UNIVERSITAT POLITÈCNICA DE CATALUNYA
(UPC) – BARCELONATECH

IN COLLABORATION WITH
UNIVERSITAT OBERTA DE CATALUNYA (UOC)

BACHELOR'S DEGREE THESIS

Applications of Metaheuristics to the Physical Internet

Author:
Quim ARNAU ORTEGA

Supervisors:
Dr. Ángel A. JUAN PÉREZ
Computer Science, Multimedia and Telecommunication dept. at UOC

Dr. Pau FONSECA I CASAS
Statistics and Operations Research dept. at UPC

A thesis developed in the
Internet Computing & Systems Optimization (ICSO at UOC)

and submitted to the
Facultat d'Informàtica de Barcelona (FIB at UPC)

in fulfillment of the requirements for the
BSc in Informatics Engineering, Majored in Computing

January 23rd, 2018

Abstract

Logistics is one of the backbones sustaining our way of living: the meals we order in a restaurant, the products we buy in the supermarket just around the corner, the gifts we buy in online retailers such as Amazon. Transportation is one of the critical processes in logistics, which is estimated to be a burden between 5 and 15% of most countries' GDP. Furthermore, transportation systems have been increasingly growing for the last decades, in terms of the weight and value of goods they ship. The U.S. case study is a clear example. However, the transportation and distribution systems we currently pursue and use are inefficient and unsustainable. The physical internet (PI) initiative was introduced in 2010 to cope with these problems. It was inspired in the way the digital internet (DI) interconnects its heterogeneous systems through data packets. Its main idea resides in using the so called π-containers: smart, traceable, modular and reusable containers interconnected with IoT systems.

This thesis provides a starting point to link modern logistics systems such as the PI with metaheuristics, a type of approximate algorithms widely used in the field of operational research (OR). A realistic novel problem within the PI consisting in container transportation throughout a spoke-hub network is defined, analyzed and solved by different means. A deterministic heuristic based on discrete-event simulation is proposed as a first approach to address this problem. Then a biased randomization of the heuristic (BRH) is incorporated into a multi-start framework (BR-MS) to generate higher-quality solutions. Finally, our methodology is extended to a metaheuristic, a variable neighborhood search (VNS) is used to tackle this problem. Several computational experiments are carried out on a set of 20 new benchmark instances, adapted from real road networks to illustrate the problem and compare the performance of the solving approaches mentioned.

Keywords: logistics, distribution systems, physical internet, smart containers, IoT systems, metaheuristics, operational research, container transportation, spoke-hub networks, heuristics, discrete-event simulation, biased randomization, multi-start framework, variable neighborhood.

Resum

La logística és un dels pilars que sustenta la nostra manera de viure: els àpats que demanem en un restaurant, els productes que comprem al supermercat de la vora, els regals que demanem a les botigues en línia com Amazon. El transport és un dels processos crítics de la logística, estimat com una càrrega entre el 5 i el 15% del PIB de molts països. A més, els sistemes de transport han anat creixent cada vegada més en les últimes dècades pel que fa al pes i el valor de les mercaderies que envien. El cas d'estudi dels EUA és un clar exemple. Tanmateix, els sistemes de transport i distribució que perseguim i utilitzen actualment són ineficients i insostenibles. La iniciativa internet físic (IF) es va introduir l'any 2010 per fer front a aquests problemes. Es va inspirar en la forma en què l'internet interconnecta els seus sistemes heterogenis a través de paquets de dades. La seva principal idea resideix a utilitzar els anomenats π-containers: contenidors intel·ligents, que poden ser rastrejats, modulars i reutilitzables, interconnectats amb sistemes IoT.

