with an initial set $\overline{\mathscr{R}_{DR}}$ that does not contain artificial variables but proper routes.
In order to define an exact algorithm, it is necessary to:

1. run the CG procedure LPcgLowerBound at each Branch&Bound node, so as to compute the corresponding lower bound. $\mathcal{M}$ will be the current subproblem while $\overline{\mathscr{R}_0}$ will contain all the routes generated in the ancestor nodes;
2. adapt the branching policy to the CG framework, as the branching decision cannot concern route variables, as said in section 5.1.2.4. To this end, a branching policy based on *edge variables* (see again 5.1.2.4) is adopted.

An edge variable $x^\star_{ij}$ is defined as follows:

$$x^\star_{ij} = \sum_{r \in \mathscr{R}} b^{ij}_r \cdot x^\star_r$$

where coefficient b^{ij}_r is equal to 1 if route r passes through edge ij (cf section A.3.1). Such a variable is suitable to be branched on if it has a fractional value, i.e. if $0 < x^\star_{ij} < 1$. This gives rise to two branches, one with $x_{ij} = 0$, the other with $x_{ij} = 1$, and to as many child nodes of the current Branch&Bound tree node n. Suppose to call these latter n_0 and n_1, respectively. Branching decisions on edge variables can be propagated to the descending nodes of the Branch&Bound tree in an effective way as follows. For the sake of simplicity, we suppose that $ij \in E^d$, but a similar reasoning can be applied to pick-up:

1. the imposition of $x_{ij} = 0$ can be achieved by removing the edge ij from the graph of the delivery PPs. As a consequence, the PPs in the CG of the child node n_0 will not be able to generate routes r traversing edge ij, hence we will have $x^\star_{ij} = 0$ for n_0. The same will hold for its offspring. If for algorithmic efficiency reasons we keep the routes generated in n and its ancestor nodes in the variable set of the MP, we will also need to impose $x_r = 0$ for each of such routes s.t. $b^{ij}_r = 1$;
2. on the contrary, the imposition of $x_{ij} = 1$ can be accomplished by removing from the graph of the delivery PPs all the edges of the subset:

$$\{il, lj : l \in D \backslash \{i, j\}\} \subset E^d$$

The effect of this removal is that the PPs in the CG of n_1 and its descendants will generate routes r that either visit i and j in sequence (i.e. $b_r^{ij} = 1$), or neither of them (i.e. $a_r^i + a_r^j = 0$, a_r^i being 1 if r visits i, cf section A.3.1). At each CG iteration, the MP constraints (A.8) related to i and j can only be satisfied with routes having $b_r^{ij} = 1$. This is well shown in figure A.2. Therefore we will have:

$$\sum_{r \in \overline{\mathscr{R}}} a_r^i x_r = \sum_{r \in \overline{\mathscr{R}}} a_r^j x_r = 1 = \sum_{r \in \overline{\mathscr{R}}} b_r^{ij} \cdot x_r^\star = x_{ij}^\star$$

i.e. the solution to which CG converges, although generally fractional, cannot have $x_{ij}^\star < 1$. Seen from another perspective, as long as the RMP solution has $x_{ij}^\star < 1$ it is not feasible and the use of artificial variables is required to satisfy the contraints (A.8); further negative reduced costs exist hence, and edge ij will be associated with an important prize, which makes the subsequent generation of routes traversing ij likely. Again, if for algorithmic efficiency reasons we keep the routes generated in n and its ancestor nodes in the variable set of the MP, we will need to impose $x_r = 0$ for each of such routes that visit only one between i and j, i.e. such that $a_r^i + a_r^j = 1$.

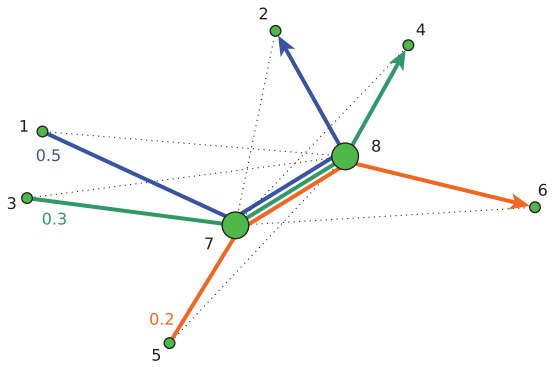


FIGURE A.2: A branching decisions of type $x_{ij} = 1$ ($x_{78} = 1$ in the example) on a node of the Branch&Bound tree is propagated to its offspring by removing from the graph of the PPs the other outgoing arcs of i and incoming arcs of j. Constraints (A.8) for both i and j can only be satisfied by routes traversing arc ij, therefore the CG will necessarily converge to a solution with $x_{ij}^\star = 1$, provided that all the columns associated with routes r s.t. $a_r^i + a_r^j = 1$ are removed from the MP.

A.5 Conclusions and Perspectives

This appendix has introduced the Multicommodity-Ring Vehicle Routing Problem, a tactical Vehicle Routing problem that descends from the MRLRP. Unlike the MRLRP, the MRVRP is more likely to be solvable by exact methods. Therefore we discussed the guidelines of a possible Branch&Price algorithm. Numerical experiments for the MRVRP are left as a perspective for future work.

Appendix B

Challenge ROADEF/EURO 2012: Machine Assignment Problem

B.1 Introduction

More and more information is available on the web. To help users finding the piece of information they need, several web search engines were created. The competition between them is tough as users want both accurate and quick results. A wide field of study concerns the design of algorithms able of promptly providing relevant results to users. The quickness of engines based on these methods relies on their design, but also on the availability of resources to assign them to. This need for huge amounts of resources – such as CPU or RAM – leads to the construction of computer clusters; but then, finding a good assignment of all the tasks to the different available machines is a key factor to improve the overall efficiency. Finding an optimal assignment in such a context is the subject of the challenge proposed by Google for the *2011-2012 ROADEF/EURO Challenge*. Its whole description can be found in [Société française de Recherche Opérationnelle et Aide à la Décision (ROADEF), 2011].

An instance of the proposed problem is made up by a set of processes, each one with specific resources consumption profiles. A set of machine is given with their resources capacities. An initial feasible assignment is provided. Processes can be moved from their initial assignment to a different machine leading to a new cost for the solution. However these moves must respect numerous hard constraints. Some constraints can be independently checked such as capacity constraints, but others link all the processes together in case of dependency between processes or restrictions due to potential conflicts. The new processes assignment must minimize a given cost function, which depends on machines

load, resources usage balance and process displacement costs.
In the following, we briefly describe the method used to solve the problem. Section B.2 presents the MILP program that was used to model the problem; Section B.3 describes the method used; Section B.4 gives the results obtained on the two sets of instances that were released by the organizers.

B.2 The model

In this section, we give a MILP program which models the problem. The notations that are used are the same as those introduced in [Société française de Recherche Opérationnelle et Aide à la Décision (ROADEF), 2011]. Otherwise they are introduced in the following subsections. The model is then given in two parts: first the objective function that is composed by five different costs; then the constraints. The constraints are of two kinds: those given by the problem and others logical constraints.

B.2.1 Parameters

- $n_m = |\mathcal{M}|$: number of machines
- n_s: number of processes per service $s \in \mathcal{S}$
- n_l: number of machines per location $l \in \mathcal{L}$

B.2.2 Variables

- x_{pm}: binary variable indicating if process $p \in \mathcal{P}$ is assigned to machine $m \in \mathcal{M}$
- y_{sl}: binary variable indicating if service $s \in \mathcal{S}$ is present at location $l \in \mathcal{L}$
- z_{rm}: integer variable measuring the excess of the safety capacity of resource $r \in \mathcal{R}$ at machine $m \in \mathcal{M}$
- t_{bm}: integer variable computing the result of the balance cost $b \in \mathcal{B}$ at machine $m \in \mathcal{M}$
- $smcVar$: integer variable measuring the service move cost

B.2.3 Objective function

$$
\begin{aligned}
totalCost &= \sum_{r\in\mathcal{R}} weight_{loadCost}(r)\cdot\Big(\sum_{m\in\mathcal{M}} z_{rm}\Big)\\
&+ \sum_{b\in\mathcal{B}} weight_{balanceCost}(b)\cdot\Big(\sum_{m\in\mathcal{M}} t_{bm}\Big)\\
&+ \sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{M}} x^0_{mp}\cdot(1-x_{mp})\cdot PMC(p)\cdot weight_{processMoveCost}\\
&+ smcVar\cdot weight_{serviceMoveCost}\\
&+ \sum_{p\in\mathcal{P}}\sum_{(m_1,m_2)\in\mathcal{M}^2, m_1\neq m_2} MMC(m_1,m_2)x^0_{m_1p}x_{m_2p}\cdot weight_{machineMoveCost}
\end{aligned}
$$

B.2.4 Constraints

$$
\begin{array}{rcll}
\sum_{p\in\mathcal{P}} x_{pm}R(p,r) & \leq & C(m,r) & \forall m\in\mathcal{M}, r\in\mathcal{R} \quad (i)\\
\sum_{p\in s} x_{pm} & \leq & 1 & \forall m\in\mathcal{M}, s\in\mathcal{S} \quad (ii)\\
\sum_{l\in\mathcal{L}} y_{sl} & \geq & spreadMin(s) & \forall s\in\mathcal{S} \quad (iii_a)\\
y_{sl} & \geq & \frac{\sum_{p\in s, m\in l} x_{pm}}{\min\{n_m,n_s,n_l\}} & \forall s\in\mathcal{S}, l\in\mathcal{L} \quad (iii_b)\\
y_{sl} & \leq & \sum_{p\in s, m\in l} x_{pm} & \forall s\in\mathcal{S}, l\in\mathcal{L} \quad (iii_c)\\
y_{sl} & \in & \{0,1\} & \forall s\in S, l\in L \quad (iii_d)\\
\sum_{p^a\in s^a}\sum_{m\in n} x_{p^am} - n_{s^a}\sum_{p^b\in s^b}\sum_{m\in n} x_{p^bm} & \leq & 0 & \forall n\in\mathcal{N}, p^a\in s^a\\
 & & & s^a \text{ dep. on } s^b \quad (iv)\\
\sum_{p\in\mathcal{P}}(x^0_{pm}+(1-x^0_{pm})\cdot x_{pm})\cdot R(p,r) & \leq & C(m,r) & \forall m\in\mathcal{M}, r\in\mathcal{R} \quad (v)
\end{array}
$$

Constraints (i) ensure the respect of the capacity constraints. Constraints (ii) assure that there is no two processes of the same service at the same machine. Constraints (iii_a) to (iii_d) assure the respect of the spread constraints. Constraints (iv) stand for the dependency constraints. Constraints (v) ensure the transient usage constraints. Additional logical constraints have to be added:

$$
\begin{array}{rcll}
z_{rm} & \geq & 0 & \forall r\in\mathcal{R}, m\in\mathcal{M}\\
z_{rm} & \geq & \sum_{p\in\mathcal{P}} x_{pm}\cdot R(p,r) - SC(m,r) & \forall r\in\mathcal{R}, m\in\mathcal{M}\\
t_{bm} & \geq & 0 & \forall b\in\mathcal{B}, \in\mathcal{M}\\
t_{bm} & \geq & target\cdot\Bigg(\Big(C(m,r_1)-C(m,r_2)\Big) & \\
 & & -\Big(\sum_{p\in\mathcal{P}} x_{pm}\cdot(R(p,r_1)-R(p,r_2))\Big)\Bigg) & \forall b\in\mathcal{B}, m\in\mathcal{M}\\
smcVar & \geq & 0 & \\
smcVar & \geq & \sum_{m\in\mathcal{M}, p\in s} x^0_{mp}\cdot(1-x_{mp}) & \forall s\in\mathcal{S}
\end{array}
$$

B.3 The method

The former model is an exact model. However it is not tractable to run it within the time limit imposed by the challenge, even on the small instances. To obtain good solutions, three different methods are used. The third method is designed to cope with the B instances, which size exceeds 5000 processes.

B.3.1 The local search

The local search is based on two simple moves: *shift* and *swap*. Both moves browse all processes and/or machines. For the B instances, only a randomly chosen subset of processes or machines are browsed because of their size. Note here that because of transient usage constraints, some processes cannot be assigned to some machines.
With the shift move, we try to move each process $p \in \mathcal{P}$ to another machine $m \in \mathcal{M}, m \neq M(p)$, where $M(p)$ is its current assignment. If this move improves the objective function cost and is valid, it is kept in memory. Once all the machines are tested, the best move for the process is kept in a stack of promising moves. The size of the stack is limited by a value Nb_{shift} keeping only the best processes to move sorted in a decreasing order of the gain in the cost function.
With the swap move, we try to swap a process $p \in \mathcal{P}$ with another process $p' \in \mathcal{P}$ such that $M(p') \neq M(p)$. If this move improves the objective function cost and is valid, then it is kept in memory. Here also once all the processes p' are tested, the best swap for p is kept in a stack of promising moves. The size of the stack is limited by a value Nb_{swap} keeping only the best processes to swap, sorted again in a decreasing order of the gain in the cost function.
In both cases, once all the promising moves are found, we build a solution with all these moves sequentially realized. We start from the first one of the stack. Then, following their position in the stack, the others are tried. For each move in the stack, if the new solution is still valid and improves the solution, the move is then performed.

B.3.2 The MILP driven search on increasing solution space

The solution obtained by the local search is then used to provide a warm start to the MILP model to further improve the cost of the new process assignment. However, since solving the exact model would be too time consuming, only a small model is built. To each machine is associated the sum of its balance cost and its load cost. Then only the $2m$ machines having the lowest and the greatest costs are selected. The processes that are assigned to them can be moved, while the others are frozen at their current

machine. If time enables, the solver is run several times increasing m and the size of the model. Between two calls to this method, the former local searches are performed for 10 iterations on the solution obtained.

B.3.3 The Super Process: aggregating processes

Although the former method enables to deal with the $A1$ and $A2$ instances, the B instances size does not allow to consider all the processes while calling the model. For that purpose, processes are aggregated to form *super processes*. These structures are then used in the MILP program rewritten to use super process instead of process as variables. This reduces the number of variables and so the new MILP program can be solved.
These super processes have for resource consumption the sum of the resource consumptions of the processes it represents, and gather several services. The combination of services enables to solve some of the dependency constraints. The creation of these super process changes from a call to another, in order to enable diversity. The number of processes in a super process is between 1 and 20, depending on the size of the instances. There are created by taking into account the repartition of the process services in the current solution. When a super process is moved from a machine to another, all the processes it represents are moved.

B.4 The results

Here we give the results. The machine used has a Pentium(R) Dual-Core E5500 2.80GHz with 3.8Go RAM on Debian version 64 version 7. The parameters Nb_{shift} is set to 50 and Nb_{swap} to the number of processes. The initial local search is run for 100 seconds, then the MILP driven search is called, possibly several times. When the remaining time is less than 60 seconds before the end, the local search is run starting from the best solution encountered.

instance	initial solution	final solution	deviation (%)	CPU time (s)
a1_1	49528750	44306501	89.4561	0.12
a1_2	1061649570	780581762	73.5254	489.32
a1_3	583662270	583006017	99.8876	421.25
a1_4	632499600	274364352	43.3778	516.01
a1_5	782189690	727578309	93.0181	510.24
a2_1	391189190	4168765	1.0657	525.90
a2_2	1876768120	895761559	47.7289	526.38
a2_3	2272487840	1399146389	61.5689	492.88
a2_4	3223516130	1773997601	55.0330	530.36
a2_5	787355300	529941864	67.3066	554.12
b1	7644173180	3554115209	46.4944	543.72
b2	5181493830	1019623424	19.6782	554.68
b3	6336834660	172170487	2.7170	512.84
b4	9209576380	4677870607	50.7935	467.04
b5	12426813010	931854930	7.4987	522.15
b6	12749861240	9525873556	74.7174	517.03
b7	37946901700	14990130791	39.5029	488.07
b8	14068207250	1215576946	8.6406	508.28
b9	23234641520	15885583344	68.3703	514.53
b10	42220868760	18099987199	42.8698	470.32

TABLE B.1: Results on the released instances.

Appendix C

Energy-aware Service Provisioning in Volunteers Clouds

C.1 Introduction

SlapOS [Smets-Solanes et al., 2011] is an open source Cloud Operating system which was inspired by recent research in Grid Computing [Cérin and Fedak, 2012] and in particular by BonjourGrid [Abbes et al., 2008, 2009], a meta Desktop Grid middleware for the coordination of multiple instances of Desktop Grid middleware. figure C.1 shows the current architecture. SlapOS is based on two types of nodes: SlapOS Nodes and SlapOS Master. SlapOS is an hybrid cloud in the sense that each of its nodes can be dedicated (Data center in figure C.1) or dynamically provisioned from volunteers (Home cloud in figure C.1). The Master's role is to install applications and run processes on SlapOS nodes. It acts as a central directory of all SlapOS Nodes, knowing where each SlapOS Node is located and which software can be installed on each node. The role of SlapOS Master is to allocate processes to SlapOS Nodes. In comparison to the traditional cloud view, SlapOS innovates in considering the possibility to build data centers-like with dedicated and volunteer nodes. The usage of volunteers node in the cloud data-center is particularly interesting for the storage of Big-Data or when dealing with massive computations. Successful projects like BOINC [Anderson, 2004] already demonstrated this last point; indeed, as we have volunteers, we can increase the computational power of the cloud. This solution is also interesting for improving the cloud elasticity at lower price. Indeed, it might be cheaper for the cloud owners, to negotiate the subscription of volunteer nodes, instead of buying new machines. Finally, this solution is also advantageous for energy saving since it promotes resources sharing. The SlapOS system aims at providing a platform as a service (PAAS). One of its core activity is to keep

available applications to its customers. In this paper, we are interested in optimizing the energy efficiency of this service provisioning system. Our optimization is based on the following scenario. Initially, we assume a finite set of requests; each referring to an application that must be accessed by a customer during a period of time. We want to deploy and maintain active the necessary applications such as to minimize the consumed energy while: (1) avoiding the overloading of physical machines and (2) easing the fault-tolerance by avoiding multiple deployments of a given application on a same machine. One challenge in this scenario is the potential unavailability of volunteer machines. Indeed, in comparison to dedicated machines, the clouds owners have few control on their availabilities. Therefore, the deployment must take into account potential migrations that can cause the unavailabilty of volunteers machines. We propose a complete solution (modeling, problem formulation and analysis, algorithms and experimental evaluations), for optimizing the described scenario. The organization of our contributions is as follows. In section C.2, we explain the main application of our work. In section C.3 we define the computational problems related to the minimization of energy consumption in our service provisioning scenario. We also provide a theoretical proof of the NP-hardness. In section C.4, we provide an Integer Programming model for service provisioning in volunteer clouds. In section C.5, we propose two heuristics for the resolution of our service provisioning problem. The first heuristic proposes to subdivide the problem in short periods on which we apply the integer linear programming model. The second heuristic is also based on the subdivision of the general problem. However, subproblems here are solved with a greedy algorithm. In section C.6 we conduct extensive experiments under different configurations. The first series of experiments are about a comparaison between the ILP solutions and the heuristics issued from the greedy schema. The second series of experiments analyse our heuristics for finding the best implementation. Section C.7 is about related works. Section C.8 concludes the paper. We would like to assert that the paper is about *modeling* first and not about experiments on a real system. However, we estimate that our modeling is a good compromise between the parameters we consider impacting the phenomenon and the realism of current cloud technologies. For instance, our parameter settings consider the SlapOS cloud system running in production at Paris 13[1].

C.2 Application of our work

SlapOS is a PAAS whose data-center combines various types of execution supports (desktop, smartphones, tablets etc.). This diversity is very interesting in the perspective of

[1] `https://slapos.cloud.univ-paris13.fr`

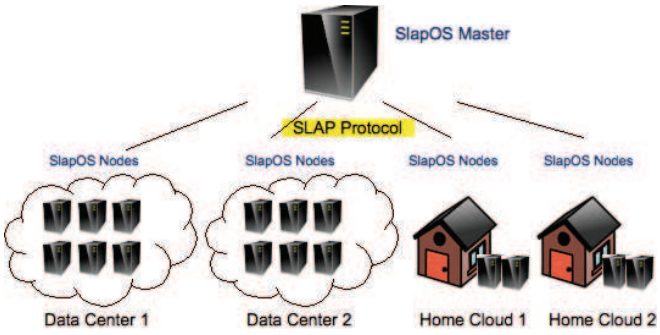


FIGURE C.1: The SlapOS Architecture.

increasing the cloud computational power from volunteer subscriptions(what is important for managing big data). In practice however, there are many challenges to address in the management of such a node diversity. For example, the SlapOS team ported the system on a Nexus S mobile phone and under Android 4.0.4. A Wordpress (which contains an Apache server) was also deployed on the Nexus and the operating system was a 'chrooted' Debian. The standby processor option (Wave Lock) was disabled and Wifi was enabled (Wifi Lock). A script was coded for writing the date in a file every 5 minutes. They measured the battery life when Wordpress was running and when it was not running. The script was running in these two cases.

On this simple deployment, the SlapOS team noticed a battery life of 22 hours without Wordpress and 12 hours with Wordpress running. They also noticed that starting a new Wordpress instance took few seconds with Android. The main conclusions of these experiments were the followings: a) SlapOS applications may reduce drastically the battery life. This high penalty might not motivate volunteers to keep their nodes in SlapOS clouds b) the penalty in restarting an application when needed is not prohibitive. In another words, we could imagine at large scale to use volunteers nodes (Tablets, Smartphones, PCs) for hosting Web applications under the condition that we know how to optimize the energy consumption.

We consider this paper as a first stage for improving the SlapOS provisioning system in this direction. In its actual version, SlapOS implements the following behavior. For joining a SlapOS cloud, any new volunteer must first register its machine to a SlapOS master node by declaring various parameters such as the memory space, disk space, CPU type, location of the machine... Then, the volunteer can choose from the catalog of SlapOS applications some that he accepts to install on his machine. Finally, the volunteer can let his machine opened for other users (the machine is registered as public) or not. For any user, SlapOS maintains a list of executable applications. These applications are those that the user accepted to install or those that are located on machines nodes registered as public.

In the optimization scenario that we consider, we assume that, the SlapOS provisioning system has a list of requests for applications. Each request refers to an application that the system must maintain available to a client during a period of time. An application can be referenced in multiple requests. For servicing the requests, the provisioning system can use either dedicated or volunteer nodes (those that support the requested applications). The objective is to build a plan for deploying and run instances of applications such as to minimize the energy consumption.

Since our research focuses on a green scheduler for large scale systems such as Clouds, the volume of information exchanged between nodes is potentially big. Frequent decisions will generate "Big-Data" volume oriented problems. Infrequent decisions may generate inaccuracies. This paper is about foundations to deal with scheduling generating large amount of bytes (for instance during the migration steps), software and tools to manage the complexity of building green data centres under environmental perspectives and sustainable development. The next section deals with the formal description of this problem.

C.3 Problem Description

We propose in this part two modeling approaches for optimizing the provisioning of requests in volunteer clouds. As we will see, such provisioning gives rise to two computational problems which are formally different.

Let us introduce first some general context notations and assumptions which are common to both modeling approaches and both problems. We assume that the plan to build must deploy a set $N = \{1, ..., n\}$ of applications on a set $M = \{1, ..., m\}$ of both dedicated and volunteer machines. For each application j, k_j instances are required. Each machine i has a capacity q_i, i.e. the maximal number of application instances that can be deployed on it at the same time. The plan must ensure that the applications are available to requesters during a time horizon $[0, T]$. At each instant t in the time horizon, the run of any application $j \in N$ on a machine $i \in M$ will cause a power consumption equal to $P_{ji}(t)$. As a consequence, the energy consumption for deploying and running the application j on the machine i all through the time horizon will be $\int_0^T P_{ji}(t)dt$. For the sake of simplicity, we will assume that variations in the power consumption during the runtimes of any application are negligible, i.e. $P_{ji}(t) \simeq P_{ji}$, with P_{ji} a constant value.