Aquesta tesi proporciona un punt de partida per vincular els sistemes logístics moderns com l'IF amb les metaheurístiques, un tipus d'algorismes aproximats àmpliament utilitzats en el camp de la investigació operativa (IO). Un problema original i realista entorn de l'IF, que consisteix en el transport de contenidors a través d'una xarxa hub-and-spoke, és definit, analitzat i resolt per mitjà de diferents mètodes. Es proposa una heurística determinista basada en la simulació d'esdeveniments discrets com a primer enfocament per abordar aquest problema. A continuació, una aleatorització esbiaixada de l'heurística (BRH) s'incorpora a un marc multi-start (BR-MS) per generar solucions de major qualitat. Finalment, la nostra metodologia s'estén a una metaheurística, un variable neighborhood search (VNS) s'utilitza per afrontar aquest problema. Diversos experiments computacionals es duen a terme en un conjunt de 20 noves instàncies de referència, adaptades de xarxes reals de carreteres per il·lustrar el problema i comparar el rendiment dels diferents mètodes.

Paraules clau: logística, sistemes de distribució, internet físic, contenidors intel·ligents, sistemes IoT, metaheurístiques, investigació operativa, transport de contenidors, xarxes hub-and-spoke, heurística, simulació d'esdeveniments discrets, aleatorització esbiaixada, marc multi-start, variable neighborhood search.

Resumen

La logística es uno de los pilares que sustenta nuestra manera de vivir: las comidas que pedimos en un restaurante, los productos que compramos en el supermercado a la vuelta de la esquina, los regalos que pedimos en tiendas en línea como Amazon. El transporte es uno de los procesos críticos de la logística, estimado como una carga entre el 5 y el 15 % del PIB de muchos países. Además, los sistemas de transporte han ido creciendo cada vez más en las últimas décadas en cuanto al peso y el valor de las mercancías que envían. El caso de estudio de EE.UU. es un claro ejemplo. Sin embargo, los sistemas de transporte y distribución que perseguimos y utilizamos actualmente son ineficientes e insostenibles. La iniciativa internet físico (IF) se introdujo en 2010 para hacer frente a estos problemas. Se inspiró en la forma en que internet interconecta sus sistemas heterogéneos a través de paquetes de datos. Su principal idea reside en utilizar los llamados π-containers: contenedores inteligentes, que pueden ser rastreados, modulares y reutilizables, interconectados con sistemas IoT.

Esta tesis proporciona un punto de partida para vincular los sistemas logísticos modernos como el IF con las metaheurísticas, un tipo de algoritmos aproximados ampliamente utilizados en el campo de la investigación operativa (IO). Un problema original y realista dentro del IF, que consiste en el transporte de contenedores a través de una red hub-and-spoke, es definido, analizado y resuelto por medio de diferentes métodos. Se propone una heurística determinista basada en la simulación de eventos discretos como primer enfoque para abordar este problema. A continuación, una aleatorización sesgada de la heurística (BRH) se incorpora a un marco multi-start (BR-MS) para generar soluciones de mayor calidad. Finalmente, nuestra metodología se extiende a una metaheurística, un variable neighborhood search (VNS) se utiliza para afrontar este problema. Varios experimentos computacionales se llevan a cabo en un conjunto de 20 nuevas instancias de referencia, adaptadas de redes reales de carreteras para ilustrar el problema y comparar el rendimiento de los diferentes métodos.

Palabras clave: logística, sistemas de distribución, internet físico, contenedores inteligentes, sistemas IoT, metaheurísticas, investigación operativa, transporte de contenedores, redes hub-and-spoke, heurística, simulación de eventos discretos, aleatorización sesgada, marco multi-start, variable neighborhood search.

Acknowledgements

Many thanks and gratitude to my supervisors, Dr. Ángel A. Juan and Dr. Pau Fonseca for their support, trust and tireless advice. It's been a pleasure to learn from your novel ideas while developing this thesis.

I'd like to thank all my colleagues at the Internet Computing & Systems Optimization (ICSO) research group, which have been incredibly supportive during my whole stay at IN3. I really appreciate your help and warm reception, as not only you did introduce me to research but taught me not to lose it when unexpected results went the other way. Special thanks to Lluc Bové, Dr. Sara Hatami, Dr. Alejandro Estrada, Dr. Javier Panadero and Dr. Laura Calvet.

Last but not least, I want to thank my family and close friends which kept laughing at me every time I tried to explain what I've been doing these last months. There's no better way to find out that you still need to improve the way to communicate your work to others.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

BR	**B**iased **R**andomization
BRP	**B**iased-**R**andomized **P**rocedure
BRH	**B**iased-**R**andomized **H**euristic
COP	**C**ombinatorial **O**ptimization **P**roblem
COPDI	**COP** with **D**ynamic **I**nputs
CT	**C**omputational **T**ime
DC	**D**istribution **C**enter
DES	**D**iscrete-**E**vent **S**imulation
DI	**D**igital **I**nternet
IOT	**I**nternet **O**f **T**hings
MCND	**M**ulti **C**ommodity **N**etwork **D**esign problem
MCS	**M**onte-**C**arlo **S**imulation
MS	**M**ulti-**S**tart
NP	**N**ondeterministic **P**olynomial time
OOP	**O**bject-**O**riented **P**rogramming
OR	**O**perational **R**esearch
PFSP	**P**ermutation **F**lowshop **S**cheduling **P**roblem
PI	**P**hysical **I**nternet
PM	**P**opulation-based **M**etaheuristics
RFID	**R**adio-**F**requency **ID**entification
SA	**S**imulated **A**nnealing
SCOP	**S**tochastic **COP**
SM	**S**olution-based **M**etaheuristics
ST	**S**hipping **T**ime
STPD	**S**kewed **T**heoretical **P**robability **D**istribution
TCARP	**T**ime **C**apacitated **A**rc **R**outing **P**roblem
TSP	**T**raveling **S**alesman **P**roblem
TT	**T**otal **T**ime
VNS	**V**ariable **N**eighborhood **S**earch
VRP	**V**ehicle **R**outing **P**roblem
WSAN	**W**ireless **S**ensor and **A**ctor **N**etworks

Chapter 1

Introduction

Logistics is one of the pillars that sustain our lifestyle, in other words, it makes possible to have a store full of fruits right in the corner, it also enables us to purchase almost any product we can imagine on the Internet and have it at home in less than a couple of days [1].

We are surrounded by a huge offer of goods that had to be manufactured, transported, stored and supplied. For instance, when we order a side dish in a restaurant all the ingredients needed had to be taken care in anticipation, that is, from the harvester that grows them, to the chef that cooks for you, passing through the manufacturer and distributor. One of the key aspects in that process is transportation. From an economical point of view, freight transportation has an enormous cost in most developed countries, the U.S. is a clear example. In 2014 purchases of transportation goods and services accounted for about 9% of U.S. gross domestic product (GDP), the public and private sectors spent over $125 billion on transportation construction and over 12 million workers were employed on this sector, representing almost 9% of the nation's labor force [2].

Furthermore, road transportation is the main way of transporting goods in Europe. In 2014 the share of European inland freight that was transported by road (74.9%) was more than four times as high as the share transported by rail (18.4%), while the remainder (6.7%) was carried along inland waterways [3].

Nonetheless, the transportations systems we are currently using are inefficient and unsustainable [4]. Most countries' worldwide logistics costs grow faster than world trade, representing a burden between 5 and 15 % of their GDP, as we have mentioned earlier. Greenhouse gas emissions keep increasing even though drastic reductions are sought by governments in Europe. For instance, the Paris agreement states to have a 20% cut in greenhouse gas emissions compared with 1990 by 2020 [5]. However, freight transportation is one of the heaviest greenhouse gas generator, energy consumer, polluter and material waster [1] and it hasn't stop growing so far [3]. Another symptom of the unsustainable system we currently have can be seen from a social perspective: the precarious work conditions that truckers have to face sometimes and more generally the logistics sector [6].

It seems more clear now the importance of logistics and more specifically the part that road transportation takes. It is also noticeable the effort needed, in terms of research, innovation, initiative, development and multidisciplinary collaboration among academia, industry and government to turn around this situation, not with marginal small steps but with a gradual change of paradigm. The margin of improvement — economically, environmentally and socially speaking — is more than ensured if we are able to establish a new sustainable and efficient transportation system.