Since some machines are volunteers, there may be short intervals of time $[t_a, t_b] \subset [0, T]$ in which some of them are unavailable. In order to properly model this, we assume that the time horizon is discretized in $\frac{T}{\Delta}$ *periods* of size Δ, which will referred to by

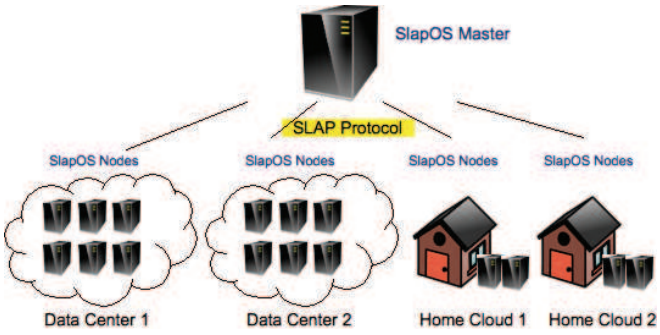


FIGURE C.2: The SlapOS Architecture.

means of index $\tau \in \mathcal{T} = \{1, ..., \frac{T}{\Delta}\}$. We also define index set $\mathcal{T}_0 = \mathcal{T} \cup \{0\}$. Moreover, in each period, a deterministic information related to the availability of any machine is given, in the form of a matrix $\mathbf{B}$. More precisely, element B_i^τ states whether or not i is available during the interval $\tau \equiv [(\tau - 1)\cdot\Delta, \tau\cdot\Delta[$. Let us observe that $\mathbf{B}$ can be taken as a bit matrix. This notion was considered in past work in volunteer computing for describing the machine availability [Iglesias et al., 2012]. Practically, we can obtain a good approximation of this matrix in negotiating the availabilities with the volunteers of the system, during the machine registration in the cloud. For the sake of conciseness, in the following we will often make use of vector notation as we have implicitly done for $\mathbf{B}$:

$$\mathbf{B} = (B_i^\tau)|_{i\in M}^{\tau\in\mathcal{T}}$$

C.3.1 First modeling approach

In the first proposed modeling approach, the energy consumption is approximated by the sum of two terms. The first is the *energy consumed in periods*. During period τ, given an application j and a machine i, this energy can be computed as follows:

$$B_i^\tau \int_{(\tau-1)\cdot\Delta}^{\tau\cdot\Delta} P_{ji}(t)dt = B_i^\tau \cdot P_{ji} \cdot \Delta = B_i^\tau \cdot E_{ji}^c$$

where $E_{ji}^c = P_{ji}\cdot\Delta$ is the energy consumption of an application j when running for a period on a machine i. The second term is the *energy consumed in transfers*: at any time, given an application j to transfer from h to i, this energy is equal to C_{jhi}.

The aim is to determine a plan that ensures at each period τ that the requested number of instances for each application are deployed; the plan must also minimize the total

energy consumed in *periods* and *transfers*; finally, it must respect the three constraints of Table C.1.

C_1	*Applications must be deployed on available machines*;
C_2	*The k_j instances of application $j \in N$ must be assigned to k_j distinct machines*;
C_3	*On any machine $i \in M$, we cannot have more than q_i applications.*

TABLE C.1: Constraints of the outlined problems

Constraints C_1 are straightforward. Relations C_3 ensure that the machine will not be overloaded; in other work, such constraints are often defined under the notion of power capping [Georgiou et al., 2014]. Finally, constraints C_2 ensure that the applications on a same machine are distinct. This is important for fault tolerance reasons, since given an application j, the greater is the number of machines its k_j instances are deployed to, the higher will be the fault tolerance degree of application j.

Symbol	Description
$[0, T]$, Δ	Time horizon and duration of periods
$\frac{T}{\Delta}$, $\tau \in \mathcal{T} = \{1, ..., \frac{T}{\Delta}\}$	Number, index and set of periods
n, $j \in N = \{1..n\}$	Number, index and set of applications
$\mathbf{k} = (k_j)\|_{j \in N}$	Vector of number of required instances of each application
m, $i \in M = \{1..m\}$	Number, index and set of machines
$\mathbf{q} = (q_i)\|_{i \in M}$	Machine capacity vector
$\mathbf{B} = (B_i^\tau)\|_{i \in M}^{\tau \in \mathcal{T}}$	Machine availability matrix
$\mathbf{E^c} = (E_{ji}^c)\|_{i \in M}^{j \in N}$	Energy consumption matrix
$\mathbf{C} = (C_{jhi})\|_{i,h \in M}^{j \in N}$	Transfer Energy matrix

TABLE C.2: Summary of input data of the outlined problems

The aim defined so far –the minimization of the overall energy consumption– can be further defined according to whether we adopt the provisioning system viewpoint or the volunteer viewpoint. In the former case, the consumed energy is the sum of energy consumed in periods and transfers for all machines. We will refer to this problem as the *MinSum-Plan Provisioning Problem on Volunteer Clouds (PPPVC$_{MinSum}$)*. In the latter case, the energy consumed by the whole system is not necessarily important. What is critical is the one consumed by the volunteer's machine. The objective function consists then in minimizing the maximal sum of consumed energy (in transfers and period) that involves any machine. We will refer to this problem as the *MinMax-Plan Provisioning Problem on Volunteer Clouds (PPPVC$_{MinMax}$)*.

We provide an example that illustrates how we compute the objective function in each one of the two problems.

We assume a case where $k_j = 1$ copy for each one of the n applications, and we consider $m = 4$ machines, which capacities are respectively $q_1 = q_2 = n$, $q_3 = q_4 = \frac{n}{2}$; the availability matrix B is defined in Table C.3, which shows that machines 3 and 4 are always available, whereas machines 1 and 2 are alternatively available.

$(\tau \mod 2)$	B_1^τ	B_2^τ	B_3^τ	B_4^τ
0	1	0	1	1
1	0	1	1	1

TABLE C.3: Availability matrix in the example

We assume that for each $j \in N$, the energy consumed in periods are $E^c_{j1} = E^c_{j2} = \frac{E}{n}$ and $E^c_{j3} = E^c_{j4} = \frac{2E}{n}$, whereas the energy consumed in migrations C_{jhi} are supposed to be equal to C regardless of j, i and h, E and C being constant terms. A possible plan solution in this setting consists of alternatively putting the instances on the machines 1 and 2 according to their availability. In this case, the objective value for PPPVC$_{MinSum}$ is equal to $|\mathcal{T}|\,E + (|\mathcal{T}|-1)\,n\,C$. For PPPVC$_{MinMax}$, the objective value for this solution will be $\lceil\frac{|\mathcal{T}|}{2}\rceil \cdot E + (|\mathcal{T}|-1)\,n\,C$. In this case, for the computation of the energy consumed in periods, we take into account the fact that the machines 1 and 2 are available at most in $\lceil\frac{|\mathcal{T}|}{2}\rceil$ periods. Between any period, both machines are involved in the transfer; hence the total transfer costs. Let us observe that the solution that we considered is not necessarily optimal for our problems. For PPPVC$_{MinSum}$, if $(|\mathcal{T}|-1)\,n\,C > |\mathcal{T}|\,E$, the best solution consists of using only the machines $m = 3$ and $m = 4$, which will lead to a total energy consumption equal to $2\,|\mathcal{T}|\,E$.

This example shows that the two problems differ. It also shows how both the energy in periods and the energy in transfers, and the ratio between the two can play a role in determining the optimal solution.

C.3.2 Second modeling approach

On many points, the first modeling approach is objectionable. Indeed, we focus only on two types of inputs for energy consumption: the one resulting from applications run in a period and the one resulting from transfers. In practice, it might be important to include other separate costs like the energy induced by the action of deploying an application on a node. The definition of the total energy consumed by a set of applications in a period is also criticizable in the first modeling. We will overcome such issues with the second modeling approach.

According to the assumptions we have made so far, given $j, j' \in N$ and $i \in M$, the power consumption for running j on i is equal to constant P_{ji} at each $t \in [0, T]$, therefore the execution of both j and j' on i causes a power consumption $P_{ji} + P_{j'i}$. However,

this formula does not take into account the *shared* or *mutual* power consumption due to the fact that the applications are run simultaneously on the same machine. More precisely, on each machine we must dissociate the base power consumption due to system operations from that induced by the running applications.

In order to take into account this further level of detail in the description of energy consumption, we introduce for each machine i and application j, two quantities: a *base energy* E_i^b due to the processing of systems operations, and an *overhead energy* E_{ji}^+ induced by the run of applications that we deployed on i. As a consequence, given a machine i and the subset $N^\tau \subset N$, $|N^\tau| \leq q_i$ of applications that have been deployed on it at period τ, the energy consumed on i in the same period is:

$$E_i^b + \sum_{j \in N^\tau} E_{ji}^+$$

The energy input dataset is now composed of $\mathbf{E} = (\mathbf{E^b}, \mathbf{E^+}, \mathbf{C})$. Therefore, one obtains a generalization of the first modeling approach, which is the specific case with $\mathbf{E^b} = \mathbf{0}$. We can consequently reformulate both PPPVC$_{MinSum}$ and PPPVC$_{MinMax}$ according to this new modeling approach.

Below, we will analyze the hardness of the computational problems.

C.3.3 Problem analysis

Since the first modeling approach turns out to be a particular case of the second one, we refer in the following analysis to the former, as every finding on its hardness can be extended to the latter.

Theorem C.1. *PPPVC$_{MinSum}$ is NP-hard in the first modeling approach.*

Proof. Proof In a previous work [Cérin et al., 2013], we have demonstrated the NP-hardness of PPPVC$_{MinSum}$ by showing that a particular case of it can be reduced in polynomial time to the axial three-index minimization assignment problem (A3IAP-Min), which is known to be NP-hard (see for instance [Karp, 1972], [Ausiello et al., 1999]). □

Corollary C.2. *Unless $P = NP$, PPPVC$_{MinSum}$ is inapproximable in the first modeling.*

This is because no polynomial time algorithm can achieve a constant performance ratio for A3IAP-Min unless $P = NP$ (see [Ausiello et al., 1999]).

Theorem C.3. *In the first modeling approach, $PPPVC_{MinMax}$ is strongly NP-hard*

Proof. Proof We obtain the proof by a reduction to a special case of the 3-Partition problem; that is NP-hard in the strong sense. An instance of this problem is given by a set S of $3k$ elements such that for each $u \in S$, we have an associated positive integer $s(u)$. We also have $\sum_{u \in S} s(u) = k \cdot T$ and for each u, we have $T/4 < s(u) < T/3$. The problem consists in finding k disjoints subsets $S_1, \ldots, S_k$ of 3 elements each such that the sum of the values of elements in each S_i is equal to T and $\bigcup_{i=1}^{k} S_i = S$.

We can solve this instance from the following PPPVC$_{MinMax}$ instance. We assume $3k$ applications to deploy on k machines in one period of time. The capacity of each machine i is $q_i = 3$ and all machines are available. For any application j, we have $k_j = 1$. We associate each application j with a distinct positive integer $u \in S$. Let us capture such associations with the formula $f(u) = j$. Finally, we set the energy consumed per period to $E_{f(u)i} = s(u), \forall i$. We then solve this PPPVC$_{MinMax}$ instance. Let Z be the makespan returned by this solution (the maximal energy consumption per machine). If $Z = T$ then we state that the partition has a solution $S_1, \ldots S_k$ where

$$S_i = \{u : \ f(u) = j \text{ and } u \text{ is deployed on the machine } i\} \tag{C.1}$$

Otherwise, we state that there is no solution to the partition problem.

It is obvious to notice that given the relation $\sum_{u \in S} s(u) = kT$, the smallest possible makespan of the associated PPPVC$_{MinMax}$ instance is T; and in this case, the energy consumed per machine is exactly T. More, if there is a solution to the PPPVC$_{MinMax}$ instance, then it will set exactly three distinct applications per machine. This is because we have a total of $3k$ applications and $q_i = 3$. Consequently, the relation(C.1) of the S_is ensures that any solution to the PPPVC$_{MinMax}$ for which the makespan is T is a solution to the partition problem. In the same way, from any solution to the partition problem, we can easily build a solution to the related PPPVC$_{MinMax}$ instance. It suffices to set on each machine i the applications $f(u)$. Since it is straightforward that the reduction can be done in polynomial time, we have the proof. □

Since the second modeling is a generalization of the first one, these NP-hardness results can be extended to it. It might however be interesting to see if the generalizations introduced by the second modeling make the problem *harder*. For this, let us consider the specific cases where the overhead induced by applications are all null ($\forall i, j \ E^{+}_{ji} = 0$). We have the following result.

Theorem C.4. *In the second modeling, $PPPVC_{MinSum}$ is NP-hard even with a null overhead for applications.*

Proof. Proof In [Cérin et al., 2013], it is shown that a specific case of this problem can be reduced to the minimum knapsack problem. □

Consequently, the second modeling approach introduces computational challenges in the resolution. Thus, we will focus on the resolution of PPPVC$_{MinSum}$ formulated in compliance with the second modeling approach.

C.4 An Integer Linear Programming model for PPPVC$_{MinSum}$

The first method to tackle PPPVC$_{MinSum}$ is by means of Integer Linear Programming (ILP), which gives rise to an *exact method*. Exact methods guarantee to find an optimal solution, in our case a minimum energy deployment, since the whole set of feasible plans is taken into account. However, exact methods can be computationally heavy when dealing with medium- to big-sized problem instances. For such instances, we will design and make use of *heuristic algorithms*.

Our ILP model is based on the following decision variables:

- binary *allocation* variables x_{ji}^{τ}, $\tau \in \mathcal{T}$, $j \in N$, $i \in M$, where x_{ji}^{τ} is equal to 1 if and only if one copy of application j is allocated on machine i at period τ;
- binary *usage* variables y_i^{τ}, $\tau \in \mathcal{T}$, $i \in M$, which state whether or not machine i is used during period τ;
- binary *migration* variables v_{jhi}^{τ}, $\tau \in \mathcal{T} \setminus \{1\}$, $j \in N$, $h, i \in M$, with v_{jhi}^{τ} equal to 1 if a copy of application j is deployed on machine h in period $\tau - 1$ and on machine i in the following period τ. In other words, $v_{jhi}^{\tau} = 1$ implies a migration cost C_{jhi} at period τ. Movement variables are obviously not defined for $\tau = 1$.

Note that since at most one instance of each application can run on one machine (for the aforementioned fault tolerance reasons), binary allocation variables x_{ji}^{τ} are sufficient to describe allocation decisions.

We can therefore express our objective function as:

$$z_{\text{MinSum}}^{\text{PPPVC}} = \sum_{\substack{i \in M \\ \tau \in \mathcal{T}}} E_i^b \, y_i^{\tau} + \sum_{\substack{j \in N \\ i \in M \\ \tau \in \mathcal{T}}} E_{ji}^{+} \, x_{ji}^{\tau} + \sum_{\substack{j \in N \\ h,i \in M \\ \tau \in \mathcal{T}}} C_{jhi} \, v_{jhi}^{\tau}. \tag{C.2}$$

$z_{\text{MinSum}}^{\text{PPPVC}}$ accounts for three energy consumption terms: base energy, overhead energy and transfers' energy, in the order they appear in (C.2). This latter could be rewritten as

follows:

$$z^{\text{PPPVC}}_{\text{MinSum}} = \sum_{\substack{i\in M\\ \tau\in\mathcal{T}}} \left(E^b_i\, y^\tau_i + \sum_{j\in N} \left(E^+_{ji}\, x^\tau_{ji} + \sum_{h\in M} C_{jhi}\, v^\tau_{jhi} \right) \right) \tag{C.3}$$

so as to remark that energy consumption can be subdivided according to machines and periods, and that given a period $\tau \in \mathcal{T}$ and a machine $i \in M$, the energy consumption can again be divided into base, overhead and transfers' energy.
As a consequence, the ILP model is:

$(\mathcal{M}^{\text{PPPVC}}_{\text{MinSum}})$

$$\begin{aligned}
&\min && z^{\text{PPPVC}}_{\text{MinSum}} && && \text{(C.4)}\\
&\text{s.t.} && y^\tau_i \le B^\tau_i && \forall\, i\in M,\ \tau\in\mathcal{T} && \text{(C.5)}\\
& && x^\tau_{ji} \le y^\tau_i && \forall\, j\in N,\ i\in M,\ \tau\in\mathcal{T} && \text{(C.6)}\\
& && \textstyle\sum_{j\in N} x^\tau_{ji} \le q_i\, y^\tau_i && \forall\, i\in M,\ \tau\in\mathcal{T} && \text{(C.7)}\\
& && \textstyle\sum_{i\in M} x^\tau_{ji} = k_j && \forall\, j\in N,\ \tau\in\mathcal{T} && \text{(C.8)}\\
& && \textstyle\sum_{h\in M} v^\tau_{jhi} \le x^\tau_{ji} && \forall\, j\in N,\ i\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.9)}\\
& && \textstyle\sum_{i\in M} v^\tau_{jhi} \le x^{\tau-1}_{jh} && \forall\, j\in N,\ h\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.10)}\\
& && \textstyle\sum_{i\in M} v^\tau_{jhi} \le 1 - x^\tau_{jh} && \forall\, j\in N,\ h\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.11)}\\
& && \textstyle\sum_{h\in M} v^\tau_{jhi} \le 1 - x^{\tau-1}_{ji} && \forall\, j\in N,\ i\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.12)}\\
& && \textstyle\sum_{i\in M} v^\tau_{jhi} \ge x^{\tau-1}_{jh} - x^\tau_{jh} && \forall\, j\in N,\ h\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.13)}\\
& && y^\tau_i \in \{0,1\} && \forall\, i\in M,\ \tau\in\mathcal{T} && \text{(C.14)}\\
& && x^\tau_{ji} \in \{0,1\} && \forall\, j\in N,\ i\in M,\ \tau\in\mathcal{T} && \text{(C.15)}\\
& && v^\tau_{jhi} \in \{0,1\} && \forall\, j\in N,\ i,h\in M,\ \tau\in\mathcal{T}\setminus\{1\} && \text{(C.16)}
\end{aligned}$$

Relation (C.4) asserts that we seek for the minimal value of $z^{\text{PPPVC}}_{\text{MinSum}}$ in the space of all feasible assignments of instances to machines all along the time horizon. Relations (C.5)–(C.13) define, in the form of constraints on the decision variables, such space of feasible (i.e., that meet the criteria) assignments. Finally, (C.14)-(C.16) state the binary nature of all of the variables.
Relations (C.5) impose that a machine cannot be used in periods of unavailability; relations (C.6) assert that if machine i is not used, then no application instance can be deployed on it, as $x^\tau_{ji} \le y^\tau_i$ forbids to have $y^\tau_i = 0$ and $x^\tau_{ji} = 1$. Condition (C.7) imposes that in case machine i is available and used at τ, then at most q_i instances can be allocated on it: infact $y^\tau_i = 1 \Rightarrow \sum_{j\in N} x^\tau_{ji} \le q_i$. Constraints (C.8) guarantee the requested number k_j of copies of each application j at each τ, on no matter which

machines.

Constraints (C.9)–(C.13), along with the expression of $z^{\text{PPPVC}}_{\text{MinSum}}$, make v variables actually account for instances transfers. In fact, given machines h and i and a period τ, if $x^{\tau-1}_{jh} = 1$ and $x^{\tau}_{jh} = 0$, then one among variables v^{τ}_{jhi}, $i \in M \setminus \{h\}$ must be set to 1: this is stated by constraints (C.13), and the ILP solver will choose the proper one according to the energy cost drivers of objective function $z^{\text{PPPVC}}_{\text{MinSum}}$. Relations (C.9) and (C.10) avoid any variable v^{τ}_{jhi} to be improperly set to one if application j has neither been deployed on machine h at period $\tau - 1$, nor on machine i at period τ. In fact, for instance for (C.9), we have:

$$x^{\tau}_{ji} = 0 \Rightarrow \textstyle\sum_{h \in M \setminus \{i\}} v^{\tau}_{jhi} = 0 \Rightarrow (\forall h \in M \setminus \{i\})\ v^{\tau}_{jhi} = 0 \qquad \text{whereas}$$
$$x^{\tau}_{ji} = 1 \Rightarrow \textstyle\sum_{h \in M \setminus \{i\}} v^{\tau}_{jhi} \leq 1$$

i.e. in the second case at most one of the v^{τ}_{jhi} can be set to 1. Relations (C.10) behave in a similar way. Constraints (C.11) and (C.12) are more subtle and used to avoid improper migration accounting when an application copy *remains* on one same machine on two consecutive time periods. Roughly speaking, if $x^{\tau-1}_{ji} = x^{\tau}_{ji} = 1$, i.e. a copy of j is deployed on machine i at period $\tau - 1$ and does not migrate at period τ, then each migration variable v^{τ}_{jih} (from i towards any other h) and v^{τ}_{jhi} (from any h towards i) is prevented to take value 1.

Note that all along the model application migrations are taken into account and paid for starting from period $\tau = 2$.

One last observation concerning the model: constraints (C.5)–(C.8), along with all concerned x^{τ}_{ji} and y^{τ}_{i} variables, can be removed in a preprocessing phase for all i, τ such that $B^{\tau}_{i} = 0$, since $\mathbf{B}$ is entirely known a priori; the same holds for constraints (C.9)–(C.13) and variables v^{τ}_{jhi} for all i, h and $\tau > 1$ such that $B^{\tau-1}_{h} = 0$ or $B^{\tau}_{i} = 0$. Here we have shown the whole of them again for explanation purposes only.

As said, the described mathematical model $\mathcal{M}^{\text{PPPVC}}_{\text{MinSum}}$ gives rise to an ILP-based algorithm, which in its most generic form is called a *Branch&Bound algorithm*. We briefly recall how does such an algorithm work, in order to focus on some concepts that have driven the computational experience described later in section C.6.

The search in the solution space of a ILP minimization problem can be computationally heavy due to the integrality constraint, but so it is not for its *LP relaxation*, i.e. the mathematical problem we obtain by relaxing such contraints, because the solution space of the latter can be efficiently explored by means of the well-known *simplex algorithm*. However, the value of the optimal solution of the relaxed problem is a *lower bound (LB)* on that of the optimal solution of the original problem, since the solution space of the former is enlarged with respect to that of the latter due to the integrality constraint

relaxation. This is a key concept: even if a ILP minimization problem is difficult to solve, it is normally easy to get a lower estimate on the value of its optimal solution.
The Branch&Bound is a strategy to implicitly enumerate all of the solutions of an ILP minimization problem which makes use of this concept. During this search, each new improving integer solution is saved as the best found one and represents an *upper bound(UB)* on the value of the optimal solution. As the search goes on, UB and LB are progressively refined: the value of the best found integer solution decreases, while the value of the lower estimate LB grows. The search is stopped when the so-called *gap* $g = \frac{UB-LB}{UB}$, i.e. the relative distance between UB and LB, converges to a value lower than a given threshold $\overline{g}$. The higher is the value of $\overline{g}$, the less will be both the accuracy of the solution and the time to stop the search, and viceversa.
However, such time depends also on the *quality* of the lower bound LB: if such lower bound is *loose*, i.e. the difference between the optimal solution of the integer problem and that of its LP relaxation is considerable, then so will be the initial gap and the time to reach the $g < \overline{g}$ condition; on the other hand, a *tight* lower bound will ease the convergence, as the gap will be small from the very beginning of the search.

C.5 Heuristic Approaches to PPPVC$_{MinSum}$

Structurally speaking, PPPVC$_{MinSum}$ consists in a sequence of two-dimensional assignment problems which span a time horizon and are interconnected between them; since two-dimensional assignment problems are known to be polynomially –thus easily– solvable, the aspect that make the degree of complexity of PPPVC$_{MinSum}$ rise is time dependence and the relations between decisions concerning different time periods. The proof comes from some preliminary tests: in spite of the relatively reduced complexity of the time dependencies (they only link couples of consecutive deployments), as the number $|\mathcal{T}|$ of periods grows –the other problem dimensions staying the same– the problem becomes from computationally light, to difficult, to nearly intractable.
An approach to deal with the hardness of PPPVC$_{MinSum}$ instances with a wide time horizon is to subdivide the overall problem in more tractable subproblems and then obtain a global solution to the former by assembling the solutions of the latter ones. In the following, we present two heuristic algorithms: a more classic *rolling horizon algorithm*, and a second one based on *Consecutive Execution Blocks*. In spite of approaching the problem in two somehow complementary fashions, both follow the aforementioned solution strategy.

C.5.1 A Rolling Horizon Algorithm

Rolling Horizon techniques are commonly used when dealing with either uncertainty over data or long-term planification/scheduling problems. PPPVC$_{MinSum}$ is easily seen to belong to the latter class of problems, which are inherently time-dependent and characterized by strong interdependencies between different time steps, thus extremely difficult to solve to optimality when dealing with a long time horizon. The literature of Combinatorial Optimization offers many examples of rolling horizon frameworks (see e.g. [Rakke et al., 2011] and [Jaillet et al., 2002]).
A rolling horizon approach to tackle such problems typically consists in defining a sequence of subproblems by shifting a *rolling time window* all along the long-term time horizon which is therefore *covered* by the time horizons of the subproblems. The latter are then easily solved (due to their shorter time horizon) and their solutions interlaced, in order to obtain a global solution to the overall planning problem.
Obviously, not only this is a *heuristic* approach, but it calls for:

- a good compromise in the subdivision in subproblems: they must be small enough to be easily solvable, yet the original problem must not be split in too many subproblem, to avoid that the growth of the computation time overcompensates the benefit of the computational ease, and the solution reconstruction process becomes too time-consuming;
- a way to efficiently take into account the dependencies between the subproblems. This is the most delicate aspect, as it requires to find a way to embed in each subproblem J_h the knowledge of the deployment decisions taken in the previously solved subproblems $J_0, ..., J_{h-1}$, and that of the machine availability in the time periods covered by the following ones.