The solution proposed by [7] is based on the *physical internet* (PI), a concept inspired in the way data packets are transported on the *digital internet* (DI). To put it in a nutshell, the PI pursues to address what the author calls *the global logistics sustainability grand challenge*, that is, "*to enable the global sustainability of physical object mobility, storage, realization, supply and usage*" using the DI as a metaphor and taking into account the differences between both models.

The PI initiative is explored in this thesis and a specific realistic novel problem concerning the transportation of containers in a spoke-hub network is tackled. The problem analyzed consists in optimizing the transportation time of a set of containers, which have different origin locations, final destinations and a deadline to fulfill. These containers can be temporarily stored in a type of intermediate nodes called hubs. We dispose of a fleet of trucks that are able to transport one container at a time, but also to load and unload different containers. On the other hand, the drivers can only work for a limited number of hours and they need to drive their trucks back to their original depots. It is already conspicuous that the cost will need to be unambiguously defined and it is not so clear whether the main interest will be to minimize the total time, because other criteria might arise as important.

1.1 Motivation

As stated before, the sector of logistics is not only relevant by its size but by its growth. The U.S. Bureau of Transportation Statistics claims that "*by 2040 long-haul freight truck traffic in the United States is expected to increase dramatically on the National Highway System*" [8]. In [9] it is also claimed a freight transport growth of 37% from 2005 to 2025 in France. And this trend is not only characteristic of France but of the whole OECD zone: "*road freight transport volume in the European Union is expected to grow 78% between 2000 and 2030*" according to [10].

A wide variety of companies, from the medium and small enterprises to the big multinationals around the world need to plan their distribution routes. However all of them do it on their own with their own resources: "*they design, run and optimize independently their private distribution networks, investing in Distribution Centers (DCs) or engaging in long-term leases or contracts*" [1]. There are 535,000 distribution centers in the U.S. only. However, most of them are used by a single company and most companies use often a single DC and generally less than 20. In [11] the potential of an open supply web, that is, an open network of shared DCs, is studied and its performance improvements are reported. With this research seems clear that there's still so much to be done in so many aspects of logistics. This should serve as motivation to keep supporting the PI initiative so that the grand challenge could be faced and brought to reality.

So it is more than obvious that with this greater demand for road transportation comes the need of innovative logistics approaches to make businesses successful reducing economic, environmental and social negative impacts.

1.2 Scope

This thesis covers topics such as metaheuristics, logistics, artificial intelligence, operational research and combinatorial optimization, among others. One of its strengths in the context of the bachelor's degree syllabus, and more specifically of our major, is that it is directly related to some courses and topics taught, e.g., Algorithms, Artificial Intelligence, Numeric Computation to remark some of the main courses.

It's also important to specify what is in the scope of the project and what is beyond it. We have mainly focused in the implementation of different techniques to solve this problem, precisely three solving approaches are stated in section 1.3. To develop them it's been necessary to do a previous study on different topics such as the PI, similar routing problems, as well as to the background theory underneath these techniques. This comprises the core of the project, however, it's also been necessary to test our algorithms with a set of instances, that is, a set of computational experiments and the further analysis of the results. Because there are not benchmark instances for this problem, as we haven't found any related literature solving it, it's also part of the scope to generate our own ones.

The time spent on some tasks has varied according to the planning and the adjustments needed. It is not our intention to perform a complex statistical analysis, but to quantify and measure the performance of our algorithms and to compare them.

1.3 Goals

The purpose of this project is to solve the preceding problem by different means. For that reason, the PI initiative has also been extensively studied, so that we can adapt its main ideas, with regard to the transportation layer, in our solving approaches. To sum up, the set of goals that we want to achieve are the following ones:

- Define a realistic novel container transportation problem in the context of the PI.
- Explore the PI concept through an extensive study of the related literature focusing on the transportation layer of the whole initiative.
- Give sufficient background on combinatorial optimization problems and their most common optimization methods.
- Perform an exhaustive literature review on similar routing problems looking for already given state-of-the-art solutions.
- Design and develop three different solving approaches:
    1. A greedy deterministic heuristic algorithm based on discrete-event simulation.
    2. A biased-randomized heuristic (BRH) within a multi-start framework (BR-MS).
    3. A variable neighborhood search (VNS) metaheuristic with the use of the randomized heuristic.

- Create a set of standardized benchmark instances with different network and truck fleet sizes.
- Perform several computational experiments, parameter fine-tuning and statistical analysis of the results.

1.4 Methodology

Because this work has been developed in the ICSO research group within the Open University of Catalonia (UOC) we have followed the same methodologies and use most of the same tools they usually use. Some other management tools and techniques have been suggested by our tutor and adopted from the beginning, e.g., using an agile methodology, keep a canvas up to date using *Trello* or any similar tool.

We decided to use *Java* as the programming language to develop our algorithms, because it is their usual first choice and it seemed convenient to adopt an object-oriented programming (OOP) language to represent the domain related to the problem. We've also used *Python* in the process of adaptation and creation of the new instances, and R mainly for the analysis of the results as well as for basic descriptive statistics. We've chosen *Git* as a version control tool, and we have indistinctly used both OS systems, *Windows 10* and *Ubuntu 16.04*. Notice that we are uploading our code in 2 different repositories so that both supervisors can have access to it.

This document has been written in LaTeX because is the de facto standard for the communication and publication of scientific documents [12] and it has been shared with our supervisors. To keep track of all the references we have used *Mendeley* and *Biblatex* because of their easy integration with *Overleaf*, a collaborative writing and publishing system for academic papers.

Monitoring tools

With regard to the project management, we have followed an agile approach, as suggested, with 2-week duration sprints, so that short-term goals can be properly defined and scheduled in the form of small tasks. This way we can ensure that any deviation on the plan is rapidly detected and taken care of. In order to do so, we have used *Microsoft team foundation server* (TFS) so that the supervisors are able to know at all time what are we working on, and how is the project going forward. In some bigger projects involving more than one person it is convenient to use specific agile methodologies, e.g., Scrum, Extreme programming, however, as this is a solo project it is more than sufficient to make short cycles and make use of a canvas.

1.5 Outline

The rest of the thesis is structured as follows. Chapter 2 provides the theoretical background and context to the PI and to combinatorial optimization problems (COP) and it introduces the solving methodologies used. The literature review on the PI and the problem defined will be presented in Chapter 3. Chapter 4 will formally define the problem presenting an example of instance and a feasible solution to it. The methodology used in our solving approaches, their validation, implementation and the strategy to create standardized instances will be explained in Chapter 5. Chapter 6 will describe the computational experiments performed. The results of the experiments will be shown in Chapter 7. Chapter 8 will present a general discussion and analysis of the results. Details on the project management will be given in Chapter 9. Chapter 10 will discuss the sustainability-related issues of this study. Finally, Chapter 11 will summarize the conclusions of this thesis remarking our contributions and giving great importance to the opening of new threads and further lines of research.

Chapter 2

Background

The following section is dedicated to explain the context and background of the PI. It'll be mostly based on the works of B. Montreuil from 2010 until now. We still will make textual quotations and in text cites when needed. However, we might adapt some texts of his articles in order to summarize the main ideas. After that, a brief description in combinatorial optimization problems will be given. Finally, the most common optimization approaches will be reviewed.

2.1 The physical internet

The term *physical internet* (PI) appeared for the first time in June 2006 on the front page of The Economist [13]. The article presented a survey of logistics but apart from the headline, there was no further mention of the term. It was this set of articles that rose Montreuil's attention and curiosity for the term, which would lead to the publication of his first PI article in 2010 [4].