To tackle the first aspect, a common solution is to have each subproblem span an identical number of time periods, which is experimentally tuned. As for the second aspect, a possible solution is to:

- make the time horizon of consecutive subproblems *overlap*;
- incorporate in the objective function of a subproblem a term derived from the knowledge of the availability of the machines in the forthcoming time periods.

We refer to figure C.3 for a small example: the overall problem has a time horizon of 8 periods ($\mathcal{T} = \{1, ..., 8\}$), whereas that of subproblems has a width of 3 periods (with the exception of the last one): subproblem 1 spans over periods $\tau = 1, 2, 3$ (red box),

subproblem 2 over periods $\tau = 3, 4, 5$ (blue box), and so on. Subproblem 1 can decide the deployment at $\tau = 3$ while considering the migration costs that such decisions will entail, as it is aware of the machine availability at $\tau = 4$ (and in general for each $\tau \in \mathcal{T}$). In a complementary way, subproblem 2 can use the deployment at $\tau = 2$ provided by subproblem 1 as initial condition.

In the following, we introduce and discuss in greater depth the proposed rolling horizon algorithm. In order to do so, let us first introduce some extensions to model $\mathcal{M}^{\text{PPPVC}}_{\text{MinSum}}$ and additional notations.

C.5.1.1 Initial Conditions

First, we extend the definition of migration variables v^{τ}_{jhi} to time period $\tau = 1$:

$$v^{\tau}_{jhi} \in \{0, 1\} \qquad \forall\ j \in N,\ i, h \in M,\ \tau \in \mathcal{T} \tag{C.17}$$

Then we extend to period $\tau = 1$ constraints (C.9)–(C.13) as follows:

$$\sum_{h \in M} v^{1}_{jhi} \leq x^{1}_{ji} \qquad \forall\ j \in N,\ i \in M \tag{C.18}$$

$$\sum_{i \in M} v^{1}_{jhi} \leq X_{jh} \qquad \forall\ j \in N,\ h \in M \tag{C.19}$$

$$\sum_{i \in M} v^{1}_{jhi} \leq 1 - x^{1}_{jh} \qquad \forall\ j \in N,\ h \in M \tag{C.20}$$

$$\sum_{h \in M} v^{1}_{jhi} \leq 1 - X_{ji} \qquad \forall\ j \in N,\ i \in M \tag{C.21}$$

$$\sum_{i \in M} v^{1}_{jhi} \geq X_{jh} - x^{1}_{jh} \qquad \forall\ j \in N,\ h \in M \tag{C.22}$$

We will denote as $\underline{\mathcal{M}}^{\text{PPPVC}}_{\text{MinSum}}$ the model we get from $\mathcal{M}^{\text{PPPVC}}_{\text{MinSum}}$ by replacing (C.16) by (C.17) and adding (C.18)–(C.22). Term $\mathbf{X} = (X_{ji})|^{j \in N}_{i \in M}$ is an *input* of the problem and represents its *initial conditions*: a preliminar deployment of the application on the machines, prior to the beginning of the time horizon, that determines the transfer costs at period $\tau = 1$. Note that by imposing $(\forall j \in N, i \in M)\ X_{ji} = 0$ we state the absence of any particular initial conditions, since this makes transfer costs at period $\tau = 1$ be equal to 0 (see (C.19)). Hence, model $\underline{\mathcal{M}}^{\text{PPPVC}}_{\text{MinSum}}$ generalizes $\mathcal{M}^{\text{PPPVC}}_{\text{MinSum}}$.

For conciseness we will denote with $\mathbf{X} = \mathbf{0}$ the absence of initial conditions to impose.

Finally, note that constraints (C.18)–(C.22) would also allow to impose *circularity conditions* to treat cases in which the availability of machines is periodic with periodicity $|\mathcal{T}| \cdot \Delta = T$; we do not enter into further details, as this would go beyond the purposes of this paper.

C.5.1.2 Further notation

We define some additional notation that we will use in the remainder of the document. Let:

- $J = (\mathcal{T}_J, N_J, M_J, \mathbf{B}_J, \mathbf{k}_J, \mathbf{q}_J, \mathbf{E}_J, \mathbf{X}_J)$ denote an *instance* (i.e. a particular case) of PPPVC$_{MinSum}$, and $\mathcal{T}_J, ..., \mathbf{X}_J$ the elements which define it. In the following, for an abuse of language we will often refer to a PPPVC$_{MinSum}$ instance as to a *problem* to simplify some explanatory parts;
- F_J denote the set of all feasible solutions of J;
- $\mathbf{s} \equiv (\mathbf{x}, \mathbf{y}, \mathbf{v}) \in F_J$ denote a feasible solution and $\mathbf{x}$, $\mathbf{y}$, $\mathbf{v}$ its *allocation*, *usage* and *migration* components; in particular, let symbol $\mathbf{x}^{\tau\prime} = (x_{ji}^{\tau})|_{j\in N, i\in M}^{\tau=\tau\prime}$ represent the component of $\mathbf{x}$ at period $\tau\prime \in \mathcal{T}$;
- z_J denote the standard objective function defined in (C.2) with the energy coefficients $\mathbf{E}_J$.

C.5.1.3 Description of the Algorithm

Let $\mathcal{A}_{\text{MinSum}}^{\text{PPPVC}}(J, z)$ be an algorithm capable of finding a solution $\overline{\mathbf{s}}_z$ for the PPPVC$_{MinSum}$ instance J which is optimal with respect to a generic objective function z. Obviously, running $\mathcal{A}_{\text{MinSum}}^{\text{PPPVC}}(J, z_J)$ is sufficient to find the optimal solution of PPPVC$_{MinSum}$ but in most cases the size of M_J, N_J, $\mathcal{T}_J$ make the problem unmanageable, even if theoretically tractable.
To solve PPPVC$_{MinSum}$ for such hard instances, we design a heuristic algorithm $\text{H}_{\text{MinSum}}^{\text{PPPVC}}$ which makes use of $\mathcal{A}_{\text{MinSum}}^{\text{PPPVC}}$ to solve more tractable subproblems and obtain good solutions in a rolling horizon fashion. The input parameters of $\text{H}_{\text{MinSum}}^{\text{PPPVC}}$ are:

- the instance $J = (\mathcal{T}_J, N_J, M_J, \mathbf{B}_J, \mathbf{k}_J, \mathbf{q}_J, \mathbf{E}_J, \mathbf{0})$ to solve;
- a *rolling window width* w;
- a *step width* u;
- a *evaluation function width* b;

We make the hypothesis that no initial conditions are given.
The basic idea is simple: we solve as many subproblems as necessary to cover the entire horizon $\mathcal{T}_J$: then a global solution to J is obtained by assembling the partial solutions of subproblems. $\text{H}_{\text{MinSum}}^{\text{PPPVC}}$ starts by solving subproblem:

$$J_0 = (\mathcal{T}_0, N_J, M_J, \mathbf{B}_0, \mathbf{k}_J, \mathbf{q}_J, \mathbf{E}_J, \mathbf{0}); \quad \mathcal{T}_0 = \{0, ..., w-1\}; \ \mathbf{B}_0 = (B_i^{\tau})|_{i\in M}^{\tau\in\mathcal{T}_0}$$

which is basically the restriction of J to the window of the first w periods of $\mathcal{T}_J$ and the associated submatrix of $\mathbf{B}$.

Problem J_0 is solved by calling $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_0, z_J+L_0)$ so as to get the solution $\overline{\mathbf{s}} = (\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{v}})$. L_0 is called *evaluation function*: it is used to get a lower estimate of the transfer costs in the b periods following the time window $\mathcal{T}_0$, i.e. $w \leq \tau < w + b$, due to the allocation decisions taken at $\tau = w - 1$. L_0 allows $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}$ to have a minimal awareness of the impact of such decisions. This is to exploit the fact that $\mathbf{B}$ is completely known a priori. We will explain shortly how does L_0 work.

Then the rolling time window is shifted forward of u periods so as to define the time horizon of subproblem $J_1 = (\mathcal{T}_1, N_J, M_J, \mathbf{B}_1, \mathbf{k}_J, \mathbf{q}_J, \mathbf{E}_J, \mathbf{X}_1)$ where:

$$\mathcal{T}_1 = \{u, ..., u + w - 1\}; \qquad \mathbf{B}_1 = (B_i^\tau)|_{i\in M}^{\tau\in\mathcal{T}_1}; \qquad \mathbf{X}_1 = \overline{\mathbf{x}}^{u-1}$$

J_1 is solved by calling $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_1, z_J + L_1)$. Figure C.3 explains precisely this phase with an example and can help understanding some key concepts.

Problem J_1 can be imposed initial conditions $\mathbf{X}_1$, as the solution of J_0, $\overline{\mathbf{s}}$, provides a deployment $\overline{\mathbf{x}}^{u-1}$ for the period immediately before the time window $\mathcal{T}_1$. The most interesting aspect is that since $u < w$, time windows $\mathcal{T}_0$ and $\mathcal{T}_1$ are partially overlapped, therefore the deployment $\overline{\mathbf{x}}^{u-1}$ is *not* the last one which is output by $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_0, z_J+L_0)$. On the contrary, $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_0, z_J + L_0)$ decided such assignment while being aware of the availability of machines for the remaining periods of $\mathcal{T}_0$, i.e. $u \leq \tau < w$.

In the example of figure C.3, $w = 3$, $u = 2$, and $b = 1$. When solving J_0, $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_0, z_J + L_0)$ provides a deployment $(\overline{\mathbf{x}}^\tau)|^{\tau\in\{1,2,3\}}$: the initial condition for subproblem J_1 is $\overline{\mathbf{x}}^2$. Such deployment has been determined while knowing the availability of machines at $\tau = 3$ and a lower estimate L_0 of how the allocation decisions at $\tau = 3$ would impact on the energy transfer costs in period $\tau = 4$. Therefore, $\overline{\mathbf{x}}^2$ can be considered to be a good initial condition for J_1, whereas if we had $u = w$, the initial condition of J_1 would be the deployment $\overline{\mathbf{x}}^3$, which $\mathcal{A}^{\text{PPPVC}}_{\text{MinSum}}(J_0, z_J + L_0)$ has decided with no knowledge of machines availability in the following time periods – except for the lower estimate provided by L_0.

The example in figure C.3 can also help explaining the evaluation function L_0. When solving J_0 we restrict the time horizon to $\mathcal{T}_0 = \{1, 2, 3\}$; however, since we have a complete knowledge of the $\mathbf{B}$ matrix, we know that if $x^3_{j2} = 1$, i.e. we allocate a copy of some application j on machine $i = 2$ at $\tau = 3$, we will have to move it in the next time period, since $B^4_2 = 0$. In such a case, the minimal transfer energy cost for application j will be:

$$\min_{i\in M: B_i^4 = 1} C_{j2i}$$

The expression of L_0 takes into account all the applications that we allocate on machines that are available at $\tau = 3$ but not at $\tau = 4$:

$$L_0 = \sum_{j \in N} \sum_{\substack{h \in M: \\ B_h^3 = 1 \\ B_h^4 = 0}} \left(\min_{\substack{i \in M: \\ B_i^4 = 1}} C_{jhi} \right) x_{jh}^3$$

The extension to the other subproblems and, in general, to the case $b > 1$ is quite straightforward: we omit it for conciseness. It can be easily seen that, in spite of the fact that choosing a large value for b may seem advantageous, such a choice would not pay off; indeed, the significance of the evaluation function decreases as b grows.

$\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ goes on by iteratively shifting the rolling time windows and solving all the subproblems. The number of subproblems to solve is $\overline{p} = 1 + \lceil \frac{|\mathcal{T}_J| - w}{u} \rceil$; the width of the time horizon of the last subproblem, $\mathcal{T}_{\overline{p}-1}$, may be less than w. The overall solution to problem J is built starting from the partial deployments obtained separately when solving the subproblems; for the overlapping time periods $\tau \in \mathcal{T}_{p-1} \cap \mathcal{T}_p$, $1 \leq p < \overline{p}$, $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ considers the deployment provided by the solution of J_p, i.e. the last run subproblem between the two. One last detail, the evaluation function for the last subproblem, $L_{\overline{p}-1}$, is equal to 0.

A brief recap on figure C.3 can help fix the main elements of $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$. In the example we have:

- $\overline{p} = 4$, i.e. four subproblems $J_0, \ldots, J_3$;
- subproblem J_0 is associated with time window $\{1, 2, 3\}$, has no initial conditions ($\mathbf{X}_0 = \mathbf{0}$), and it is solved by running $\mathcal{A}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}(J_0,\, z_J + L_0)$: evaluation function L_0 considers the availability of machines at $\tau = 3$. $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ saves partial deployments $\overline{\mathbf{x}}^1$ and $\overline{\mathbf{x}}^2$;
- subproblem J_1 has time window $\mathcal{T}_1 = \{3, 4, 5\}$, initial condition is $\mathbf{X}_1 = \overline{\mathbf{x}}^2$ and evaluation function L_1 is based on $(B_i^\tau)|_{i \in M}^{\tau=6}$; $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ saves partial deployments $\overline{\mathbf{x}}^3$ and $\overline{\mathbf{x}}^4$;
- subproblem J_2 spans over time window $\{5, 6, 7\}$, has initial condition $\mathbf{X}_2 = \overline{\mathbf{x}}^4$ and a evaluation function L_2 based on $(B_i^\tau)|_{i \in M}^{\tau=8}$; $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ saves deployments $\overline{\mathbf{x}}^5$ and $\overline{\mathbf{x}}^6$;
- finally, subproblem J_3 is associated with time window $\{7, 8\}$ and has initial condition $\mathbf{X}_3 = \overline{\mathbf{x}}^6$, while $L_3 = 0$; $\mathrm{H}_{\mathrm{MinSum}}^{\mathrm{PPPVC}}$ saves partial deployments $\overline{\mathbf{x}}^7$ and $\overline{\mathbf{x}}^8$.

C.5.2 Greedy schema

The rolling horizon heuristic is like a divide-and-conquer algorithm that decomposes the entire time period in small time horizons in which we have a PPPVC_{MinSum} instance, solved by linear programming. The advantage in doing so is that the runtimes of the

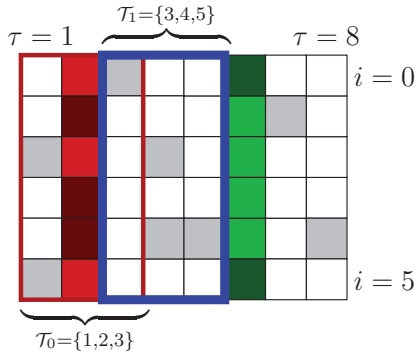


FIGURE C.3: An example of $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ running on a small instance J with a set of 6 machines and a time horizon of 8 periods. The figure shows the availability matrix $\mathbf{B}$ and the rolling time window at iteration $p = 1$ (blue box). The rolling time window has width $w = 3$, while the step width is $u = 2$ and the bound width is $b = 1$. Subproblem J_0 (red box) has already been solved and the deployment $\overline{\mathbf{x}}^1$ at period $\tau = 2$ (red column) can act as initial condition for subproblem J_1. At the same time, the knowledge of the availability matrix at period $\tau = 6$ (green column) allows to compute, when solving J_1, a lower estimate of transfer costs due to allocation decisions at period $\tau = 5$.

linear program can be reduced in considering a set of small time horizons instead of the entire time period. This approach however has a drawback: one needs to state in input how we decompose the entire time period. In this part, we will introduce a greedy heuristic where this parameter is not required. Instead of the decomposition in problem instances, proposed in the rolling horizon heuristic, the greedy heuristic views the solution of PPPVC_{MinSum} as a set of consecutive execution blocks. Below, we define this notion.

C.5.2.1 Definitions

Definition C.5. (Consecutive execution block). We define a consecutive execution block as a tuple $Ceb = (\tau, M', N', D, A')$. Here $\tau \in \mathcal{T}$, $M' \subseteq M$, $N' \subseteq N$, $A' \subseteq M' \times N'$ and $D = (d_0, \ldots, d_{|M'|})$ is such that $d_i \in \{1, \ldots |\mathcal{T}|\}$.

A consecutive execution block $Ceb = (\tau, M', N', D, A')$ describes a deployment of applications on machines, starting from a period (τ). In the definition, M' and N' define the subset of applications and machines assigned in the block. A' is a set of assignment couples. If $(x, y) \in A'$ then an instance of the application x is deployed on the machine y during d_y times units.

Each Ceb can be captured by a two dimensional cut stating the machines that it uses, their capacities and their availabilities. As illustration, we consider in figure C.4, five machines and seven applications. We used the gray color to state that a machine is not available. In the first block of this figure (Ceb_1), five applications are deployed on

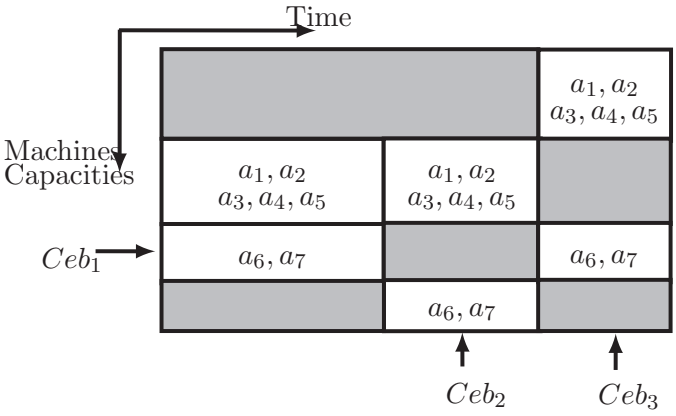


FIGURE C.4: Ceb illustration on 4 machines

machines 2 and two are deployed on machine 3. In the second Ceb, the deployment on machines 2 are not changed. But the applications a_6 and a_7 are now deployed on machine 4. This also implies that between Ceb_1 and Ceb_2, these applications have been migrated.

One interest in this representation is that it suggests that we might need multiple Cebs for solving PPPVC$_{MinSum}$. For instance, Ceb_1 is not sufficient for solving the problem. This is because it cannot ensure that all application instances will be deployed in all time units. We consider the following definition.

Definition C.6. (Valid, ordered Ceb list). Given a list of Cebs: $(\tau_1, M_1, N_1, D_1, A_1), \ldots (\tau_l, M_l, N_l, D_l, A$ we will say that it is valid if: (1) according to the definition of the A_is and the one of B, there is no assignment made at a period on an unavailable machine; (2) at each period and for each machine, the number of assigned applications does not exceed its capacity. The list is ordered if $\tau_1 < \cdots < \tau_l$.

Such a list can be a potential solution to the resolution of PPPVC$_{MinSum}$. The validity is important for respecting the constraints of the problem. The ordering ensures that the Cebs are set one after another. However, these conditions are not sufficient for obtaining a solution to the problem. We consider below other definitions.

Definition C.7. (Ending point of a Ceb list). Given an ordered list of Cebs: $(\tau_1, M_1, N_1, D_1, A_1), \ldots (\tau_l,$ we define its ending point as the first period $\tau' \geq \tau_l$ where according to the assignments $A_1, \ldots, A_l$, we have for a given application j less than k_j copies assigned at the period τ'.

For instance in figure C.4, the ending point of Ceb_1, Ceb_2 corresponds to the period at which a_1 must be migrated from the machine 2 to machine 1.

The ending point of the ordered list $L = Ceb_1, \dots Ceb_l$ will be denoted by $\alpha(Ceb_l)$ or $\alpha(L)$. We will also refer to this point as the ending period of Ceb_l. One can notice that if $\alpha(L) < |\mathcal{T}|$, then L does not solve PPPVC$_{MinSum}$.

Definition C.8. (Adjacent Ceb). Let us consider a valid and ordered Ceb list: $Ceb_1, \dots Ceb_{l-1}, Ceb_l$. We will say that Ceb_{l-1} and Ceb_l are adjacent if the following conditions are met: (1) $\tau_l = \alpha(Ceb_{l-1}) + 1$; (2) A_l is defined by reassigning the applications x such that $(x, y) \in A_{l-1}$ and y is not available at the period τ_l.

For solving PPPVC$_{MinSum}$, we must consider adjacents Cebs for ensuring that all application instances are always assigned. In figure C.4, it is straightforward to notice that Ceb_1 and Ceb_2 are adjacent. Indeed, Ceb_2 reassigns the applications a_6, a_7 that must me moved at the ending point of Ceb_1.

Now, solving the allocation problem with Cebs is based on the following fact.

Fact C.9. Any solution for the allocation problem can be formulated as a valid and ordered list of adjacent (and valid) Cebs $L = Ceb_1, Ceb_2, \dots, Ceb_l$ such that: (1) All application instances are deployed in Ceb_1; (2) $\alpha(L) = |\mathcal{T}|$

This fact is at the core of greedy search scheme. Based on it, our idea for solving the allocation problem is to greedily construct the Ceb list that will respect these two conditions. The process works as follows.

At the beginning, we have an empty Ceb list. We choose an initial Ceb ($\tau = 0$) in which all application copies are deployed. The initial Ceb is chosen such as to minimize the total energy consumption. Then, the Ceb is put in the list and we compute the ending point e. If $e = |\mathcal{T}|$, the scheme is stopped; otherwise, one chooses an adjacent Ceb and adds it such as to have an ordered list. With the second Ceb put in the list, one computes the ending point e of the resulting Ceb list and compares it with $|\mathcal{T}|$. If the two values differ, we repeat the process by choosing another adjacent Ceb; otherwise, we stop the scheme execution. The pseudo-code of the described process is given in Algorithm C.5.

The proposed greedy search scheme occurs through multiple choices of adjacent Cebs. For a complete algorithm description, we must detail the choice of adjacent Cebs. This is addressed in the following subsection.

C.5.2.2 Choosing a valid Ceb

From the results of the section C.3.3, we can easily notice that given Ceb_l, the choice of Ceb_{l+1} for the minimization of the PPPVC$_{MinSum}$ objective function is NP-hard. For

GreedySearch()

1 Choose a valid block, $Ceb_0 = (0, M_0, N_0, D_0, A_0)$ s.t.
$(\forall j) \left(\sum_{\substack{x \in M \\ (x,j) \in A_0}} 1 \right) = k_j$
2 Add Ceb_0 to $CebList$
3 $e = \alpha(CebList)$
4 $l = 1$
5 **while**$(e < |\mathcal{T}|)$
6 Choose a valid ceb Ceb_l that is adjacent to Ceb_{l-1}
7 Add Ceb_l to $CebList$
8 $e = \alpha(CebList)$
9 $l = l + 1$
10 **return** $CebList$

Figure C.5: Greedy Search for building a provisioning plan.

easing this computation, one can use randomization. The idea is to randomly pick machines and deploy on applications instances while respecting the capacities constraints. Doing so however, we are completely unaware of the combinatory in the minimization of the energy consumption. In this part, we propose to reduce the search space in the Ceb_{l+1} by introducing a search principle.

Principle C.10. (Move When Forced (MWF)). In the construction of Cebs, between two periods e and $e + 1$, we move an application copy from the machine on which it is deployed iff the machine is no more available at the period $e + 1$.

The idea in MWF is to avoid avoid migrations when it is possible. This choice is motivated by the fact that any migration increase the energetic cost of the deployment plan.

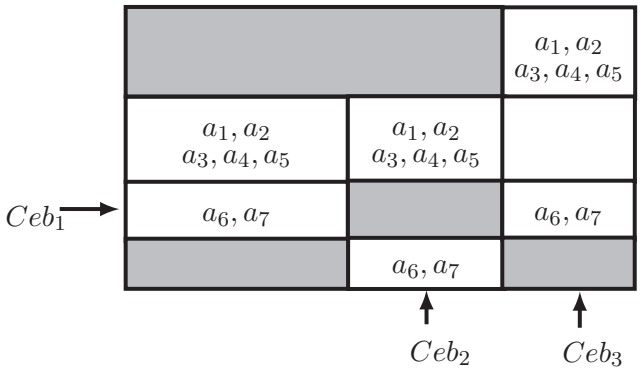


Figure C.6: Example of assignment that does not follow the MWF principle.

In figure C.6, we give an example of assignment that does not respect the MWF principle. The violation of the principle is due to the fact that in Ceb_3, the applications

a_1, a_2, a_3, a_4, a_5 are migrated whereas the machine 2 is always available. For respecting the principle, these applications must be kept in the machine 2 in Ceb_3.

We cannot guarantee that the optimal solution is always constrained within the MWF principle. Moreover, it is obvious to notice that PPPVC$_{MinSum}$ remains NP-hard even with MWF; indeed, the proofs of section C.3.3 still hold. Despite these drawbacks, let us observe that the principle reduces the search space of Cebs in Algorithm C.5. Indeed, it states to only consider Cebs that are such that the deployment of applications on machines does not change from their starting to their ending period. In Algorithm C.5, this suggests that for building Ceb_{l+1} from Ceb_l, we can only focus on the how we re-deploy (at the period $\alpha(Ceb_l) + 1$), the applications that must be moved at the period $\alpha(Ceb_l)$. In what follows, we will show how to build Cebs in Algorithm C.5 according to the MWF principle.

C.5.3 Energy aware algorithms for CEB design

A simple solution for building Cebs consists in using randomization. Doing so, we do not take into account the minimization of the energy consumption in Ceb design. This will be the case in the solution that we propose. In this solution, we distinguish about two design cases in Algorithm C.5. The first case is the design of the initial Ceb Ceb_1. The second case is the design of an intermediate (or final) Ceb. Here, we assume that after building a Ceb list $Ceb_1, \ldots, Ceb_l$, we are looking for an adjacent Ceb Ceb_{l+1}. In this part, we propose a solution for both of these cases. We start with the intermediate case.

C.5.3.1 Building an intermediate Ceb: local objective function

Given $Ceb_1, \ldots, Ceb_l$, we want to build Ceb_{l+1}. The first computational challenge here is to decide on the local energetic costs that we must minimize when building Ceb_{l+1}. For this we observe that from Ceb_l to Ceb_{l+1}: (1) some applications might be transferred; (2) we might not have the same deployment of applications on machines; (3) some machines that were not used in Ceb_l might be used. From these observations, we can conclude that Ceb_{l+1} must be built such as to minimize the sum of : the energy consumed by applications in Ceb_{l+1}, the energy required for transferring applications between Ceb_l and Ceb_{l+1} and the new base energy to account in Ceb_{l+1}. When considering the objective function of PPPVC$_{MinSum}$, we can notice that this choice proposes to minimize its *local value* between Ceb_l and Ceb_{l+1}.

The minimization of the local value of the objective function is not always a good choice. Indeed, this choice suggests to favor the Cebs whose ending period are short because the shorter is the ending period of a Ceb, the smaller we can expect that its energy consumption will be. Doing so however, the Algorithm C.5 will tend to use multiple Cebs for building the complete Ceb list in which all application are deployed until the final period t. However, the presence of multiple Cebs suggest that we make many transfers and therefore consume more energy. In figure C.7, we provide an illustration. Let us suppose that the energy consumed by an application on a machine is equal to 1 unit. Let us also assume that the transfer costs and the base energy are all equal to one unit. While the local minimization of Cebs propose to have the complete Ceb list Ceb_1, Ceb_2, Ceb_3, the best solution consists of choosing the list $Ceb_1, Ceb_{2'}$.

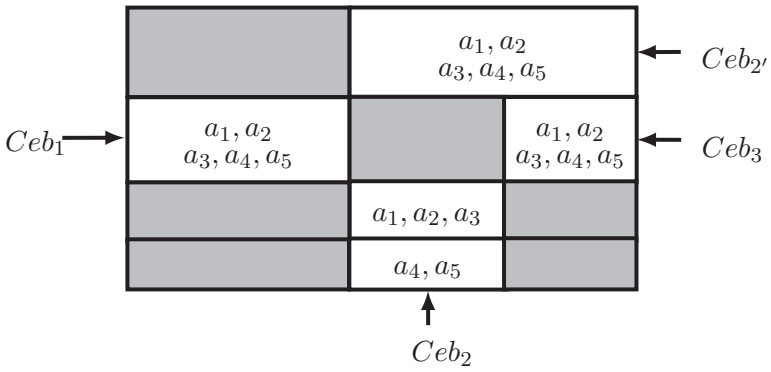


FIGURE C.7: Example with 4 machines, 5 applications and 2 possible list of Cebs as solution.

The previous observation leads us to conclude that in the choice of Ceb_{l+1} from Ceb_l, we must take into account both: the energy consumption and the time in which machines will be available. More precisely, given a Ceb at a period $\alpha(Ceb_l)+1$, let us assume that its machine i will be available during d_i time units; let us also assume that the local value of the energy consumed on it is S_i; this includes: the sum of energy consumed by applications deployed on it, the sum of energy for making transfers on it and eventually a base energy if i was not used in Ceb_l. Then, we propose to choose choose Ceb_{l+1} if the value $\sum_{i \in M} \frac{Z_i}{d_i}$ is minimal among all other valid and adjacent Cebs.

We will refer to $\sum_{i \in M} \frac{Z_i}{d_i}$ as the *mean energy consumed in a time unit* when choosing Ceb_{l+1}. We propose to consider this objective function in the sequel. In the next, we will formulate a computational process for building Cebs according to it.

C.5.3.2 Building an intermediate Ceb: a greedy approach

Let us re-consider the local solution $Ceb_1, \ldots, Ceb_l$. Following the MWF principle, we can simply compute the subset of applications instances that must be migrated at the period $\alpha(Ceb_l) + 1$. The construction of Ceb_{l+1} will then be based on this subset. For each application j that is concerned by a migration, we propose first to compute the set $M(j)$ of machines where the instances to migrate are located on. We also compute the set $M_a(j)$ of machines where the instances of j can be migrated to while respecting (1) the constraint of not assigning the same application twice on a machine; (2) the constraint of not exceeding machine capacity. The problem for the construction of Ceb_{l+1} is to migrate for all applications j, the instances that are located on $M(j)$ on the subset of machines $M_a(j)$ such as to minimize the mean energy per time unit.

For tackling this *local problem* let us observe that we can adapt our prior ILP modelling. However, this is not necessarily a good option regarding the runtimes since the local problem is NP-hard. Indeed the NP-hardness proof of PPPVC$_{MinSum}$ proposed in [Cérin et al., 2013] can be applied here. For easing the construction, we propose a greedy algorithm. Its main idea consists of considering separately the resolution of the problem for the instances related to an application. This means that we subdivide our local problem (migration of instances in $M(j)$, for all j) into a successive set of sub-problems, each related to a unique application. *The simplification that this greedy approach brings is that since the number of instances of a given application is smaller than the total number of instances of all applications, it is easier to solve each sub-problem than to solve the local problem.*

In subdiving the local problem, we obtain a computational process with two stages for the construction of Ceb_{l+1}. In the first, we order at the period $\alpha(Ceb_l) + 1$ the applications that are concerned by a migration. Then, according to this order, we successively migrate the instances of a given application.

The challenge in the greedy approach is to find a good basis for deciding on the ordering of applications. In the next, we will introduce some metrics for this purpose.

Definition C.11. (Minimal induced energy). Given the Ceb list $Ceb_1, \ldots, Ceb_l$, the energy induced by the migration of the application j from the machine u to the machine v at the period $\alpha(Ceb_l) + 1$ is

$$Ei(j, u, v) = (E^b_v + E^+_{jv}).d_v + C_{juv};$$

Here, we assume that starting from the period $\alpha(Ceb_l) + 1$, the machine v is available during d_v time units.

Ei captures the minimal energetic cost that must be taken into account for a migration, according to the MWF principle.

Definition C.12. (Minimal expected energy). Given the Ceb list $Ceb_1, \ldots, Ceb_l$, we define the energy expected for the move of the application j as

$$\mathsf{E}[j] = \sum_{u \in M(j)} \sum_{v \in M_a(j)} \frac{Ei(j,u,v)}{|M_a(j)|}$$

$\mathsf{E}[j]$ is a sum of the mean induced costs per application instances. For its computation, we consider all possible moves of an application instance.

We propose to base the ordering of applications in the greedy approach on the minimal energy expected: *the applications whose minimal expected energy is greater must be prioritized over those whose expected energy are smaller*. The motivation behind this choice is to give more possibilities for the deployment of applications whose migration costs are a priori big.

For illustrating this motivation, let us assume that we decide to migrate first the instances of an application j_1 and then the ones of j_2. As we migrate the instances of j_1, the machines are more loaded and we can reach a situation where for some of the them, the maximal capacity is reached. This means that if at the beginning of the migrations of j_1, $M_a(j_2)$ was the set of possible machines on which the instances of j_2 could have been deployed, at the end of the migrations of j_1, one must perhaps consider a subset of this initial set. Now, if the minimal expected energy of j_1 was greater than the one of j_2, it will be a good choice to consider j_1 primarily in order to have more opportunities in the minimization of the energetic cost issued from the deployment of this application.

With the description of the metric used for ordering, we can now refine the description of our computational process for the construction of Ceb_{l+1}. The process has two stages. In the first stage it performs three main tasks: (1) computation at the period $\tau = \alpha(Ceb_l)+1$ of the instances and applications concerned by a migration; (2) computation at the period τ of the values of Ei and E; (1) sorting of applications by expected energy consumption (the sorting gives priority to higher values of E). The output ordering of applications $j^1, \ldots, j^v$ that is produced will be used in the second stage for building Ceb_{l+1}.

After this ordering stage, we complete the computation of Ceb_{l+1} within an iterative stage. In each iteration, we move the instances of an application. More precisely, let us assume that from the first stage, we obtained the ordering $j^1, \ldots, j^v$. At each iteration $\lambda \in \{1, \ldots, v\}$, we move the instances of j^λ. We repeat the iterations until moving all applications.

For refining the description of our greedy approach, we must now state how we proceed for deciding on the migration of the instances of a given application. This will be addressed in what follows.

C.5.3.3 Building an intermediate Ceb: general algorithm

Let us assume that we must migrate the instances of the application j^λ. For deciding about the actions to perform our idea is to modelize the problem as an assignment problem that can be solved by a classical assignment method as the Hungarian algorithm.

For this, we propose first to compute the set $Ma(j^\lambda)$ of potential machines on which the migrations can be performed. For each machine in $i \in Ma(j^\lambda)$, we then compute the maximal time $mt(i)$ during which it will be available without interruption, starting from the period τ. Finally with each possible assignment of instance $(x, y) \in M(j^\lambda) \times M_a(j^\lambda)$, we associate the *mean energy consumed in a time unit* $P[(x, y)] = \dfrac{Ei(j^\lambda, x, y)}{mt(y)}$. Given these costs, we propose to choose for the migrations of j^λ the assignment function r that will lead to the minimization of

$$P(j^\lambda) = \sum_{x \in M(j^\lambda)} \sum_{y \in M_a(j^\lambda)} P[(x, y)].r[(x, y)]$$

Here $r[(x, y)]$ is a boolean quantity set to 1 if the copy located on machine x is moved to y. r also ensures that instances are moved towards distinct machines.

It is obvious to notice that the computation of r can be done in polynomial time in using the Hungarian method. In our formulation we considered the energy efficiency cost for being coherent with the analysis made in Section C.5.3.1. Alternatively, one can consider at this stage the local value of the objective function, also described in this section.

Now that we have a solution for migrating the instances of an application, we can summarize our general computational process for the choice of Ceb_{l+1}. This is done in Algorithm C.8. When combined to Algorithm C.5, the Algorithm C.8 states how we can build all the intermediate and and final Ceb. What misses for a complete description of Algorithm C.5 is how we design the first Ceb, Ceb_1. This point is addressed in the sequel.

C.5.3.4 Building an initial Ceb

As stated in the beginning of Section C.5.3, we can distinguish between two types of Cebs: intermediate and initial ones. Below, we showed how to compute intermediate Cebs; let us now focus on the design of the initial one (Ceb_1).

For the design of the initial Ceb, we propose to adapt the computational process of the intermediate phase to the specificities of the initial Ceb: (1) the base energy must be counted for any used machine; (2) we assume that all application instances must be migrated, however, the transfer costs for the migrations are all null. One can notice that it is easy to adapt Algorithm C.8 for including these two considerations.

C.5.4 Variants of the energy aware algorithm

In Algorithm C.8, we subdivided the migration of a set of instances into various assigment problem, each related to the migration of a set of instances related to a given application. These latter problems are then solved by the Hungarian method. Let us however observe that these subdivisions will produce an approximated solution of the initial migration problem. Indeed, one can observe that since the Hungarian method is polynomial, the resolution of our set of sub-problems can be done in polynomial time while our initial problem is NP-hard. The key in our approximation is in the optimization of the base energy consumption. Indeed, the decision to migrate instances towards a *free* machine on which there is no instance has an impact on the total base energy consumption of the migration problem. In the migration problem, these decisions must be done in considering all instances. With our subdivision, each subset of instances related to an application will decide independently on the *free* machines that will be used.

One interest of Algorithm C.8 is that we can formulate some variants from it. Let us refer to the proposed algorithm as *Ceb_max[ME]* (the "max" suffix here is for the ordering in the second stage and [ME] refers to the local objective function: *Mean Energy per time unit*). An idea for a variant is to reconsider the sorting of the applications in the first phase. Thus, we can formulate *Ceb_rand[ME]* where the applications are initially sorted in a randomized way or *Ceb_min[ME]* where initially, applications are sorted in considering the one that consume the minimal expected energy first. Another possible formulation is that in the choice of Cebs, instead of minimizing the energy consumed in a time unit, one can consider only the local energy consumption as formulated in Section C.5.3.1. With this, we can have three other heuristics for the local choice of Cebs: *Ceb_max[LO]*, *Ceb_rand[LO]* and *Ceb_min[LO]*.

We end here the description of heuristics approaches for the resolution of PPPVC$_{MinSum}$ At this stage, let us observe that our integer linear model and heuristics can easily adapted to the resolution of PPPVC$_{MinMax}$. For the integer linear programming, it suffices to express the objective function as the the maximum (instead of the sum) of energy consumed on machines. Our heuristics occur through the subdivision of the service provisioning problem into subproblems. For an adaptation to PPPVC$_{MinMax}$, one can simply change the objective function in the subproblems for capturing the maximal energy consumed in a machine. In the next section, we present an experimental evaluation of the solutions proposed in this study.

EnergyChoiceAdjCEB(C)

```
argtype  C: cebOrderedList;
argcond  C = {c_1,...,c_{l+1}};
1  Compute (from CebList) the list J of applications concerned by a move
2  //First Stage: ordering
3  foreach(j ∈ J)
4     Compute M(j) and M_a(j)
5  foreach(j ∈ J)
6     foreach(u ∈ M(j)) foreach(v ∈ M_a(j))
7        Compute Ei(j,u,v)
8     Compute E[j]
9  τ_l = α(CebList) + 1
10 N_l = J
11 Sort the applications in J on the E[j]s and get the order j^1,...,j^v
12 //Second Stage: iterative migrations
13 λ = 1
14 while(λ < v)
15    Update M_a(j^λ)
16    Associate with each couple (x,y) ∈ M(j^λ) × M_a(j^λ) the cost
   P[(x,y)] = Ei(j^λ,x,y) / mt(y)
17    Compute (by the Hungarian algorithm) the assignment function
      r : M(j^λ) × Ma(j^λ) ⟶ {0,1} that minimizes
      P(j^λ) = Σ_{x∈M(j^λ)} Σ_{y∈M_a(j^λ)} P[(x,y)].r[(x,y)]
18    foreach((x,y) ∈ M(j^λ) × M_a(j^λ))
19       if(r[(x,y)] = 1)
20          state that in Ceb_{l+1}, an instance of x will be assigned to the machine
   y for the time duration mt(y)
21    λ = λ + 1
22 return Ceb_{l+1}
```

Figure C.8: Energy aware choice of adjacent CEBs.

C.6 Experiments

We did several experiments with the ILP modeling and the heuristics proposed in this study. We have three main purposes in the experiments. The first was to compare the quality of the solutions by the proposed heuristics with the results of the ILP-based algorithm. The second objective was to determine which one among the heuristics offers the best compromise between solution quality and computational time.

The last objective was to understand the differences in the behavior of heuristics. We will discuss on these points in this section. We organize our presentation in two general parts. In the first, we describe the instances used for the validation. Then, we present the outputs of our algorithms.

C.6.1 Instances

We generated a set of 120 problem instances. Each instance was issued from a configuration that defines: a number of applications, a number of machines, the time period, bounds on instances numbers and capacities of machines and the probability of availability of a machine. We considered in our experiments 8 configurations summarized in Table C.4.

Configuration	N	M	$\|\mathcal{T}\|$	q_{min}	q_{max}	k_{min}	k_{max}	av.
#1	30	50	5	3	7	3	4	85%
#2	30	50	10	3	7	3	4	85%
#3	60	100	10	3	7	3	5	85%
#4	90	150	12	3	7	3	5	85%
#5	100	150	12	3	6	4	5	80%
#6	120	180	16	3	7	3	5	85%
#7	130	170	16	3	7	4	5	80%
#8	100	180	16	3	7	3	4	80%

TABLE C.4: Description of configurations

Let us observe that in most experimental studies related to clouds[2], the mean availability observed in data centers is close to 99%. Here, we chose a lower rate for taking into account the presence of volunteer machines. We set the time period for building a plan between 5h and 16h. Somehow, these bounds suggest that we can efficiently predict the availability matrix of the machines data centers between 5 and 16 hours. Let us observe that prior work in volunteer computing showed that this is reasonable [Kondo et al., 2008].

[2]International Working Group on Cloud Computing Resiliency: http://iwgcr.org/

From each of our 8 configurations, we generated 15 problem instances. In each instance, the machine availability matrix $\mathbf{B}$ has been generated according to the aforementioned mean availability parameter; the values of $k_j, j \in N$ were randomly chosen between k_{min} and k_{max}, and the same was done for $q_i, i \in M$ with respect to q_{min} and q_{max}. The process to generate the energy consumption coefficients of matrices $\mathbf{E^b}$, $\mathbf{E^+}$ and $\mathbf{C}$ is described below.

C.6.1.1 Energy consumption of an instance

The generation of the energy consumption per instance was done in three stages. The first stage consisted of generating an instance network i.e an organization of the machines within a communication network. In it, there are two types of communication between two machines: local communication (done with switches and without crossing a router) and global communication (include at least one router). We set the diameter of the network to 3; this means that there is at most 3 routers between two machines that communicate. The decision to put $0, 1, 2$ or 3 routers were based on randomness and some rules. Firstly, we randomly ranged the machines in groups. Given a total of m machines, we created approximately $8\log(m)$ groups. The machines within a group were connected directly with a switch (without routers). Then, we associated each group with a router that serves as its frontal for external communication. Finally, we randomly connected the routers such as to ensure that there is at most one intermediate router that join two groups.

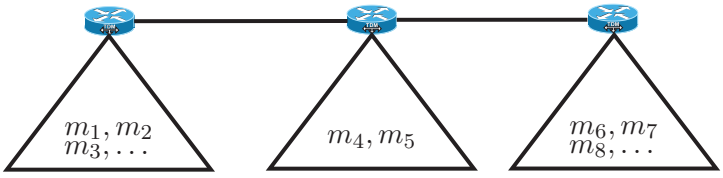


FIGURE C.9: Example of instance network

An illustration of an instance network is provided in figure C.9. Here m_1, m_2 and m_3 are machines of a group. They are connected with a switch. m_1 and m_4 are connected throughout two routers; m_1 and m_7 are connected throughout three routers.

After the generation of an instance network, the next stage consisted of associating energy values with the nodes of the network. Given an instance network, we randomly associated Ethernet cards, machines and routers of the network with random power consumption costs chosen from predefined data sets. Ethernet cards power consumption

are the ones defined in [Chiaravalloti et al., 2011]. For routers and machines power consumption, we used open data available for Cisco routers[3] machines[4].

The third stage in the generation of the energy consumption values consisted of inferring the values from the built setting. Assuming that the power consumption loaded for a machine i (in the network) was P_i, we set its base energy to $P_i.\Delta_t$ where $\Delta_t = 1h$. The overhead energy per instances were set as a fraction between 1 and 10% of the base energy. Finally, we generated the energy required for transferring applications with the Ecofen model [Orgerie et al., 2011]. This model provides analytical formula for estimating the energy consumption in communication that include routers and switches. We considered that it will take between 1 and 5 minutes to transfer an application between two machines. For the sake of reproducibility, the set of instances that we used in our experiments are obained based on he source code available online[5].

Now that we described our instances, in the next, we will present the obtained results.

C.6.2 Solutions of ILP and the rolling horizon heuristic

The ILP-based algorithm has been run on small instances only, i.e. on instances generated starting from configurations #1 and #2: for the instances that descends from the other configurations, the determination of an exact solution is too heavy and leads to a failure in the computation due lack of memory. Table C.5 represents the results of both the ILP-based algorithm (referred to as X) and the rolling horizon heuristic algorithm $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ (referred to as H). The columns represent:

- the name of the instance,
- the value s_X of the optimal solution of X and the time t_X to compute it,
- the initial value LB_r of the lower bound, i.e. the value of the LP relaxation of the problem,
- the gap $\%\frac{s_X}{LB_r}$ between the optimal solution and the initial value of LB,
- the value s_H of the solution of H and the time t_H to compute it,
- the gap $\%\frac{s_H}{s_X}$ between the solution of H w.r.t that of X, and finally
- the gap $\%\frac{s_H}{LB_r}$ between the solution of H and the initial lower bound.

[3] See the appendix of the document http://lib.tkk.fi/Dipl/2012/urn100676.pdf
[4] http://www.complang.tuwien.ac.at/anton/computer-power-consumption.html
[5] http://lipn.univ-paris13.fr/~cerin/software/testing.tar.gz

For the solving of X, the threshold gap value has been set at $\overline{g} = 1\%$, and a time limit of one hour has been imposed. The parameters of the heuristic algorithm $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ for this instances have been chosen as follows: $w = 3$, $u = 2$, and $b = 1$; moreover, each subproblem has been given a time limit of 1200s. One can observe that:

instance	s_X	t_X	LB_r	$\%\frac{s_X}{LB_r}$	s_H	t_H	$\%\frac{s_H}{s_X}$	$\%\frac{s_H}{LB_r}$
1-01	896178	1706.65	886523.68	1.09	897893	940.42	0.19	1.27
1-02	749447	95.21	743430.57	0.81	749078	214.96	-0.05	0.75
1-03	484018	3600.49	477810.84	1.30	485485	1320.33	0.30	1.58
1-04	890179	646.50	881405.95	1.00	896414	1380.31	0.70	1.67
1-05	733593	3600.12	725200.57	1.16	741660	2403.13	1.09	2.22
1-06	480074	1559.00	474445.97	1.19	480803	1340.64	0.15	1.32
1-07	938582	3600.18	922422.66	1.75	941028	2401.50	0.26	1.98
1-08	783543	3600.18	770983.08	1.63	783121	1384.12	-0.05	1.55
1-09	508302	2534.18	501912.58	1.27	509250	955.34	0.19	1.44
1-10	904051	961.40	896409.03	0.85	910785	1659.95	0.74	1.58
1-11	763952	160.25	757793.63	0.81	766483	1455.96	0.33	1.13
1-12	496902	3600.48	489802.02	1.45	499273	2404.44	0.47	1.90
1-13	867297	3600.46	857147.00	1.18	869760	1766.79	0.28	1.45
1-14	724622	3482.39	718524.32	0.85	727534	1919.81	0.40	1.24
1-15	476746	3600.13	467670.59	1.94	477874	2402.90	0.24	2.14
2-01	2251226	3600.23	1849607.52	21.71	1869367	3761.36	-20.43	1.06
2-02	1557018	3602.21	1510761.18	3.06	1526500	3862.64	-2.00	1.03
2-03	1015663	3601.71	996176.87	1.96	1008419	2003.98	-0.72	1.21
2-04	1993220	3600.27	1927008.78	3.44	1954332	3849.41	-1.99	1.40
2-05	1602416	3602.15	1517633.35	5.59	1542040	4105.45	-3.92	1.58
2-06	1048699	3600.21	1016848.45	3.13	1034999	3934.96	-1.32	1.75
2-07	1846205	3600.19	1806873.22	2.18	1825439	2893.27	-1.14	1.02
2-08	1456212	3600.26	1379380.74	5.57	1407554	4919.36	-3.46	2.00
2-09	947427	3600.18	931224.08	1.74	946520	3052.72	-0.10	1.62
2-10	1826786	3601.27	1784687.28	2.36	1821341	5767.03	-0.30	2.01
2-11	1496544	3600.18	1455984.85	2.79	1479731	4973.55	-1.14	1.60
2-12	989479	3600.30	965759.58	2.46	982771	5809.80	-0.68	1.73
2-13	1874976	3600.17	1825726.16	2.70	1851715	4900.19	-1.26	1.40
2-14	1464040	3601.97	1430398.41	2.35	1452263	2929.57	-0.81	1.51
2-15	992173	3600.21	965201.11	2.79	982205	2247.02	-1.01	1.73

TABLE C.5: Results of the ILP-based algorithm and the $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ heuristic on instances of configurations #1 and #2.

- the initial value of the LB is in general very good, as the optimal solution of the exact algorithm X is always near to it;
- the heuristic algorithm $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ can provide good solutions, as the gap w.r.t the lower bound is in most of the cases less than 2% (column $\%\frac{s_H}{LB_r}$);
- for more complicated instances, $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ is not only faster that X, but is sometimes capable of yielding an better solution (instances with $\%\frac{s_H}{s_X} < 0$).

The first of the three observation is particularly important, as it states the the quality of the lower bound offered by the LP relaxation of model $\mathcal{M}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ is high. As a consequence, even when it is not possible to compute an exact solution of an instance, the

solutions yielded by the heuristic solutions can be compared to the LB. That is what we will for what concerns the instances generated from configurations #3 to #8.
Table C.6 show the results of rolling horizon heuristic algorithm $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ (again referred to as H). The columns after the name of the instance represent:

- the time t_{LB_r} to compute the initial value of LB, and the lower bound itself, LB_r,
- the value s_H of the solution of H and the time t_H to compute it,
- the gap $\%\frac{s_H}{LB_r}$ between the solution of H and the initial lower bound.

The parameters of the heuristic algorithm $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ for this instances have been chosen as before: $w = 3$, $u = 2$, and $b = 1$, and a time limit of 1200s for each subproblem. The

instance	t_{LB_r}	LB_r	s_H	t_H	$\%\frac{s_H}{LB_r}$
3-01	1831.4	4167901.71	4227538	19662.12	1.41
3-02	1828.4	3305405.45	3357322	11333.72	1.55
3-03	1827.7	2147851.28	2232913	10175.91	3.81
3-04	1819.4	4973040.46	5053814	21214.64	1.60
3-05	1819.3	4085212.73	4155926	20497.15	1.70
3-06	1823.3	2577570.73	2621794	11971.45	1.69
3-07	1840.9	3971590.40	4035897	19439.77	1.59
3-08	1838.1	3290266.23	3338863	15481.89	1.46
3-09	1832.3	2142820.77	2213897	3754.47	3.21
3-10	1836.4	4400374.44	4468776	20096.71	1.53
3-11	1833.5	3545883.71	3600244	20972.15	1.51
3-12	1834.5	2263383.64	2312729	10754.07	2.13
3-13	1812.4	4191775.06	4249681	17828.55	1.36
3-14	1824.8	3288416.41	3338704	21354.51	1.51
3-15	1825.6	2186498.70	2218960	10994.75	1.46

TABLE C.6: Results of the $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ algorithm on instances of configurations #3.

computation of the initial LB has been imposed a time limit of 1800s. The results of $\mathrm{H}^{\mathrm{PPPVC}}_{\mathrm{MinSum}}$ are somehow contradictory: on one hand, they are very good for what concern the energetic cost, which is likely to be very close to that of the optimal solution. On the other hand, their determination is time-consuming.

C.6.3 Solutions of the greedy algorithms

In this part, we considered the greedy algorithms of Section C.5.4. In the case of *Ceb_rand[Z]*, we ran the heuristics 30 times and keep the mean runtime and the mean energy consumption issued from the generated plans. Z here can refer either to [LO] (the objective function applied locally) or [ME] (the mean energy per unit of time applied locally).

Firstly, we compared the results of the greedy algporithms with the ILP. For this, we considered the 30 instances of configurations #1 and #2. In figure C.10, we depict the ratio between the energy consumption of the plan computed by the greedy algorithms versus the one computed by the ILP. As one can notice, the

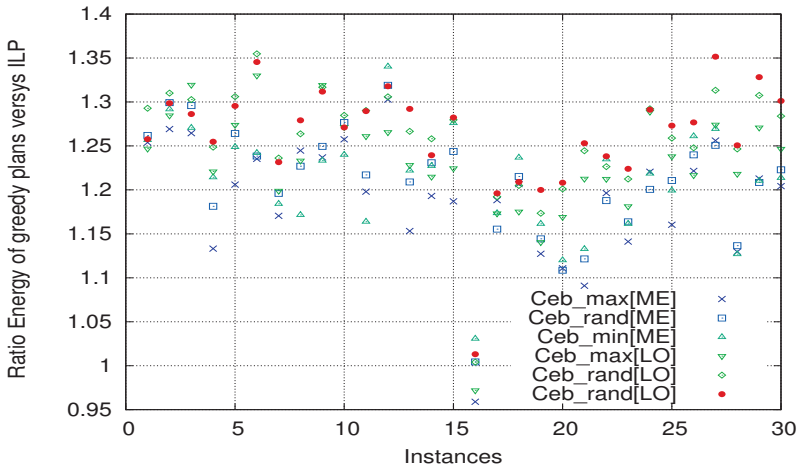


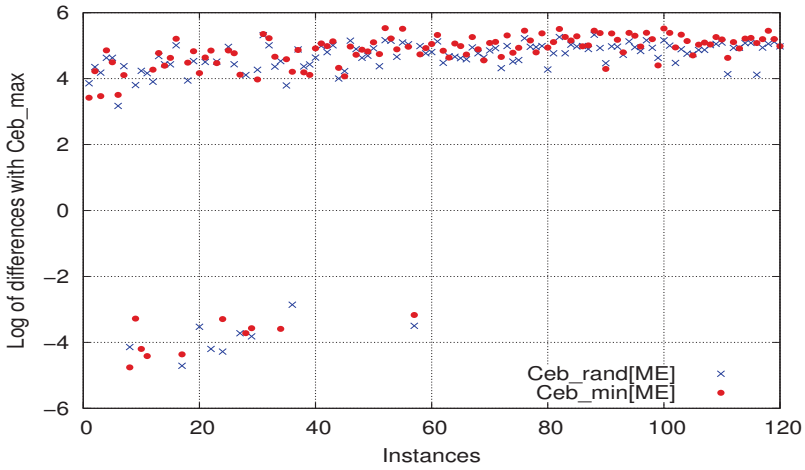
FIGURE C.10: Ratio of the energy plan of the greedy algorithms versus the one of the ILP

For the ILP model, we used the CPLEX solver with the configuration of the previous section. One can remark from these results that the solutions of the plan of the greedy algorithms where at most 1.4 times more energy-consuming than the one of the ILP. We even have situations where the plans generated by the greedy algorithms were better. But, this is due to the fact that we do not run the ILP until getting the optimal solution. In addition to these results, let us observe that while the ILP-based solutions were sometimes found after one hour, the greedy algorithms solved each of the 30 instances in less than one second.

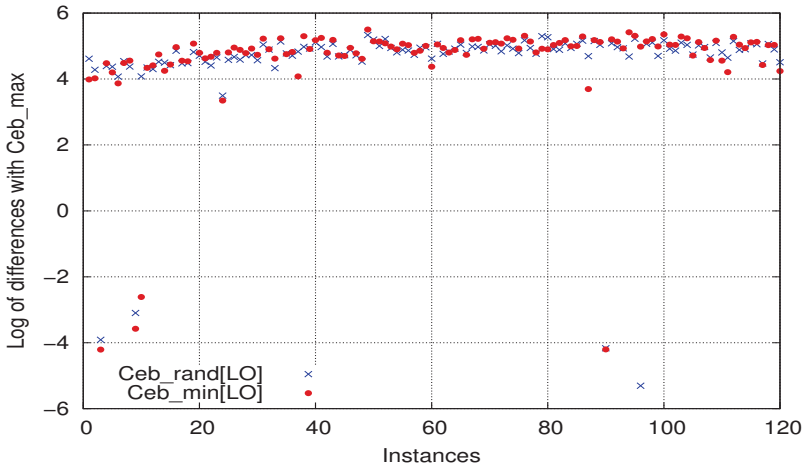
These experiments were only done on instances of the configuration #1 and #2. We also tried to compare the ILP and the greedy algorithms on other configurations. But, in many cases, it was not possible to obtain a solution or even a lower bound to the ILP model due to a lack of memory. Based on these elements, our conclusion is that the greedy algorithms can produce good solutions for the problem in a real-time setting. Moreover, they are more scalable when dealing with large instances of the problem.

The goal of our next investigation in the experiments was to state whether or not the greedy algorithms are all equivalent. On this point, our first result is that whatever objective function, the max ordering (or the heuristics *Ceb_max[Z]*) is the best instantiation of the general scheme of the greedy algorithms. In figure C.11, we depict the differences between the energy consumed in the plan computed by *Ceb_rand[Z]*, *Ceb_min[Z]* and the one computed by *Ceb_max[Z]*. Given an instance, let us denote

by *E_max*, *E_rand* and *E_min*, the consumption produced by *Ceb_max[Z]*, *Ceb_rand[Z]* and *Ceb_min[Z]*. The points in the figures represent the differences $\log(E\_rand - E\_max)$ and $\log(E\_min - E\_max)$ if these values are computable; $-\log(-E\_rand + E\_max)$ and $-\log(-E\_min + E\_max)$ otherwise. The results show that in more than 90% of cases, the best plan between the greedy heuristics comes from *Ceb_max[Z]*. Therefore, the max ordering dominates the others. In these results, we used the logarithmic of differences for the sake of presentation. Let us oberve that with a logarithmic of difference equal to 8, we have a difference in 10^8 in energy consumption.



(A) Mean Energy per time Unit

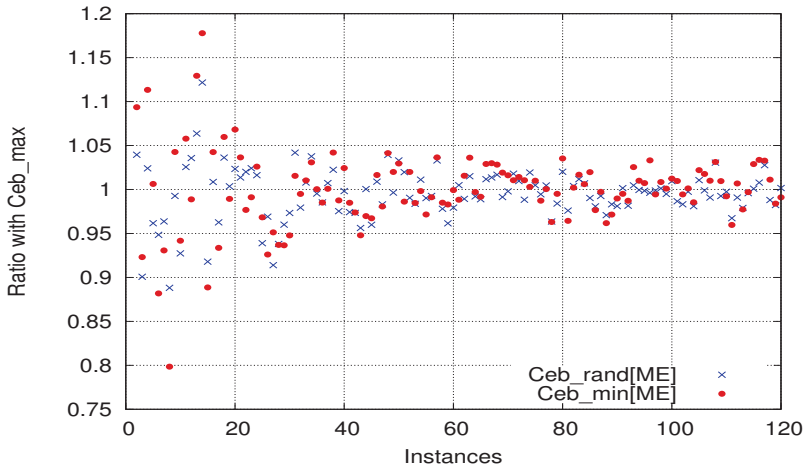


(B) Local Obj. Function

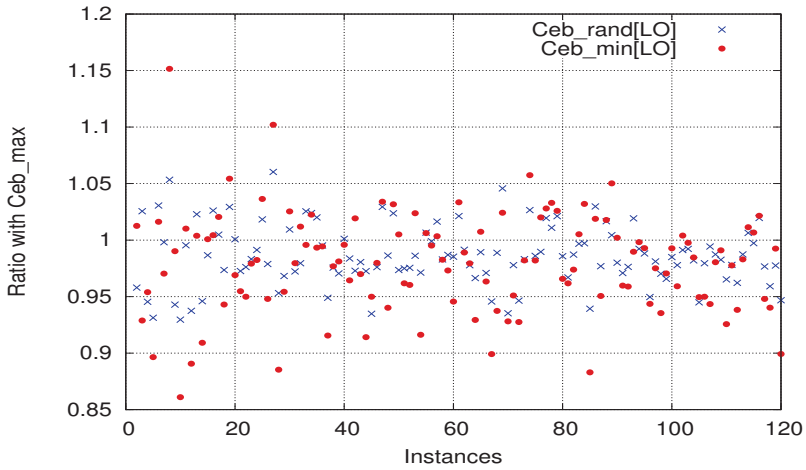
FIGURE C.11: Logarithmic differences of the energy consumption with *Ceb_max[Z]*

For understanding more the differences between the variants of our energy-aware algorithm, we also measured in the experiments the total transfer energy built issued from the solution of each heuristics. We then compared them in order to see if the good results of *Ceb_max[Z]* were due to the fact that it built plans with a lower consumption in transfers. The ratio between the transfer energy of *Ceb_rand[Z]*, *Ceb_min[Z]* versus

the one of *Ceb_max[Z]* are depicted in figure C.12. We cannot ensure from this figure that the domination of *Ceb_max[Z]* comes from the minimization of the transfer energy. As suggested by the variability in transfers, the base energy and the consumption of applications on machines are also important in the performance of *Ceb_max[Z]*.



(A) Mean Energy per time Unit



(B) Local Obj. Function

FIGURE C.12: Ratio of transfer energy with *Ceb_max[Z]*

At this stage we know that the *Ceb_max[Z]* heuristics dominate the other. But depending on the objective function (Z), what is the best heuristic? In figure C.13, we depict the ratio between the objective functions of heuristics issued from [LO] (the objective function applied locally) and [ME] (the mean energy per unit of time applied locally).

The results clearly demonstrate the superiority of [ME] heuristics as predicted in our prior discussion (See Section C.5.3.1). We also investigated on the impact of the transfers in these differences. In Figure C.14, we computed the prior ratios only on the transfer energy produced by the heuristics. The results clearly show that the transfer costs with [LO] are higher than with [ME]. This again confirms the analysis made in Section C.5.3.1.

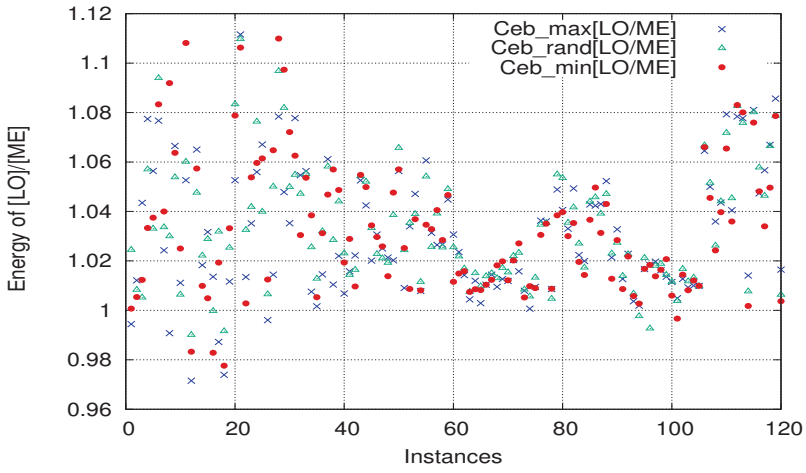


FIGURE C.13: Ratio of the energy plan of [LO] versus those of [ME]

Indeed, in the choices of Cebs, [ME] heuristics consider the period in which machines will be available. This helps for reducing the situations in which applications will be deployed on machines that are available only for a short period of times (what is not opttimized in [LO] heuristics).

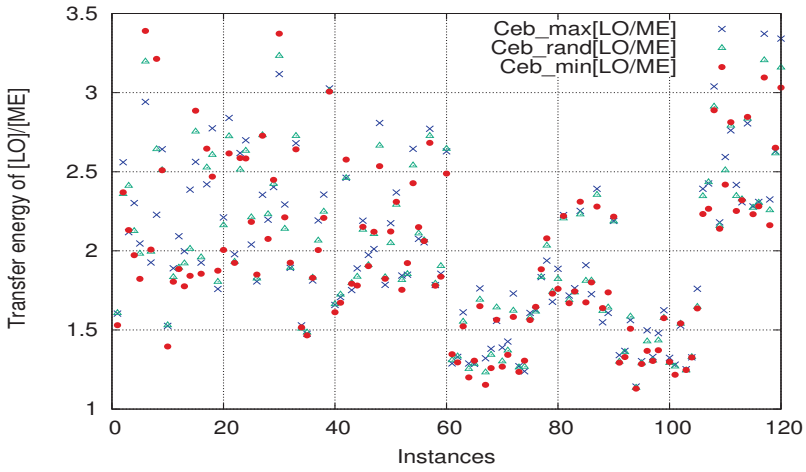


FIGURE C.14: Ratio of the transfer energy plan of [LO] versus those of [ME]

Finally, we also measured the runtimes of the greedy algorithms. Onn each instance, we observed few differences (less than one second). Moreover, the maximal runtime for processing and instance was equal to 15 seconds. With such a bound, the algorithms can be used in practice within a cloud scheduler.

Summarizing, our experiments globally brings three conclusions. The first is that the ILP model and the rolling heuristic can only be used on very short problem instances. Our experiments also gave insights on the sizes of the instances for which these approaches are irrelevant. Our second conclusion is that the greedy heuristics offer a good compromise (quality, runtimes). This means that for the green scheduler that we pursue in our study,

these heuristics are suitable. Finally, *Ceb_max[ME]* has the best behavior between the heuristics. This result was also expected from our analysis of the greedy algorithms.

C.7 Related work

The context of our work is related to volunteer computing and Desktop Grid Computing [Cérin and Fedak, 2012]. Desktop Grids (DG) have been successfully used for solving scientific applications at low cost. DGs middleware such as Condor [Butt et al., 2006], BOINC [Anderson, 2004], XtremWeb [Fedak et al., 2001], OurGrid [Andrade et al., 2003] provide researchers a wide range of high throughput computing systems by utilizing idle resources. However, we did not find in this context a work that aims at optimizing the energy consumption in a setting similar to the one considered in this paper.

In [Berl et al., 2010] authors introduce a synthesis of the usage of methods and technologies used for energy-efficient operation of computer hardware and network infrastructure. They consider the ICT field in general and they focus on energy-aware scheduling in multiprocessor and grid systems, on the power minimization in clusters of servers, on the power minimization in wireless and wired networks. Clouds are not specifically considered in their work.

In [Beloglazov and Buyya, 2010] authors propose an efficient resource power management policy for virtualized Cloud data centers. The objective is to continuously consolidate virtual machines. They show that the dynamic reallocation of virtual machines brings substantial energy savings. They propose four criteria for migrating the virtual machines. Authors do not discuss about the quality of the solution and they recognize that despite the fact that they use heuristics, the algorithms provide good experimental results. We believe that our solution can be extended for consolidating virtual machines in a volunteer context.

In [Beloglazov and Buyya, 2012] Beloglazov and Buyya propose online algorithms for virtual machines placement with guaranty of performance. They conduct competitive analysis and prove competitive ratios of optimal online deterministic algorithms for a single virtual machine migration and dynamic virtual consolidation problems. An interesting future work is to extend this result to volunteers clouds contexts.

Beaumont, Eyraud-Dubois and Larchevêque consider in [Beaumont et al., 2013] the problems of reliable service allocation in clouds. Among the different papers introduced in this section, it is a paper that cover similar problems than ours. They consider first that mapping virtual machines having heterogeneous computing demands onto physical

machines having heterogeneous capacities can be modeled as a multi-dimensional bin-packing problem. They assume that each virtual machine comes with its failure rate (i.e. the probability that the machine will fail during the next time period). But they do not consider that services should be duplicated on different machines to derive a robust solution.

In [Young et al., 2013] authors study the problem of energy-constrained dynamic allocation of tasks to a heterogeneous cluster computing environment. The fundamental difference of this work with our work is that we consider a multi-period task execution while they focus on a single one.

In [Barrondo et al., 2012] authors address knowledge-free Bag-of-tasks non preemptive scheduling problem on heterogeneous grids, where scheduling decisions are free from information of resources and application characteristics. They analyze the energy consumption of job allocation strategies based on variations of the replication threshold. We can view this work as studying the completion time of a job from which they derive the energy consumption. More precisely, they investigate the energetic performance of classical scheduling strategies according to the makespan metric. None of the analyzed algorithms is similar to what we propose. Our particularity is to propose an energy-aware scheduling algorithm which is not, in our cloud context, makespan-oriented. In [Borgetto et al., 2012] authors study the problem of energy-aware resource allocation for hosting long-term services or on-demand compute jobs in clusters. They do not capture the possibility of copies but they have a constraint on the RAM available on each node. The objectives and constraints lead to greedy algorithms. This work is similar to our work but the main differences are that for each job, allocated to a machine, we must have to decide the fraction of CPU to use and we need also for an estimate of the RAM consumption of the job. The fraction of CPU is for studying a form of heterogeneity but it does not include all the cases that can be derived from our heterogeneous modeling.

We do not model the commonly-used Dynamic Voltage and Frequency Scaling (DVFS) power management technique [Le Sueur and Heiser, 2010, Weiser et al., 1994] as it is now available on most processors including processors for smartphones and tablets. DVFS is able to reduce the power consumption of a CMOS integrated circuit, by scaling the frequency at which it operates, and when the load varies dynamically.

In [Cérin et al., 2013], authors propose models for optimizing the run of applications in our volunteer context. In the first model, they assume that they have a finite number of homogeneous volunteers nodes on which the applications can be run. Homogeneous volunteers means that the computing nodes used in the cloud platform have the same specification. This implies for instance that each node can run the same maximal number of applications and that each machine is assumed to have similar (or identical) power

consumption characteristics. The paper also considered heterogeneous machines. We propose in this paper a more general view of the problem compared to this work. In particular, the problem studied in [Cérin et al., 2013] is a variant of PPPVC$_{MinSum}$ restricted to one period ($|\mathcal{T}| = 1$). Because, we are interested in a more general problem, the ILP modeling and the heuristics proposed in this paper also differ from the one that are described in [Cérin et al., 2013]. Finally, in comparison to this work, we covered a larger set of examples in terms of variations of the availability matrix, capacity of machines, number of instances and energy consumptions costs.

C.8 Conclusion

In this paper, we studied the optimization of the energy consumed in the provisioning of services within a volunteer cloud. We provide a complete solution that includes: a modeling of the service provisioning problem, an analysis of its hardness, various resolution approaches and their experimental analysis. Our study showed that we can find good algorithms for the optimization of our service provisioning in real time settings.

For future work, our first objective is to implement the proposed algorithms in the SlapOS system. One main challenge for such an implementation will be the efficient implementation of service migrations. For this we can use the work of [Courteaud et al., 2012] related to SlapOS. We also plan to include a learning system, for computing the key parameters (base energy, overhead etc.) that are required in our model. Such a learning must be based on a monitoring system that collects data throughout the provisioning of applications. For this, we intend to modify the run of our applications in order to include code for energy measurement. The benchmark codes of Green Graph 500 [6] can serve here as reference. Finally, let us observe that network communications also consumes lots of energy. For instance, the cellular connectivity is one of the biggest contributors of energy consumption in a smartphone [Carroll and Heiser, 2010]. We do believe that our model must be affined for taking into account possible interactions between the users and the cloud in the service provisioning.

[6] http://green.graph500.org/code.php

Appendix D

MRLRP Appendix

D.1 Complete Symbol Table

In the following we recall all the symbols used in the paper, specifying for each one the meaning and the section the symbol has been introduced in.
Symbols are divided in sections D.1.1 to D.1.3 according to which part of the paper they refer to.

D.1.1 General notation

symbol	meaning	introduced in
G, V, E, A	graph and its node, edge and arc sets	section 2.3
$k \in K$	set of gates	
$u \in U$	set of UDCs	
$\mathcal{S}_U$	collection of subsets of U for subtour elimination constraints	
$l \in L$	set of SPLs	
$i \in P$	set of pick-up demands	
$i \in D$	set of delivery demands	
P_k, D_k	partitions of P and D according to the gate of each demand	
$(u, v) \in A_U$	set of ring arcs (first level)	
A'_U	set pairs of UDC, on which ring installation costs are defined	
E^p, E^d	pick-up and delivery (second level) edge sets	
q_i	quantity of goods associated to demand i	
k_i	gate associated to demand i	
F_u	fixed cost of UDC u	
Q_u	capacity of UDC u	
q_{uv}	capacity of ring arc (u, v)	
N	maximum number of UDC that can be opened (budget constraint)	
B	maximum number of UDC a gate can be linked to	
c_{uv}	per-flow-unit transportation cost of ring arc (u, v)	
c_{uk}, c_{ku}	per-flow-unit transportation costs between gate k and UDC u	
c_{ij}	routing cost on edge $(i, j) \in E$	
g_{uv}	ring installation cost, to connect UDC u and v in both directions	
$-\delta_h^-$, δ_h^+	fleet rebalancing bounds for generic route terminal point $h \in U \cup L$	
q	maximum load of second level vehicle	
M	maximum trip length of second level vehicle	
$r \in \mathscr{R}$	set of second level service routes	
R_r	sequence of demands associated to route r	
E_r	set of edges associated to route r	
$q(r)$	total load of route r	
$c(r)$	total routing cost of route r	
$q_k(r)$	load of route r according to the gate of the demands it serves	
$\mathscr{R}^p$, $\mathscr{R}^d$	sets of pick-up and delivery routes	
$\mathscr{R}_i$	routes that visit demand i	
$\mathscr{R}_h^+$	routes that start at UDC (or SPL) h	
$\mathscr{R}_h^-$	routes that end at UDC (or SPL) h	
$\mathscr{R}_u^{+d}$	delivery routes that start at UDC $u \in U$	
$\mathscr{R}_u^{-p}$	pick-up routes that end at UDC $u \in U$	

TABLE D.1: All the general symbols.

D.1.2 Decision variables of MILP formulation for MRLRP

symbol	meaning	introduced in
y_u	installation of UDC u	section 2.4.1
z_{uv}	connection of UDC u and v in both direction	
χ_{ku}	choice of linking gate k to UDC u	
x_r	choice of service route r	
φ_{ku}	flow of goods from gate k to UDC u	
φ_{uk}	flow of goods from UDC u to gate k	
φ_{uv}^{dk}	flow of delivery goods of commodity k through ring arc (u,v) (outflows)	
φ_{uv}^{pk}	flow of pick-up goods of commodity k through ring arc (u,v) (inflows)	
ϕ_{ku}	maximum between φ_{ku} and the quantity of commodity k that leaves u for delivery	
ϕ_{uk}	maximum between φ_{uk} and the quantity of commodity k that arrives at u from pick-up	
ζ_u	variables to strengthen subtour elimination constraints (first form)	section 2.5.2.1
$\hat{u}$	fictitious UDC to strengthen subtour elimination constraints (second form)	section 2.5.2.2
$\hat{U}$	extended set of UDCs	
$\hat{A}'_U$	extended set of ring arcs	
$\hat{D}(S)$	cut-set of UDC collection $S \in \mathcal{S}_U$ w.r.t. the extended set of ring arcs	
$z_{\hat{u}u}$	connection of fictitious UDC $\hat{u}$ and actual UDC u	

TABLE D.2: Symbols linked to MILP formulation for MRLRP.

D.1.3 GALW specific notation

symbol	meaning	introduced in
R	generic sequence of demands	section 2.6.1
$r(R)$	minimum cost route that descends from sequence R	
symbols specific to GALW route generator		
$\overline{\mathscr{R}}$	output subset of service routes	section 2.6.1
$\overline{\mathscr{R}}_i$	routes of $\overline{\mathscr{R}}$ that visit demand i	
β	reduction of maximum length of generated routes between two iterations	
τ	threshold to stop iterations of route generation	
$d(R,i)$	nearest neighbor function	
(p_q, p_M, p_ω)	vector of weights in function $d(R,i)$	
ω	minimum number of routes per demand at each iteration	
symbols specific to GALW assignment subproblem		
O	subset of chosen UDC	section 2.6.2
$F_{u,o}$	lower bound on ring construction costs	
ξ_o^u	additional decision variables	
b	number of arcs to be possibly added in the objective function	section 2.7
S	number of solutions to which the following stages of GALW are applied	
symbols specific to GALW ring multiflow subproblem		
O_{dk}^+	subset of opened UDC with availability of commodity k for delivery	section 2.6.4
O_{dk}^-	subset of opened UDC with request of commodity k for delivery	
O_{dk}^2	set of delivery source-sink pairs (delivery of commodity k)	
$\Phi_{uv}^{dk}, \Phi_{uv}^{pk}$	flow from source u to sink v (delivery, pick-up of commodity k)	
$\theta'_{\{uv\}}, \theta''_{\{uv\}}$	clockwise and counterclockwise paths between opened UDCs u and v	
$(\Phi_{uv}^{dk})'$, $(\Phi_{uv}^{dm})''$	clockwise and counterclockwise parts of Φ_{uv}^{dk}	

TABLE D.3: Specific symbols concerning GALW matheuristic.

D.2 Generation of MRLRP instances from benchmark CLRP ones

As previously recalled, in [Prins et al., 2006], the benchmark names follow the pattern n-m-c[b], with n being the number of customers, m the number of depots and c the number of clusters into which customers are grouped; the final 'b' denotes $q = 150$, otherwise $q = 70$. The demands of the n customers are integer and are uniformly distributed in [11,20], while the depot capacities ensure at least 2 depots are opened. As in the original paper, the second-level route costs are obtained starting from Euclidean distances multiplied by 100 and rounded up to the nearest integer. MRLRP features have been added as follows. We will denote as $i = i_{\text{rnd}}(i_a, i_b)$ the function to pick a random value out of a range of integers.

D.2.1 Pollution indicators

To estimate the pollution indicators of the ecological instance of a scenario, we supposed to deal with real vehicles so as to consider realistic values. For the second level, we considered vehicles with a useful load of 1 T and a pollution amount of 5 grCO2/km (grammes of CO2 per km). For the first level, we supposed that gates-UDCs and UDC-UDC transportation are performed by vehicles with a pollution emission of respectively 60 and 7.5 grCO2/T $\cdot$ km (grCO2 per tonne per km): the latter value is lower as we suppose that ring transportation is performed with less polluting vehicles, like for instance trains, and that they are fully loaded, unlike long-haul trucks going from gates to UDCs. The value of all parameters has been chosen according to the report [Boulter and Barlow, 2005]. The useful load of second level vehicles suggested us a reference conversion factor Q^{-1} from tonnes to load unit of our problem; note that conversion from km to length units is not necessary, since it would simply scale every cost parameter of the problem by the same factor. Therefore, the environmental cost of a second level service path is obtained by multiplying its length by a factor of 5 grCO2 per length unit, whereas for first level, given a gate k and two UDCs u and v, environmental costs c_{ku}, c_{uk}, c_{uv} and c_{vu} are expressed in grCO2 per load unit per length unit, and are obtained by multiplying constants 60 Q^{-1} and 7.5 Q^{-1} by the respective distances.

D.2.2 UDCs and demands

UDCs and demands are generated from the original benchmark: nevertheless, some processing is needed to adapt the original benchmark graph to the MRLRP features.

D.2.2.1 Location of points

UDC are initially located according to the depot points of the original benchmark instance and given the same capacities and fixed costs. Then, to create the *city context* that is peculiar to the MRLRP, we need the UDCs to be located in such a way as to surround all, or at least the great majority, of the customers, i.e. the demands' locations. To achieve this, we first take the convex hull H_0 of depot and customer points in the original benchmark; then, we eventually exchange some depot and clients locations,in order to have all the depots located on vertices of H_0; finally, we shift the depots along the perimeter of H_0 to maximize the area of the convex polygon given by the depots only.

D.2.2.2 Demand sets

In most of the instances, the original CLRP customers give rise directly to the MRLRP demands D and P: the former are randomly inserted into the latter in such a way as to have $|D| = |P|$. However, an augmentation of the demands w.r.t. the initial (CLRP) ones may be necessary. To achieve this, we generate a *second demand* for some of the existing points in the graph – instead of generating new points. The second demand has quantity $q = i_{\text{rnd}}(11, 20)$, i.e. like those in the original benchmark, and may be of the same type (delivery, pick-up) or not. However, new demands are generated in such a way as to preserve $|D| = |P|$. Since the overall demand is greater than that of the original CLRP instance, which we denote by Q_{tot}, we increase the capacity of each UDC by a factor $\frac{\sum_{i \in D \cup P} q_i}{Q_{tot}}$ and finally round it up; then, we do the same for the fixed costs, to preserve the cost/capacity ratio, and for the Euclidean distance multiplier (whose initial value is 100) for the same reason.

D.2.2.3 Additional UDCs

We generate possible additional UDCs (as required for scenario 2 of each collection) in the ring between H_0 and the polygon that is similar to H_0, has the same centroid and an area that is 85% of that of H_0. By doing so, the newly generated points are in the peripheral region of the *city*. The capacity of the new UDCs is generated as $i_{\text{rnd}}(\lceil 0.8 \cdot Q_U^{avg} \rceil, \lfloor Q_U^{avg} \rfloor)$, Q_U^{avg} being the average capacity of the initial UDCs, whereas the fixed cost is obtained by multiplying the capacity by the average cost/capacity ratio of the initial UDCs.

D.2.2.4 Tightness of the UDC capacities and minimal number of UDCs needed

Once the UDCs and demands have been obtained in this way, we need to ensure that $\max_{u,v \in U, u \neq v} (Q_u + Q_v) < \sum_{i \in D \cup P} q_i \leq \frac{1}{f_q} \sum_{u \in U} Q_u$. The left inequality ensures that at least three UDCs will be needed, whereas the right one prevents an instance from being too tight, as $f_q \cdot \sum_{i \in D \cup P} q_i$ is an estimation of the total occupation of the UDC capacity due to the demand allocation on both the first and the second level; as for f_q, its value must be comprised in $[1, 2]$, with $f_q = 1$ and $f_q = 2$ representing the two extreme cases of, respectively, all direct and all indirect shipments; we always choose $f_q = 2$. If necessary, we alternate two steps as long as required for both the above inequalities to be verified: a) we iteratively increment each demand by 1 load unit until the left inequality is verified; b) we iteratively increment by 1 the capacity of each UDC but the greatest one, without exceeding it, until the right inequality is satisfied.

D.2.3 Gates and SPLs

Table D.4 shows how many gates and SPLs have been generated, according to the values of n and m in the original CLRP benchmark instance. The value between parentheses

n	m	$\|K\|$	$\|L\|$
20	5	5(10)	5(10)
50	5	5(10)	5(10)
100	5	5(10)	10(15)
100	10	5(10)	10(15)

TABLE D.4: Number of gates and SPL in the generated MRLRP instances.

is the maximum value, i.e. the one in the corresponding scenario of a collection that considers an augmented number of gates or SPLs. To locate both gates and SPLs, we take the convex hull H_0 and we enlarge it by an extension factor $f > 1$ to obtain the polygon $H(f)$ according to the extension principle shown in figure D.1; n_{H_0}, p_{H_0} and A_{H_0} are the number of vertices, the perimeter and the area of H_0, respectively. Then, the SPL positions are randomly generated inside $H(1.1)$ in such a way as to have a distance greater than or equal to $0.3\frac{p_{H_0}}{n_{H_0}}$ from any UDCs and any other SPLs, while the gates' positions are randomly generated in the ring between $H(1.2)$ and $H(1.6)$ in such a way as to be uniformly distributed all around H_0.

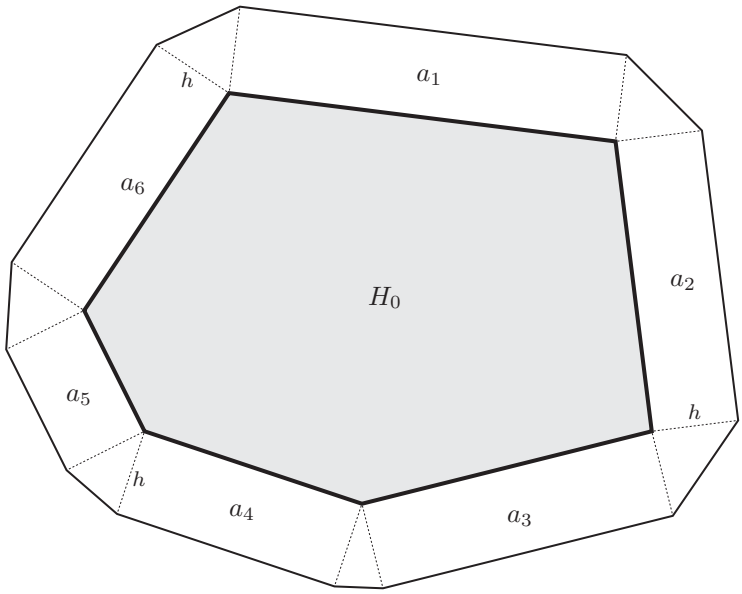


FIGURE D.1: Polygon $H(f)$ obtained by extension of the convex polygon H_0 according to the extension factor $f > 1$: height h is chosen so to have $\sum_i a_i = h \cdot p_{H_0}$ equal to $(f-1) \cdot A_{H_0}$.

D.2.4 Second-level vehicles' features

For q the value of the original benchmark instance is taken, whereas for M we choose $M = 50$ when $q = 70$, and $M = 65$ when $q = 150$, except for collection galwc06, in which $q = 70$ and $M = 60$.

D.2.5 Budget constraint and maximum number of UDCs a gate can address

The budget constraint is always relaxed for economic instances by fixing $N = |U|$, as it has already been said in 2.7.1, whereas for green instances we impose $N = 4$, 6 or 10 depending on whether $|U| = 5$, 10 or 15. The constant B has always been given the value 2.

D.2.6 Demands and their assignment to gates

We randomly choose $2|K|$ demands in $D \cup P$ to assure that each gate is assigned at least two demands: the remaining $|D| + |P| - 2|K|$ are randomly assigned. When a second demand needs to be generated for a customer point in the graph, and has the same type as the first one, it is guaranteed not to be associated with the same gate as the first one.

D.2.7 First-level arcs' features

The flow transportation costs c_{ku} and c_{uk} are both obtained by multiplying the distance between gate k and UDC u by $c^{\star}_{PU} = 2.5$, whereas for c_{uv}, $u, v \in U$, we multiply the distance between u and v by $c^{\star}_{ring} = 0.5$; both constants $c^{\star}_{PU}$ and $c^{\star}_{ring}$ are costs per length unit and per load unit. Regarding ring construction costs, we suppose that there is a subset $U^{\star} \subset U$ of UDCs which are more expensive to connect in the ring: UDCs in $U^{\star}$ are picked randomly, and $|U^{\star}| = 1$ or $|U^{\star}| = 2$, depending on whether the base scenario has $|U| = 5$ or $|U| = 10$. Cost g_{uv}, $u, v \in U, u < v$ is obtained by multiplying the distance between u and v by the cost per length unit $g^{\star}_{uv}$, with $g^{\star}_{uv} = 50$ if neither of u and v is in $U^{\star}$, or $g^{\star}_{uv} = i_{\text{rnd}}(50, 100)$ otherwise. If needed, the ring construction costs generated in this way are adjusted to guarantee that the triangular inequality is always verified. Finally, capacity q_{uv}, $u, v \in U$, is obtained:

$$q_{uv} = \max\left(Q_u,\ \left(\tfrac{1}{2} \cdot i_{\text{rnd}}(\lceil 0.8 \cdot Q_u \rceil, \lfloor 1.1 \cdot Q_u \rfloor)\ +\ \tfrac{1}{2} \cdot i_{\text{rnd}}(\lceil 0.8 \cdot \sum_{i \in D \cup P} q_i \rceil, \lfloor 1.1 \cdot \sum_{i \in D \cup P} q_i \rfloor)\right)\right)$$

Costs c_{uv} and g_{uv} described so far represent the low levels of ring transportation and installation costs, i.e. the ones of economic instance of type L|L; for the high level of both types of costs, we multiply the corresponding low-level cost by 5.

D.2.8 Fleet balance bounds

For each $u \in U$, we estimate the number of vehicles needed as

$$\nu_u = \lceil \sum_{i \in D \cup P} q_i \cdot \frac{Q_u}{\sum_{v \in U} Q_v} \frac{|U|}{N} \frac{1}{q} \rceil + 1$$

to take into account both the budget constraint and the capacity q of second-level vehicles; then, we randomly choose both δ^-_u and δ^+_u as $i_{\text{rnd}}(0, \lfloor \frac{\nu_u}{2} \rfloor)$; if $\lfloor \frac{\nu_u}{2} \rfloor = 0$, we randomly choose $\{\delta^-_u, \delta^+_u\}$ in $\{\{1,0\},\{0,1\},\{1,1\}\}$. For each SPL l, we randomly choose both δ^-_l and δ^+_l as $i_{\text{rnd}}(0, \max_{u \in U} \lfloor \frac{\nu_u}{2} \rfloor)$; if $\max_{u \in U} \lfloor \frac{\nu_u}{2} \rfloor = 0$, we randomly choose also $\{\delta^-_l, \delta^+_l\}$ in $\{\{1,0\},\{0,1\},\{1,1\}\}$.

D.3 Complete Results

In the following the reader will find all the results of the experimental sessions described in section 2.7.2.

Instance	Features							Exact(X)				GALW							Hybrid(Y)	
	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	$\%^r$	r	$T/\%^z$	#ps	$\%^X$	t	r	a	T	#ps	$\%^Y$	$\%^X$	T
galwc01-0-L|L	5	5	5	15	5	70	50	3.5	9.6	24.8	21503	2.1	0.0	0.4	2.9	3.6	3122	0.0	2.1	3.6
galwc01-0-L|H	5	5	5	15	5	70	50	5.0	9.5	30.8	21503	4.9	0.0	0.4	4.2	4.9	3121	0.6	4.3	4.9
galwc01-0-H|L	5	5	5	15	5	70	50	6.5	9.6	16.3	21503	6.2	0.0	0.4	3.0	3.7	2926	4.6	1.7	2.4
galwc01-0-H|H	5	5	5	15	5	70	50	9.6	9.6	15.4	21503	9.3	0.0	0.4	7.4	8.1	3021	5.8	3.7	2.9
galwc01-1-L|L	5	5	5	20	5	70	50	3.3	14.0	48.7	37839	3.3	0.0	0.7	4.2	5.2	4033	0.1	3.2	4.8
galwc01-1-L|H	5	5	5	20	5	70	50	5.9	14.2	169.9	37839	4.8	0.0	0.7	8.9	9.8	4039	0.5	4.3	9.4
galwc01-1-H|L	5	5	5	20	5	70	50	7.2	14.1	80.7	37839	2.8	0.0	0.7	4.3	5.3	4140	0.0	2.8	5.0
galwc01-1-H|H	5	5	5	20	5	70	50	10.3	14.0	109.7	37839	5.0	0.0	0.7	3.8	4.8	4223	1.3	3.7	6.1
galwc01-2-L|L	5	10	5	15	10	70	50	4.5	24.9	642.5	64737	6.2	1.1	0.4	28.5	29.2	8388	2.4	3.9	37.3
galwc01-2-L|H	5	10	5	15	10	70	50	7.4	25.4	442.0	64737	5.1	1.1	0.4	31.0	31.7	8489	0.5	4.7	39.8
galwc01-2-H|L	5	10	5	15	10	70	50	9.0	25.4	320.5	64737	5.0	1.1	0.4	39.4	40.1	8630	0.3	4.7	33.6
galwc01-2-H|H	5	10	5	15	10	70	50	12.3	25.2	419.1	64737	8.3	1.3	0.4	64.2	65.0	8665	3.8	4.7	30.7
galwc01-3-L|L	10	5	5	15	5	70	50	4.7	9.6	16.8	21503	3.8	0.0	0.4	4.0	4.7	3068	2.5	1.4	2.3
galwc01-3-L|H	10	5	5	15	5	70	50	7.6	9.5	24.9	21503	3.8	0.0	0.4	2.8	3.5	3025	2.4	1.5	3.6
galwc01-3-H|L	10	5	5	15	5	70	50	8.3	9.6	10.2	21503	4.8	0.0	0.4	2.8	3.6	3067	3.7	1.1	1.8
galwc01-3-H|H	10	5	5	15	5	70	50	12.2	9.6	20.3	21503	5.9	0.0	0.4	3.5	4.2	3143	4.7	1.2	2.5
galwc01-4-L|L	5	5	10	15	5	70	50	4.4	13.6	31.0	31503	2.5	0.0	0.4	4.2	5.0	4348	0.0	2.5	2.7
galwc01-4-L|H	5	5	10	15	5	70	50	8.6	13.7	69.8	31503	4.6	0.0	0.4	6.0	6.7	4309	0.9	3.7	3.6
galwc01-4-H|L	5	5	10	15	5	70	50	8.9	13.7	29.3	31503	4.0	0.0	0.4	5.8	6.5	4313	2.2	1.9	2.7
galwc01-4-H|H	5	5	10	15	5	70	50	13.0	14.1	53.4	31503	5.8	0.0	0.4	5.1	5.8	4158	2.4	3.5	3.2
galwc01-0-G	5	5	5	15	4	70	50	1.1	10.1	8.2	21503	8.1	0.0	0.4	1.9	2.5	3072	2.4	5.8	2.8
galwc01-1-G	5	5	5	20	4	70	50	1.4	14.8	22.5	37839	6.3	0.0	0.7	3.0	3.9	4094	0.3	6.0	4.0
galwc01-2-G	5	10	5	15	6	70	50	1.8	26.1	86.6	64737	8.1	0.9	0.4	5.5	6.2	8242	2.5	5.8	41.5
galwc01-3-G	10	5	5	15	4	70	50	6.7	10.0	9.7	21503	6.1	0.0	0.4	2.1	2.9	3097	0.7	5.4	2.6
galwc01-4-G	5	5	10	15	4	70	50	0.7	14.2	9.2	31503	3.8	0.0	0.5	2.3	3.0	4284	0.5	3.3	2.2
galwc02-0-L|L	5	5	5	15	5	70	50	5.7	3.0	8.0	14066	1.4	0.0	0.3	1.6	2.2	2319	0.0	1.4	1.6
galwc02-0-L|H	5	5	5	15	5	70	50	10.0	3.0	10.3	14066	5.9	0.0	0.3	2.3	3.0	2418	1.3	4.6	2.0
galwc02-0-H|L	5	5	5	15	5	70	50	5.8	3.0	9.5	14066	1.7	0.0	0.3	2.3	2.9	2352	0.0	1.7	1.9
galwc02-0-H|H	5	5	5	15	5	70	50	7.4	3.0	4.0	14066	4.7	0.0	0.3	2.4	3.1	2421	0.2	4.5	1.7
galwc02-1-L|L	5	5	5	20	5	70	50	6.0	9.0	13.3	33616	0.9	0.0	0.6	2.4	3.4	3253	0.1	0.8	1.9
galwc02-1-L|H	5	5	5	20	5	70	50	10.4	9.2	189.8	33616	2.5	0.0	0.7	2.4	3.4	3303	1.2	1.3	3.2
galwc02-1-H|L	5	5	5	20	5	70	50	7.0	9.1	32.8	33616	3.1	0.0	0.7	2.2	3.2	3209	0.0	3.1	2.8
galwc02-1-H|H	5	5	5	20	5	70	50	9.0	9.1	430.5	33616	3.1	0.0	0.6	2.9	3.8	3195	0.5	2.6	3.9
galwc02-2-L|L	5	10	5	15	10	70	50	5.0	7.6	59.7	39438	3.6	1.1	0.3	4.7	5.3	4707	0.1	3.5	7.5
galwc02-2-L|H	5	10	5	15	10	70	50	8.7	7.7	78.7	39438	4.0	1.7	0.3	4.2	4.8	4667	2.5	1.5	11.0
galwc02-2-H|L	5	10	5	15	10	70	50	10.9	7.6	61.5	39438	0.2	1.1	0.3	5.5	6.1	4676	0.0	0.2	5.7
galwc02-2-H|H	5	10	5	15	10	70	50	14.6	7.6	80.8	39438	6.4	2.1	0.3	4.3	4.9	5012	2.8	3.8	17.7
galwc02-3-L|L	10	5	5	15	5	70	50	3.2	3.1	2.8	14066	0.3	0.0	0.3	2.1	2.7	2324	0.1	0.2	1.4
galwc02-3-L|H	10	5	5	15	5	70	50	9.6	3.1	8.7	14066	2.3	0.0	0.3	1.9	2.6	2347	2.1	0.2	1.7
galwc02-3-H|L	10	5	5	15	5	70	50	3.2	3.0	4.2	14066	1.8	0.0	0.3	2.3	3.0	2440	0.2	1.6	1.6
galwc02-3-H|H	10	5	5	15	5	70	50	7.7	3.0	4.8	14066	2.7	0.0	0.3	2.5	3.1	2322	1.1	1.6	1.9
galwc02-4-L|L	5	5	10	15	5	70	50	6.3	5.2	8.0	23190	0.2	0.0	0.3	2.4	3.1	3478	0.0	0.2	2.0
galwc02-4-L|H	5	5	10	15	5	70	50	12.3	5.2	18.0	23190	3.2	0.0	0.3	2.2	2.8	3419	1.0	2.3	2.1
galwc02-4-H|L	5	5	10	15	5	70	50	6.1	5.2	6.4	23190	1.6	0.0	0.3	3.6	4.3	3547	0.0	1.6	2.5
galwc02-4-H|H	5	5	10	15	5	70	50	7.9	5.2	12.6	23190	4.6	0.0	0.3	2.5	3.2	3375	0.4	4.2	2.6
galwc02-0-G	5	5	5	15	4	70	50	1.7	3.2	4.2	14066	8.5	0.0	0.3	1.5	2.1	2427	2.0	6.6	2.2
galwc02-1-G	5	5	5	20	4	70	50	1.4	9.6	13.1	33616	15.0	0.0	0.6	1.7	2.6	3243	1.0	14.1	2.7
galwc02-2-G	5	10	5	15	6	70	50	2.1	8.1	32.1	39438	8.3	0.9	0.3	3.2	3.8	4919	4.6	3.9	9.7
galwc02-3-G	10	5	5	15	4	70	50	1.8	3.3	5.2	14066	5.8	0.0	0.3	1.3	1.9	2303	1.3	4.6	1.6
galwc02-4-G	5	5	10	15	4	70	50	1.7	5.4	9.9	23190	6.4	0.0	0.3	1.9	2.5	3392	0.0	6.4	2.4
galwc03-0-L|L	5	5	5	15	5	150	65	18.6	6.6	10.8	18761	2.1	0.0	0.5	1.9	2.7	2082	0.0	2.1	2.2
galwc03-0-L|H	5	5	5	15	5	150	65	16.6	6.6	23.6	18761	6.3	0.0	0.5	4.9	5.7	2099	1.5	4.8	2.5
galwc03-0-H|L	5	5	5	15	5	150	65	26.5	6.6	20.2	18761	6.6	0.0	0.5	1.9	2.7	2012	1.2	5.4	1.8
galwc03-0-H|H	5	5	5	15	5	150	65	25.3	6.6	23.5	18761	6.9	0.0	0.5	5.0	5.8	2129	2.4	4.6	2.2
galwc03-1-L|L	5	5	5	20	5	150	65	16.0	51.3	113.8	68987	3.7	0.0	1.2	2.5	4.0	3513	1.5	2.2	3.8
galwc03-1-L|H	5	5	5	20	5	150	65	18.3	51.8	149.9	68987	7.4	0.0	1.1	9.2	10.6	3488	3.4	4.1	4.1
galwc03-1-H|L	5	5	5	20	5	150	65	23.9	51.8	131.6	68987	2.8	0.0	1.2	2.8	4.3	3476	0.0	2.8	3.7
galwc03-1-H|H	5	5	5	20	5	150	65	25.6	51.2	230.1	68987	7.0	0.0	1.1	4.2	5.6	3399	4.2	3.0	3.5
galwc03-2-L|L	5	10	5	15	10	150	65	19.2	24.1	181.9	79140	0.6	2.0	0.6	26.2	27.1	8449	0.0	0.6	13.7
galwc03-2-L|H	5	10	5	15	10	150	65	18.7	24.2	117.6	79140	0.8	1.1	0.6	25.7	26.6	8550	0.1	0.7	11.4
galwc03-2-H|L	5	10	5	15	10	150	65	20.9	24.2	151.1	79140	0.6	1.1	0.6	33.6	34.5	7849	0.0	0.6	11.5
galwc03-2-H|H	5	10	5	15	10	150	65	23.2	24.3	130.8	79140	5.6	1.7	0.6	39.6	40.5	8348	5.0	0.6	13.2
galwc03-3-L|L	10	5	5	15	5	150	65	18.6	6.7	11.3	18761	2.7	0.0	0.5	1.7	2.5	2085	0.0	2.7	1.8
galwc03-3-L|H	10	5	5	15	5	150	65	13.3	6.7	16.3	18761	4.3	0.0	0.5	1.9	2.7	2127	1.9	2.4	2.5
galwc03-3-H|L	10	5	5	15	5	150	65	24.1	6.6	14.2	18761	6.0	0.0	0.5	4.1	5.0	2183	3.9	2.3	1.9
galwc03-3-H|H	10	5	5	15	5	150	65	22.6	6.7	16.0	18761	4.5	0.0	0.5	3.7	4.5	2084	1.9	2.7	2.5
galwc03-4-L|L	5	5	10	15	5	150	65	18.1	11.0	22.7	31590	1.9	0.0	0.5	2.4	3.2	3319	0.0	1.9	2.7
galwc03-4-L|H	5	5	10	15	5	150	65	16.3	11.0	20.0	31590	3.9	0.0	0.5	6.2	7.0	3223	0.9	3.0	3.1
galwc03-4-H|L	5	5	10	15	5	150	65	26.7	11.0	29.4	31590	4.0	0.0	0.5	6.4	7.2	3287	1.4	2.7	2.3
galwc03-4-H|H	5	5	10	15	5	150	65	28.1	11.2	32.7	31590	3.6	0.0	0.5	9.5	10.3	3237	2.5	1.1	3.1
galwc03-0-G	5	5	5	15	4	150	65	1.6	6.9	3.6	18761	2.1	0.0	0.5	1.1	1.8	2124	0.0	2.1	1.4
galwc03-1-G	5	5	5	20	4	150	65	NS	54.2	NS	68987	$-\infty$	0.0	1.1	1.9	3.3	3388	0.0	$-\infty$	2.7
galwc03-2-G	5	10	5	15	6	150	65	2.6	25.1	56.2	79140	10.4	1.6	0.6	6.0	6.9	8834	6.1	4.6	17.2
galwc03-3-G	10	5	5	15	4	150	65	0.4	7.0	3.3	18761	2.6	0.0	0.5	1.1	2.0	2085	0.0	2.6	1.5
galwc03-4-G	5	5	10	15	4	150	65	0.5	11.8	5.2	31590	2.9	0.0	0.5	1.5	2.3	3111	0.5	2.4	1.9

TABLE D.5: Result of the three methods on all the instances of collections galwc01, galwc02 and galwc03.

	Features							GALW						Hybrid(Y)
Instance	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%Y$	T
galwc06-0-L|L	5	5	5	25	5	70	60	0.0	1.7	2.0	4.0	3180	3.0	3.9
galwc06-0-L|H	5	5	5	25	5	70	60	0.0	1.6	2.4	4.3	3107	8.7	5.5
galwc06-0-H|L	5	5	5	25	5	70	60	0.0	1.7	3.7	5.7	3135	0.1	4.0
galwc06-0-H|H	5	5	5	25	5	70	60	0.0	1.6	3.5	5.5	3092	2.5	5.5
galwc06-1-L|L	5	5	5	40	5	70	60	0.0	4.0	5.7	10.0	7376	2.0	8.3
galwc06-1-L|H	5	5	5	40	5	70	60	0.0	4.0	9.2	13.6	7395	6.4	12.5
galwc06-1-H|L	5	5	5	40	5	70	60	0.0	4.0	12.3	16.7	7558	3.4	13.1
galwc06-1-H|H	5	5	5	40	5	70	60	0.0	4.4	11.7	16.4	7580	1.1	15.4
galwc06-2-L|L	5	10	5	25	10	70	60	1.1	1.4	11.0	12.7	9428	1.4	33.2
galwc06-2-L|H	5	10	5	25	10	70	60	1.2	1.4	15.3	17.0	9679	4.6	29.0
galwc06-2-H|L	5	10	5	25	10	70	60	1.1	1.4	17.0	18.7	9389	0.2	33.7
galwc06-2-H|H	5	10	5	25	10	70	60	1.3	1.4	26.6	28.3	9359	4.7	24.3
galwc06-3-L|L	10	5	5	25	5	70	60	0.0	1.7	2.4	4.4	3155	1.1	4.1
galwc06-3-L|H	10	5	5	25	5	70	60	0.0	1.6	3.3	5.3	3134	8.6	5.7
galwc06-3-H|L	10	5	5	25	5	70	60	0.0	1.6	4.3	6.3	3111	0.0	4.2
galwc06-3-H|H	10	5	5	25	5	70	60	0.0	1.6	5.7	7.7	3165	1.6	5.7
galwc06-4-L|L	5	5	10	25	5	70	60	0.0	1.5	2.8	4.7	4448	0.6	4.2
galwc06-4-L|H	5	5	10	25	5	70	60	0.0	1.4	3.2	4.9	4260	8.0	6.1
galwc06-4-H|L	5	5	10	25	5	70	60	0.0	1.4	4.2	5.9	4401	0.0	3.7
galwc06-4-H|H	5	5	10	25	5	70	60	0.0	1.4	4.1	5.8	4472	1.1	5.0
galwc06-0-G	5	5	5	25	4	70	60	0.0	1.7	2.7	4.7	3148	1.4	5.4
galwc06-1-G	5	5	5	40	4	70	60	0.0	4.1	7.6	12.0	7537	1.5	14.3
galwc06-2-G	5	10	5	25	6	70	60	1.5	1.4	8.5	10.2	9396	4.2	22.2
galwc06-3-G	10	5	5	25	4	70	60	0.0	1.6	1.8	3.7	3116	1.3	3.7
galwc06-4-G	5	5	10	25	4	70	60	0.0	1.5	2.7	4.4	4410	1.6	4.3
galwc07-0-L|L	5	5	5	25	5	150	65	0.0	2.6	6.0	8.9	4791	0.7	7.4
galwc07-0-L|H	5	5	5	25	5	150	65	0.0	2.7	7.3	10.3	4880	0.1	7.1
galwc07-0-H|L	5	5	5	25	5	150	65	0.0	2.6	8.1	11.0	4813	0.2	7.8
galwc07-0-H|H	5	5	5	25	5	150	65	0.0	2.6	8.0	10.9	4839	0.1	9.6
galwc07-1-L|L	5	5	5	40	5	150	65	0.0	7.7	18.2	26.2	10902	0.3	23.1
galwc07-1-L|H	5	5	5	40	5	150	65	0.0	7.8	14.7	22.8	10769	0.7	27.3
galwc07-1-H|L	5	5	5	40	5	150	65	0.0	7.6	16.9	24.8	10658	0.2	18.8
galwc07-1-H|H	5	5	5	40	5	150	65	0.0	8.2	23.8	32.2	10861	1.5	31.7
galwc07-2-L|L	5	10	5	25	10	150	65	1.2	2.6	22.7	25.7	13286	3.0	31.1
galwc07-2-L|H	5	10	5	25	10	150	65	1.1	2.6	15.1	18.1	13426	2.1	21.6
galwc07-2-H|L	5	10	5	25	10	150	65	2.0	2.6	23.6	26.5	13463	3.8	64.0
galwc07-2-H|H	5	10	5	25	10	150	65	1.1	2.7	29.0	32.0	13753	4.1	37.8
galwc07-3-L|L	10	5	5	25	5	150	65	0.0	2.6	5.8	8.7	4752	0.0	8.9
galwc07-3-L|H	10	5	5	25	5	150	65	0.0	2.7	6.6	9.6	4886	1.8	10.6
galwc07-3-H|L	10	5	5	25	5	150	65	0.0	2.6	7.5	10.4	5037	0.0	9.0
galwc07-3-H|H	10	5	5	25	5	150	65	0.0	2.6	8.0	10.9	4861	2.1	12.4
galwc07-4-L|L	5	5	10	25	5	150	65	0.0	2.6	7.7	10.6	6936	0.0	9.0
galwc07-4-L|H	5	5	10	25	5	150	65	0.0	2.6	16.0	18.9	7048	3.1	11.5
galwc07-4-H|L	5	5	10	25	5	150	65	0.0	2.6	19.9	22.9	7175	0.4	9.9
galwc07-4-H|H	5	5	10	25	5	150	65	0.0	2.6	13.6	16.6	7354	4.1	9.7
galwc07-0-G	5	5	5	25	4	150	65	0.0	2.6	3.9	6.8	4714	1.3	7.3
galwc07-1-G	5	5	5	40	4	150	65	0.0	7.8	12.2	20.3	10733	0.9	19.2
galwc07-2-G	5	10	5	25	6	150	65	1.1	2.5	12.1	14.9	13666	1.5	38.8
galwc07-3-G	10	5	5	25	4	150	65	0.0	2.6	5.5	8.4	4731	0.0	10.3
galwc07-4-G	5	5	10	25	4	150	65	0.0	2.5	5.6	8.4	6975	0.5	7.5

TABLE D.6: Result of GALW and the hybrid method on all the instances of collections galwc06 and galwc07.

	Features							GALW						Hybrid(Y)
Instance	$\|K\|$	$\|U\|$	$\|L\|$	$\|D\|$	N	q	M	t	r	a	T	#ps	$\%Y$	$T/\%^z$
galwc08-0-L|L	5	5	10	50	5	70	50	0.0	9.2	11.2	20.7	6350	0.5	20.1
galwc08-0-L|H	5	5	10	50	5	70	50	0.0	9.4	6.3	16.1	6424	1.3	19.6
galwc08-0-H|L	5	5	10	50	5	70	50	0.0	9.7	8.9	19.0	6386	6.1	15.2
galwc08-0-H|H	5	5	10	50	5	70	50	0.0	9.5	8.1	18.0	6373	1.8	20.8
galwc08-1-L|L	5	5	10	80	5	70	50	0.0	28.1	31.1	59.7	16593	0.7	271.0
galwc08-1-L|H	5	5	10	80	5	70	50	0.0	26.6	29.6	56.8	16697	1.8	89.3
galwc08-1-H|L	5	5	10	80	5	70	50	0.0	27.1	28.9	56.6	16500	5.2	249.7
galwc08-1-H|H	5	5	10	80	5	70	50	0.0	27.1	76.2	103.8	16611	1.2	109.1
galwc08-2-L|L	5	10	10	50	10	70	50	1.6	7.3	56.5	64.2	15019	0.0	69.8
galwc08-2-L|H	5	10	10	50	10	70	50	1.1	7.0	43.0	50.4	14599	0.5	159.7
galwc08-2-H|L	5	10	10	50	10	70	50	1.9	7.1	104.6	112.1	14973	1.9	168.4
galwc08-2-H|H	5	10	10	50	10	70	50	1.1	7.2	54.4	62.0	14959	1.4	137.4
galwc08-3-L|L	10	5	10	50	5	70	50	0.0	9.6	11.6	21.6	6412	0.1	16.9
galwc08-3-L|H	10	5	10	50	5	70	50	0.0	9.8	11.3	21.5	6386	1.0	20.1
galwc08-3-H|L	10	5	10	50	5	70	50	0.0	10.2	11.4	22.0	6355	5.7	19.5
galwc08-3-H|H	10	5	10	50	5	70	50	0.0	9.2	6.7	16.4	6296	0.8	17.3
galwc08-4-L|L	5	5	15	50	5	70	50	0.0	9.0	13.2	22.7	9334	0.5	21.0
galwc08-4-L|H	5	5	15	50	5	70	50	0.0	9.3	13.7	23.4	9225	1.8	36.8
galwc08-4-H|L	5	5	15	50	5	70	50	0.0	8.8	14.1	23.4	9261	3.7	22.1
galwc08-4-H|H	5	5	15	50	5	70	50	0.0	9.0	22.3	31.7	9416	0.7	31.6
galwc08-0-G	5	5	10	50	4	70	50	0.0	9.4	5.8	15.6	6243	2.3	16.6
galwc08-1-G	5	5	10	80	4	70	50	0.0	27.3	22.1	49.9	16622	1.7	100.6
galwc08-2-G	5	10	10	50	6	70	50	1.0	7.0	24.6	32.0	14872	3.8	203.8
galwc08-3-G	10	5	10	50	4	70	50	0.0	9.4	6.8	16.5	6333	2.1	19.6
galwc08-4-G	5	5	15	50	4	70	50	0.0	9.1	10.0	19.5	9210	1.5	27.1
galwc10-0-L|L	5	5	10	50	5	70	50	0.0	7.7	17.1	25.2	10814	0.0	18.0
galwc10-0-L|H	5	5	10	50	5	70	50	0.0	7.8	16.0	24.1	10647	3.2	23.3
galwc10-0-H|L	5	5	10	50	5	70	50	0.0	7.7	26.5	34.6	10820	0.5	42.2
galwc10-0-H|H	5	5	10	50	5	70	50	0.0	7.9	15.2	23.5	11015	0.9	39.0
galwc10-1-L|L	5	5	10	80	5	70	50	0.0	24.0	106.7	131.2	28020	0.6	768.3
galwc10-1-L|H	5	5	10	80	5	70	50	0.0	23.9	262.9	287.2	27891	2.3	170.9
galwc10-1-H|L	5	5	10	80	5	70	50	0.0	25.9	71.7	98.1	28246	1.3	771.6
galwc10-1-H|H	5	5	10	80	5	70	50	0.0	25.4	64.3	90.3	28832	3.0	400.2
galwc10-2-L|L	5	10	10	50	10	70	50	1.4	6.8	242.9	250.1	26212	0.2	281.1
galwc10-2-L|H	5	10	10	50	10	70	50	1.1	7.0	261.3	268.7	25931	3.4	161.5
galwc10-2-H|L	5	10	10	50	10	70	50	1.5	6.6	463.3	470.2	26003	0.4	294.0
galwc10-2-H|H	5	10	10	50	10	70	50	1.5	6.7	837.9	845.0	25450	2.7	583.1
galwc10-3-L|L	10	5	10	50	5	70	50	0.0	8.2	31.0	39.7	10912	0.2	55.7
galwc10-3-L|H	10	5	10	50	5	70	50	0.0	7.9	17.1	25.5	11028	1.5	36.2
galwc10-3-H|L	10	5	10	50	5	70	50	0.0	7.7	19.1	27.2	10952	0.0	26.1
galwc10-3-H|H	10	5	10	50	5	70	50	0.0	7.8	15.8	24.0	10813	1.2	32.9
galwc10-4-L|L	5	5	15	50	5	70	50	0.0	7.8	19.5	27.7	13963	0.1	23.8
galwc10-4-L|H	5	5	15	50	5	70	50	0.0	7.6	34.6	42.6	13762	2.5	48.1
galwc10-4-H|L	5	5	15	50	5	70	50	0.0	7.6	20.0	28.0	13641	0.0	34.0
galwc10-4-H|H	5	5	15	50	5	70	50	0.0	7.9	56.7	65.0	14032	1.2	30.4
galwc10-0-G	5	5	10	50	4	70	50	0.0	7.8	14.9	23.1	10807	1.4	36.9
galwc10-1-G	5	5	10	80	4	70	50	0.0	23.6	56.5	80.6	27978	2.8	115.7
galwc10-2-G	5	10	10	50	6	70	50	0.9	6.8	41.2	48.3	26042	1.5	340.4
galwc10-3-G	10	5	10	50	4	70	50	0.0	7.9	12.7	20.9	10645	0.6	24.9
galwc10-4-G	5	5	15	50	4	70	50	0.0	7.8	16.9	25.0	13290	2.0	36.6
galwc11-0-L|L	5	10	10	50	10	70	50	1.4	7.9	466.4	474.8	14835	2.0	2681.4
galwc11-0-L|H	5	10	10	50	10	70	50	1.8	8.0	514.2	522.7	14663	3.1	(0.6%)
galwc11-0-H|L	5	10	10	50	10	70	50	1.1	8.1	414.7	423.1	14671	2.6	3211.2
galwc11-0-H|H	5	10	10	50	10	70	50	1.1	8.0	548.9	557.3	14532	2.6	1628.3
galwc11-1-L|L	5	10	10	80	10	70	50	1.1	21.6	476.7	499.0	36996	$-\infty$	$(+\infty)$
galwc11-1-L|H	5	10	10	80	10	70	50	1.1	22.3	(3.5%)	3636.1	36457	-8.0	(15.8%)
galwc11-1-H|L	5	10	10	80	10	70	50	1.4	21.7	(3.8%)	3635.5	36861	-4.3	(11.4%)
galwc11-1-H|H	5	10	10	80	10	70	50	1.6	22.7	(6.0%)	3651.0	37080	-8.3	(15.3%)
galwc11-2-L|L	5	15	10	50	15	70	50	69.7	8.1	612.8	621.3	27451	-18.6	(28.9%)
galwc11-2-L|H	5	15	10	50	15	70	50	70.2	8.3	1563.2	1571.9	27986	$-\infty$	$(+\infty)$
galwc11-2-H|L	5	15	10	50	15	70	50	68.7	8.2	2482.2	2490.9	27455	$-\infty$	$(+\infty)$
galwc11-2-H|H	5	15	10	50	15	70	50	68.4	8.3	3335.4	3344.2	27629	$-\infty$	$(+\infty)$
galwc11-3-L|L	10	10	10	50	10	70	50	1.1	8.3	834.0	842.8	14691	-0.1	(2.0%)
galwc11-3-L|H	10	10	10	50	10	70	50	1.1	8.0	236.5	245.0	14532	22.9	(3.6%)
galwc11-3-H|L	10	10	10	50	10	70	50	1.3	8.0	679.2	687.6	14513	2.7	(2.9%)
galwc11-3-H|H	10	10	10	50	10	70	50	1.1	8.0	474.8	483.4	14495	22.0	(6.7%)
galwc11-4-L|L	5	10	15	50	10	70	50	1.2	7.6	391.6	399.7	19571	5.8	879.2
galwc11-4-L|H	5	10	15	50	10	70	50	1.1	8.0	1240.5	1248.9	19519	0.9	(6.0%)
galwc11-4-H|L	5	10	15	50	10	70	50	1.1	7.5	381.2	389.2	20048	5.8	1705.0
galwc11-4-H|H	5	10	15	50	10	70	50	1.1	7.5	310.6	318.5	19612	4.1	(0.4%)
galwc11-0-G	5	10	10	50	6	70	50	1.4	7.9	90.3	98.7	14641	2.6	423.2
galwc11-1-G	5	10	10	80	6	70	50	0.9	22.4	66.5	89.4	37138	3.4	2983.0
galwc11-2-G	5	15	10	50	10	70	50	62.2	8.2	48.4	57.1	28059	3.0	(4.2%)
galwc11-3-G	10	10	10	50	6	70	50	0.9	7.8	52.1	60.4	14573	1.6	737.1
galwc11-4-G	5	10	15	50	6	70	50	1.7	7.8	72.2	80.5	19628	3.6	2058.4

TABLE D.7: Result of GALW and the hybrid method on all the instances of collections galwc08, galwc10 and galwc11.

Bibliography

H. Abbes, C. Cérin, and M. Jemni. BonjourGrid as a Decentralised Job Scheduler. In *APSCC*, pages 89–94. IEEE, 2008.

H. Abbes, C. Cérin, and M. Jemni. BonjourGrid: Orchestration of multi-instances of grid middlewares on institutional Desktop Grids. In *IPDPS*, pages 1–8. IEEE, 2009.

Agence de l'Environnement et de la Maîtrise de l'Energie. Les Centres de Distribution Urbaine : quels outils d'évaluation environnementale ? http://tinyurl.com/prpjckq, 2004.

Agence Nationale de la Recherche. Villes durables: Projet MODUM. http://snipurl.com/25d4v66, 2010.

A. Aguilera. Growth in Commuting Distances in French Polycentric Metropolitan Areas: Paris, Lyon and Marseille. *Urban Studies*, 42(9):1537–1547, 2005.

Z. Akça. *Integrated location, routing and scheduling problems: Models and algorithms.* PhD thesis, Lehigh University, 2010.

Z. Akça, R.T. Berger, and T.K. Ralphs. A Branch-and-Price Algorithm for Combined Location and Routing Problems Under Capacity Restrictions. In *Proceedings of the Eleventh INFORMS Computing Society Meeting*, pages 309–330, 2009.

D. Aksen and K. Altinkemer. A location-routing problem for the conversion to the "click-and-mortar" retailing: The static case. *European Journal of Operational Research*, 186(2):554–575, 2008.

S. Almoustafa, S. Hanafi, and N. Mladenovic. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394, 2013.

F. Alonso, M.J. Alvarez, and J.E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59(7):963–976, 2008.

A. Amaya, A. Langevin, and M. Trépanier. The Capacitated Arc Routing Problem with Refill Points. *Operations Research Letters*, 35(1):45–53, 2007.

D. Ambrosino and M.G. Scutellà. Distribution network design: New problems and related models. *European Journal of Operational Research*, 165(3):610–624, 2001.

D.P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In R. Buyya, editor, *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE Computer Society, 2004.

N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg. OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. In *Job Scheduling Strategies for Parallel Processing*, pages 61–86. Springer, 2003.

E. Angelelli and M.G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002a.

E. Angelelli and M.G. Speranza. The Application of a Vehicle Routing Model to a Waste-Collection Problem: Two Case Studies. *The Journal of the Operational Research Society*, 53(9):944–952, 2002b.

D. Applegate, R.E. Bixby, V. Chvátal, and W. Cook. Concorde TSP Solver. `http://www.tsp.gatech.edu/concorde.html`, 2006.

J. Araque, L. Hall, and T. Magnanti. Capacitated trees, capacitated routing, and associated polyhedra. CORE Discussion Papers 1990061, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 1990.

D. Arpin, G. Laporte, and Y. Nobert. *Optimal Solutions to Capacitated Multidepot Vehicle Routing Problems*. Cahiers du GÉRAD. École des hautes études commerciales, 1984.

G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., 1999.

G. Ausiello, V. Bonifaci, S. Leonardi, and A. Marchetti-Spaccamela. *Prize Collecting Traveling Salesman and related problems*, volume Handbook of Approximation Algorithms and Metaheuristics, pages 40.1–40.13. Gonzales Ed., 2007.

N. Azi. *Méthodes Exactes et Heuristiques pour le Problème de Tournées avec Fenêtres de Temps et Réutilisation de Véhicules*. PhD thesis, Université de Montréal, 2010.

R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2):347–380, 2009.

R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

R. Baldacci, A. Mingozzi, and R. Roberti. New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. *Operations Research*, 59(5):1269–1283, 2011a.

R. Baldacci, A. Mingozzi, and R. Wolfler Calvo. An Exact Method for the Capacitated Location-Routing Problem. *Operations Research*, 59(5):1284–1296, 2011b.

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

J.F. Bard, L. Huang, M. Dror, and P. Jaillet. A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30(9):821–834, 1998.

C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329, 1996.

S.S. Barreto, C. Ferreira, J. Paixão, and B. Sousa Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3):968–977, 2007.

A. Barrondo, A. Tchernykh, E. Schaeffer, and J.E. Pecero. Energy efficiency of knowledge-free scheduling in Peer-to-Peer Desktop Grids. In *HPCS*, pages 105–111, 2012.

O. Beaumont, L. Eyraud-Dubois, and H. Larchevêque. Reliable Service Allocation in Clouds. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 55–66. IEEE Computer Society, 2013.

J.M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler Calvo. A Branch-and-Cut method for the Capacitated Location-Routing Problem. *Computers & Operations Research*, 38(6):931–941, 2011.

A. Beloglazov and R. Buyya. Energy Efficient Allocation of Virtual Machines in Cloud Data Centers. In *CCGRID*, pages 577–578. IEEE, 2010.

A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

A.M. Benjamin and J.E. Beasley. Metaheuristics for the Waste Collection Vehicle Routing Problem with Time Windows, Driver Rest Period and Multiple Disposal Facilities. *Computers & Operations Research*, 37(12):2270–2280, 2010.

R.T. Berger. *Location-Routing Models for Distribution System Design.* PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, 1997.

R.T. Berger, C.R. Coullard, and M.S. Daskin. Location-Routing Problems with Distance Constraints. *Transportation Science*, 41(1):29–43, 2007.

A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, D.M. Quan, and K. Pentikousis. Energy-Efficient Cloud Computing. *The Computer Journal*, 53(7):1045–1051, 2010.

L. Bertazzi and M.G. Speranza. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1(4):307–326, 2012.

M. Boccia, T. G. Crainic, A. Sforza, and C. Sterle. A Metaheuristic for a Two Echelon Location-Routing Problem. In *SEA*, pages 288–301, 2010.

M. Boccia, T.G. Crainic, A. Sforza, and C. Sterle. Location-Routing Models for Designing a Two-Echelon Freight Distribution System. Technical Report 06, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, Canada, 2011.

D. Borgetto, H. Casanova, G. Da Costa, and J.-M. Pierson. Energy-aware service allocation. *Future Generation Computer Systems*, 28(5):769–779, 2012.

P.G. Boulter and T.J. Barlow. ARTEMIS: Average Speed Emission Functions for Heavy-duty Road Vehicles. TRL Unpublished Project Report UPR/IEA/12/05. TRL Limited, Wokingham, 2005.

J.C.S. Brandão and A. Mercer. The Multi-Trip Vehicle Routing Problem. *The Journal of the Operational Research Society*, 49(8):799–805, 1998.

W. Browne and J. Allen. The impact of sustainability policies on urban freight transport and logistics systems. In *World Transport Research, Vol 1: Transport Modes and Systems Oxford*, pages 505–518. Elsevier, 1999.

L.I. Burke and D. Tuzun. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

A.R. Butt, R. Zhang, and Y.C. Hu. A self-organizing flock of condors. *Journal of Parallel and Distributed Computing*, 66(1):145–161, 2006.

Canberra Bureau of Transport Economics. Urban congestion – The implications for greenhouse gas emissions. Technical Report BTE Information Sheet 16, Canberra Bureau of Transport Economics, 2001.

Capgemini. 2016 Future Supply Chain. http://tinyurl.com/nyrv2br, 2008.

A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Annual Technical Conference*, pages 1–12, 2010.

D. Cattaruzza, N. Absi, D. Feillet, and J. Gonzalez-Feliu. Vehicle Routing for City Logistics. Technical Report 3, Ecole des Mines de Saint-Etienne - CMP-SFL, 2013.

D. Cattaruzza, N. Absi, and D. Feillet. The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates. Technical Report 1, École des Mines de Saint-Étienne, CMP Georges Charpak, 2014a.

D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the Multi Trip Vehicle Routing Problem. *European Journal of Operational Research*, 236(3): 833–848, 2014b.

C. Cérin and G. Fedak. *Desktop Grid Computing*. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series, 2012.

C. Cérin, P. Gianessi, Y. Ngoko, C. Jiang, and J. Wan. Modeling Energy Savings in Volunteers Clouds. In *International Conference on Cloud Computing and Big Data (CloudCom-Asia), Fuzhou, China, December 16-18*, pages 52–59, 2013.

I.M. Chao, B.L. Golden, and E. Wasil. A New Heuristic for the Multi-depot Vehicle Routing Problem That Improves Upon Best-known Solutions. *American Journal of Mathematical and Management Science*, 13(3-4):371–406, 1993.

R. Chepesiuk. Decibel hell: the effects of living in a noisy world. *Environmental Health Perspectives*, 113(1):A34–A41, 2005.

S. Chiaravalloti, L. Budzisz, and F. Idzikowski. Power consumption of WLAN network elements. Technical Report TKN-11-002, Telecommunication Networks Group, Technical University Berlin, 2011.

N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, 1981.

S. Coene, A. Arnout, and F.C.R. Spieksma. On a periodic vehicle routing problem. *The Journal of the Operational Research Society*, 61(12):1719–1728, 2010.

Commission of the European Communities. Eurostat. http://epp.eurostat.ec.europa.eu, 2003.

Commission of the European Communities. Freight Transport Logistics in Europe – the key to sustainable mobility. Technical Report COM(2006) 336 final, Commission of the European Communities, 2006.

Commission of the European Communities. Sustainable Urban Transport Plans - Preparatory Document in relation to the follow-up of the Thematic Strategy on the Urban Environment. http://tinyurl.com/q8gvg6j, 2007.

Commission of the European Communities. A sustainable future for transport: Towards an integrated, technology-led and user friendly system. Technical Report COM(2009) 279, Commission of the European Communities, 2009a.

Commission of the European Communities. Action Plan on Urban Mobility. Technical Report COM(2009) 490, Commission of the European Communities, 2009b.

Commission of the European Communities. Roadmap to a Single European Transport Area - Towards a competitive and resource efficient transport system. Technical Report COM(2011) 144, Commission of the European Communities, 2011.

R. G. Conrad and M. A. Figliozzi. The Recharging Vehicle Routing Problem. In eds T. Doolen and E. Van Aken, editor, *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.

C. Contardo, V.C. Hemmelmayr, and T.G. Crainic. Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, 39(12):3185–3199, 2012.

C. Contardo, J.F. Cordeau, and B. Gendron. An Exact Algorithm Based on Cut-and-Column Generation for the Capacitated Location-Routing Problem. *INFORMS Journal on Computing*, 26(1):88–102, 2014a.

C. Contardo, J.F. Cordeau, and B. Gendron. A GRASP + ILP-based metaheuristic for the capacitated location-routing problem. *Journal of Heuristics*, 20(1):1–38, 2014b.

J.F. Cordeau, G. Laporte, and M. Gendreau. *A Tabu Search Heuristic for Periodic and Multi-depot Vehicle Routing Problems*. Centre for Research on Transportation, 1995.

J.F. Cordeau, T.G. Crainic, and V.C. Hemmelmayr. An Adaptive Large Neighborhood Search Heuristic for Two-Echelon Vehicle Routing Problems Arising in City Logistics. Technical Report 42, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, Canada, 2011.

R. Courteaud, Y. Xu, and C. Cérin. Practical solutions for resilience in SlapOS. In *CloudCom*, pages 488–495. IEEE, 2012.

T.G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438, 1997.

T.G. Crainic, N. Ricciardi, and G. Storchi. Advanced Freight Transportation Systems for Congested Urban Areas. *Transportation Research Part C: Emerging Technologies*, 12(2):119–137, 2004.

T.G. Crainic, S. Mancini, G. Perboli, and R. Tadei. Clustering-Based Heuristics for the Two-Echelon Vehicle Routing Problem. Technical Report 46, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, Canada, 2008.

T.G. Crainic, N. Ricciardi, and G. Storchi. Models for Evaluating and Planning City Logistics Systems. *Transportation Science*, 43(4):432–454, 2009.

T.G. Crainic, S. Mancini, G. Perboli, and R. Tadei. A GRASP with Path-Relinking metaheuristic for the Two-Echelon Vehicle Routing Problem. In *Proceedings of the 9th Metaheuristics International Conference (MIC 2011)*, pages 1–7, 2011a.

T.G. Crainic, S. Mancini, G. Perboli, and R. Tadei. Heuristics for the Two-Echelon Vehicle Routing Problem: A Multi-Start Approach. Technical Report 16, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, Canada, 2011b.

B. Crevier, J.F. Cordeau, and G. Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

L. Dablanc. Goods Transport in Large European Cities: Difficult to Organize, Difficult to Modernize. *Transportation Research Part A: Policy and Practice*, 41(3):280–285, 2007.

A. Del Pia and C. Filippi. A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. *International Transactions in Operational Research*, 13(2):125–141, 2006.

L. Delaître. *Méthodologie pour optimiser le transport de marchandises en ville. Application aux villes moyennes et dans le cadre de l'agglomération de La Rochelle.* PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2008.

H. Derbel, B. Jarboui, S. Hanafi, and H. Chabchoub. An Iterated Local Search for Solving A Location-Routing Problem. *Electronic Notes in Discrete Mathematics*, 36: 875–882, 2010.

M. Desrochers. *An Algorithm for the Shortest Path Problem with Resource Constraints.* Cahiers du GÉRAD. École des hautes études commerciales, Groupe d'études et de recherche en analyse des décisions, Montréal, Québec, 1988.

G.M. D'Este. Urban freight movement modelling. In *Handbook of Transport Modelling*, pages 539–552. Elsevier, 2000.

W.J. Dietrich. Approches et pistes suivies en Suisse. In *Études et recherches: L'intégration des marchandises dans le système des déplacements urbains - Actes des Treizièmes Entretiens Jacques Cartier, Montréal, octobre 2000*, pages 191–198. Laboratoire d'Économies des Transports, Lyon, France, 2001.

M. Dror. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42(5):977–978, 1994.

O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized Column Generation. *Discrete Mathematics*, 194(1):229–237, 1997.

J.F. Ehmke. City Logistics. In *Integration of Information and Optimization Models for Routing in City Logistics*, pages 9–22. Springer, 2012.

S. Erdoğan and E. Miller-Hooks. A Green Vehicle Routing Problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, 2012.

J.W. Escobar, R. Linfati, and P. Toth. A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research*, 40(1): 70–79, 2013.

G. Fedak, C. Germain, V. Neri, and F. Cappello. Xtremweb: a generic global computing system. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 582–587, 2001.

D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. A Branch-And-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. *Operations Research*, 45(3): 378–394, 1997.

M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. Solving the Orienteering Problem through Branch-and-Cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Technical report, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, 1990.

S. Fujii, Y. Iida, and T. Uchida. Dynamic simulation to evaluate vehicle navigation. In *Vehicle Navigation & Information Systems Conference Proceedings*, pages 239–244, 1994.

D. Gargett and D. Cosgrove. Predicting traffic growth in Australian cities. In *Papers of the Australasian Transport Research Forum*, 2004.

M. Gendreau and J.Y. Potvin. *A Guide to Tabu Search.* Centre for Research on Transportation, 2003.

Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux. Energy Accounting and Control with SLURM Resource and Job Management System. In M. Chatterjee, J.N. Cao, K. Kothapalli, and S. Rajsbaum, editors, *ICDCN*, volume 8314 of *Lecture Notes in Computer Science*, pages 96–118. Springer, 2014.

B. Gerardin, D. Patier, J.L. Routhier, and E. Segalou. Diagnostic du transport de marchandises dans une agglomération - Programme National Marchandises en Ville. Technical Report 1, Laboratoire d'économie des transports - LET, 2000.

B.E. Gillett and J.G. Johnson. Multi-terminal vehicle-dispatch algorithm. *Omega*, 4(6): 711–718, 1976.

P.C. Gilmore and R.E. Gomory. A Linear Programming Approach to the Cutting Stock Problem–Part II. *Operations Research*, 11(6):863–888, 1963.

J. Gonzalez-Feliu. Two-echelon freight transport optimisation: unifying concepts via a systematic review. *Working papers on operations management*, 2(1):18–30, 2011.

J. Gonzalez-Feliu and J. Morana. A la recherche d'une mutualisation des livraisons en milieu urbain: le cas du groupe NMPP. *Revue Française de Gestion Idustrielle*, 29 (2):71–92, 2010.

J. Gonzalez-Feliu, G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem. Technical Report 02, Politecnico di Torino, 2008.

W.J. Guerrero, C. Prodhon, N. Velasco, and C.A. Amaya. Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, 146(1):359–370, 2013.

P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

V.C. Hemmelmayr, K.F. Doerner, R.F. Hartl, and S. Rath. A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics*, 19(2): 129–156, 2013.

F.L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal Mathematical Physics*, 20:224–230, 1941.

J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

D.L. Iglesias, D. Kondo, and J.M. Marquès. Long-term availability prediction for groups of volunteer resources. *Journal of Parallel and Distributed Computing*, 72(2):281–296, 2012.

P. Jaillet, J.F. Bard, L. Huang, and M. Dror. Delivery Cost Approximations for Inventory Routing Problems in a Rolling Horizon Framework. *Transportation Science*, 36 (3):292–300, 2002.

B. Jarboui, H. Derbel, S. Hanafi, and N. Mladenović. Variable neighborhood search for location routing. *Computers & Operations Research*, 40(1):47–57, 2013.

M. Jepsen, S. Spoorendonk, and S. Ropke. A Branch-and-Cut Algorithm for the Symmetric Two-Echelon Capacitated Vehicle Routing Problem. *Transportation Science*, 47(1):23–37, 2013.

W.C. Jordan. Truck backhauling on networks with many terminals. *Transportation Research Part B: Methodological*, 21(3):183–193, 1987.

I. Kara. Arc based integer programming formulations for the Distance Constrained Vehicle Routing problem. In *2011 3rd IEEE International Symposium on Logistics and Industrial Informatics (LINDI)*, pages 33–38, 2011.

I. Karaoglan and C. Koc. A branch and cut algorithm for the vehicle routing problem with multiple use of vehicles. In *Proceedings of the 41st International Conference on Computers & Industrial Engineering*, 2011.

I. Karaoglan, F. Altiparmak, I. Kara, and B. Dengiz. The location-routing problem with simultaneous pickup and delivery: Formulations and a heuristic approach. *Omega*, 40 (4):465–477, 2012.

R.M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

A.G.H. Kek, R.L. Cheu, and Q. Meng. Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots. *Mathematical and Computer Modelling*, 47(1–2):140–152, 2008.

B.I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

K.M. Kockelman. To LDT or Not to LDT: An Assessment of the Principal Impacts of Light-Duty Trucks. *Transportation Research Record*, 1738:3–10, 2000.

D. Kondo, A. Andrzejak, and D.P. Anderson. On correlated availability in Internet-distributed systems. In *GRID*, pages 276–283. IEEE, 2008.

G. Laporte. Generalized subtour elimination constraints and connectivity constraints. *The Journal of the Operational Research Society*, 37(5):509–514, 1986.

G. Laporte. Location-routing problems. In *Vehicle Routing: Methods and Studies*, pages 163–196. North Holland, Amsterdam, 1988.

G. Laporte. A survey of algorithms for location-routing problems. *Investigación Operativa*, 1(1):93–123, 1989.

G. Laporte and Y. Nobert. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224–226, 1981.

G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6(9):291–310, 1986.

G. Laporte, Y. Nobert, and S. Taillefer. *Solving a Family of Multi-depot Vehicle Routing and Location-routing Problems*. Cahiers du GÉRAD. École des hautes études commerciales, 1987.

E. Le Sueur and G. Heiser. Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns. In *Proceedings of the 2010 Workshop on Power Aware Computing and Systems (HotPower'10)*, pages 1–8, 2010.

A. N. Letchford, S. D. Nasiri, and D. O. Theis. Compact Formulations of the Steiner Traveling Salesman Problem and Related Problems. submitted to European Journal of Operational Research, 2012.

A.N. Letchford, R.W. Eglese, and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94(1):21–40, 2002.

C.K.Y. Lin, C.K. Chow, and A. Chen. A location-routing-loading problem for bill delivery services. *Computers & Industrial Engineering*, 43(1-2):5–25, 2002.

J.R. Lin and H.C. Lei. Distribution systems design with two-level routing considerations. *Annals of Operations Research*, 172(1):329–347, 2009.

S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516, 1973.

S.C. Liu and S.B. Lee. A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration. *The International Journal of Advanced Manufacturing Technology*, 22(11-12):941–950, 2003.

M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2002.

J. Lysgaard. CVRPSEP: A Package of Separation Routines for the Capacitated Vehicle Routing Problem. Technical report, Handelshøjskolen i Århus. Institut for Driftsøkonomi og Logistik, 2003.

J. Lysgaard, A.N. Letchford, and R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

F. Masoero, G. Perboli, and R. Tadei. Valid Inequalities for the Two-Echelon Capacitated Vehicle Routing Problem. Technical Report 39, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, Canada, 2009.

Metrolinx. Freight on Transit Handbook - Case Studies. http://tinyurl.com/loqzyhf, 2012.

C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the Association for Computing Machinery*, 7(4): 326–329, 1960.

H. Min. Consolidation Terminal Location-Allocation and Consolidated Routing Problems. *Journal of Business Logistics*, 17(2):235–263, 1996.

H. Min, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108(1):1–15, 1998.

A. Mingozzi, R. Roberti, and P. Toth. An Exact Algorithm for the Multitrip Vehicle Routing Problem. *INFORMS Journal on Computing*, 25(2):193–207, 2013.

Minister of the State Environmental Protection Administration of China. Report on the State of the Environment in China 2000. http://tinyurl.com/n6ofemz, 2007.

N.T. Moungla, L. Létocart, and A. Nagih. Solutions diversification in a column generation algorithm. *Algorithmic Operations Research*, 5(2):86–95, 2010.

I. Muter, J.F. Cordeau, and G. Laporte. A Branch-and-Price Algorithm for the Multidepot Vehicle Routing Problem with Interdepot Routes. *Transportation Science*, 48 (3):425–441, 2014.

G. Nagy and S. Salhi. The many-to-many location-routing problem. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 6(2):261–275, 1998.

G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

G.L. Nemhauser and M.W.P. Savelsbergh. Functional description of MINTO, a Mixed INTeger Optimizer. Technical report, Georgia Institute of Technology, Atlanta, GA, 1996.

P.K. Nguyen, T.G. Crainic, and M. Toulouse. A tabu search for Time-dependent Multizone Multi-trip Vehicle Routing Problem with Time Windows. *European Journal of Operational Research*, 231(1):43–56, 2013.

V.P. Nguyen, C. Prins, and C. Prodhon. Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216(1):113–126, 2012a.

V.P. Nguyen, C. Prins, and C. Prodhon. A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence*, 25(1):56–71, 2012b.

A. Olivera and O Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47, 2007.

A.C. Orgerie, L. Lefevre, I. Guerin-Lassous, and D.M. Lopez Pacheco. ECOFEN: An End-to-end energy Cost mOdel and simulator For Evaluating power consumption in large-scale Networks. In *Proceedings of the 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6. IEEE Computer Society, 2011.

D. Patier. *La Logistique dans la ville.* CELSE, 2002.

J. Perl and M.S. Daskin. A Warehouse Location Routing Model. *Transportation Research Part B*, 19(5):381–396, 1985.

R.J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1–3):69–92, 2003.

D. Pisinger and S. Røpke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

J.Y. Potvin and J.M. Rousseau. An Exchange Heuristic for Routeing Problems with Time Windows. *The Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

É. Prescott-Gagnon, G. Desaulniers, and L.M. Rousseau. *Heuristics for an Oil Delivery Vehicle Routing Problem*. Cahiers du GÉRAD. Groupe d'études et de recherche en analyse des décisions, 2010.

C. Prins. Efficient Heuristics for the Heterogeneous Fleet Multitrip VRP with Application to a Large-Scale Real Case. *Journal of Mathematical Modelling and Algorithms*, 1(2):135–150, 2002.

C. Prins. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

C. Prins, C. Prodhon, and R. Wolfler Calvo. Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. *4OR*, 4(3):221–238, 2006.

C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler Calvo. Solving the Capacitated Location-Routing Problem by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic. *Transportation Science*, 41(4):470–483, 2007.

C. Prodhon and C. Prins. A Memetic Algorithm with Population Management (MA—PM) for the Periodic Location-Routing Problem. In M.J. Blesa, C. Blum, C. Cotta, A.J. Fernández, J.E. Gallardo, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 43–57. Springer Berlin Heidelberg, 2008.

C. Prodhon and C. Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17, 2014.

J.G. Rakke, M. Stålhane, C. Rørholt Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C-emerging Technologies*, 19(5):896–911, 2011.

S. Rath and W.J. Gutjahr. A math-heuristic for the warehouse location–routing problem in disaster relief. *Computers & Operations Research*, 42:25–39, 2014.

J. Renaud, G. Laporte, and F.F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235, 1996.

Y. Rochat and É.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

J.P. Rodrigue. *The Geography of Transportation Systems*. Routledge - Taylor & Francis Group, 2013.

S. Ropke and D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40(4): 455–472, 2006.

M.A. Salazar-Aguilar, A. Langevin, and G. Laporte. The synchronized arc and node routing problem: Application to road marking. *Computers & Operations Research*, 40(7):1708–1715, 2013.

S. Salhi and R.J. Petch. A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips. *Journal of Mathematical Modelling and Algorithms*, 6(4):591–613, 2007.

M. Schneider, A. Stenger, and D. Goeke. The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. Technical Report 02, Technische Universität Kaiserslautern, 2012.

M. Schneider, A. Stenger, and J. Hof. An Adaptive VNS Algorithm for Vehicle Routing Problems with Intermediate Stops. Publications of Darmstadt Technical University, Institute for Business Studies (BWL) LPIS-01/2014, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 2014.

M. Schwengerer, S. Pirkwieser, and G.R. Raidl. A Variable Neighborhood Search Approach for the Two-Echelon Location-Routing Problem. In Jin-Kao Hao and Martin Middendorf, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2012.

R. D. Singh. *Location-routing Problems*. PhD thesis, Pennsylvania State University, 1998.

J.P. Smets-Solanes, C. Cérin, and R. Courteaud. SlapOS: A Multi-Purpose Distributed Cloud Operating System Based on an ERP Billing Model. In H.A. Jacobsen, Y. Wang, and P. Hung, editors, *IEEE SCC*, pages 765–766. IEEE, 2011.

O.J. Smith, N. Boland, and H. Waterer. Solving shortest path problems with a weight constraint and replenishment arcs. *Computers & Operations Research*, 39(5):964–984, 2012.

Société française de Recherche Opérationnelle et Aide à la Décision (ROADEF). Google ROADEF/EURO challenge 2011–2012: Machine Reassignment. http://challenge.roadef.org/2012/en/, 2011.

M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

É.D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

É.D. Taillard, G. Laporte, and M. Gendreau. Vehicle Routing With Multiple Use Of Vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070, 1995.

E. Taniguchi, R.G. Thompson, and T. Yamada. Recent advances in modelling city logistics. In E. Taniguchi and R.G. Thompson, editors, *City Logistics II*, pages 3–34. Institute of Systems Science Research, Kyoto, 2001.

E. Taniguchi, T. Yamada, and Y. Kakimoto. *Models for evaluating City Logistics measures*, volume Proceedings of the Eastern Asia Society for Transportation Studies: Hanoi 2001, vo1.3, no.2, pages 511–526. Eastern Asia Society for Transportation Studies, 2003.

C.D. Tarantilis, E.E. Zachariadis, and C.T. Kiranoudis. A Hybrid Guided Local Search for the Vehicle-Routing Problem with Intermediate Replenishment Facilities. *INFORMS Journal on Computing*, 20(1):154–168, 2008.

P.M. Thompson and H.N. Psaraftis. Cyclic Transfer Algorithm for Multivehicle Routing and Scheduling Problems. *Operations Research*, 41(5):935–946, 1993.

F.A. Tillman and T.M. Cain. An Upperbound Algorithm for the Single and Multiple Terminal Delivery Problem. *Management Science*, 18(11):664–682, 1972.

United Nations. Population Challenges and Development Goals. Technical Report ST/ESA/SER.A/248, United Nations, 2005.

F. Vanderbeck and M.W.P. Savelsbergh. A generic view of Dantzig–Wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

F. Vanderbeck and L. Wolsey. Reformulation and decomposition of integer programs. CORE Discussion Papers 2009016, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2009.

T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3):611–624, 2012.

J.G.S.N. Visser, A.J. Van Binsbergen, and T. Nemoto. Urban Freight Transport Policy and Planning. In E. Taniguchi and R.G. Thompson, editors, *City Logistics I*. Institute of Systems Science Research, Kyoto, 1999.

M. Weiser, B.B. Welch, A.J. Demers, and S. Shenker. Scheduling for Reduced CPU Energy. In *OSDI*, pages 13–23. USENIX Association, 1994.

World Health Organization. Global status report on road safety 2013: Supporting a decade of action. Technical report, World Health Organization, 2005.

A. Wren and A. Holliday. *Computer Scheduling of Vehicles from One Or More Depots to a Number of Delivery Points*. University of Leeds, Centre for Computer Studies, Operational Research Unit, 1972.

P.C. Yellow. A Computational Modification to the Savings Method of Vehicle Scheduling. *Journal of the Operational Research Society*, 21(2):281–283, 1970.

B.D. Young, J. Apodaca, L.D. Briceno, J. Smith, S. Pasricha, A.A. Maciejewski, H.J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou. Deadline and energy constrained dynamic resource allocation in a heterogeneous computing environment. *The Journal of Supercomputing*, 63(2):326–347, 2013.