As stated in [14] "*the PI is an open global logistics system founded on physical, digital and operational interconnectivity through encapsulation, interfaces and protocols*", in other words, it aims to enable an efficient, sustainable, adaptable and resilient Logistics web. This term can be defined as the "*set of openly interconnected physical, digital, human, organizational and social agents and networks aiming to serve efficiently and sustainably the logistics needs of people, organizations, territories and society*" [1]. The Logistics web layer would play the role of interface between the PI and the users of logistics services [15] and it would be compound of 5 interconnected substructures: the realization web (production centers), the distribution web (open warehouses and distribution centers), the mobility web (hubs, transits, movers and other infrastructure), the supply web (open suppliers and subcontractors) and the service web (open users and service providers) [1].

2.1.1 Goals

The author presents three perspectives from which the goals of the phisical internet could be classified in:

- Economical: reduce the global economic burden in global logistics mentioned earlier, while unlocking highly significant gains in production, transportation, and business productivity.

- Environmental: reduce by an order of magnitude the direct and indirect logistics-induced global pollution, including greenhouse gas emission and material waste, as well as energy consumption.
- Societal: improve the working conditions in the logistic sector, as well as increase the quality of life of the overall population by making much more accessible the physical objects across the world.

2.1.2 The grand challenge

But it is necessary to begin with the actual picture of the current logistics system. The author in [7] starts with the assertion that "*the way physical objects are transported, handled, stored, realized, supplied and used throughout the world is not sustainable economically, environmentally and socially*". He termed this problem as the "*global logistics sustainability grand challenge*", and its goal is to "*enable the global sustainability of physical object mobility (transportation, handling), storage, realization (production, assembly, finishing, refurbishing and recycling), supply and usage*".

The information and communication technologies community was once stuck due to millions of unconnected computers. When looking for a way to transform the current paradigm, at that time they relied on a logistics metaphor: building the information highway. The key enabler was the transmission of formatted data packets through heterogeneous equipment following specific standards and protocols, having as a result an open and interconnected distributed network infrastructure. And it completely reshaped the way communication takes place in the present.

Now the idea is to do the same in the other direction: the physical world should exploit the DI metaphor in order to interconnect a distribution network of physical objects. However, it's not about copying the same paradigm but to use it as an inspiration. We need to take into account the differences: a truck has to be driven by a person, it's not like data transfers that might happen almost instantly. It will always take time to transport a truck from Paris to Rome, but there are many ways of doing it.

2.1.3 Unsustainability symptoms

The author summarizes the evidences supporting the unsustainability assertion through a set of thirteen symptoms, which are described below and sustained by some additional references.

1. **We are shipping air and packaging**. In the U.S. trailers are approximately 60% full when traveling loaded [16, 17]. The global transport efficacy has been estimated to be lower than 10% [18].
2. **Empty travel is more usual than expected**. In the U.K. 27% of truck kilometers was reported to be empty travel in 2004 [19]. The U.S. industry average was that 20% of all miles are driven with a completely empty trailer [18].
3. **Truckers are suffering from precarious work conditions**. Truckers spend a lot of hours on the road, often away from home for long periods. [20] found that 58%

of the accidents reported by drivers in the U.S. were deemed to be fatigue and sleep deprivation related. [21] stated that the shift workers with the lowest mean hours of daily sleep are truck drivers, at 3.5 hours/24 hours.

4. **Products mostly sit idle, stored where unneeded**. Manufacturers, distributors and retailers among others, are continuously storing huge quantities of products through their networks of warehouses and DCs. The average investment in all U.S. business inventories was $101 B in 2005 [22], as an indication of the size of inventory.

5. **Production and storage facilities are poorly used**. Because of the seasonal nature of some products warehouses are underutilized in some periods of the year, whereas in others they are full and suffering from inefficiencies to be managed.

6. **Many products never sold, never used**. In the food industries, those products not meeting specific sizes, colors or shapes will be automatically discarded. In addition, many of these products are wasted at an alarming rate: 12% in transit and 25% at retail.

7. **Products do not reach those who need them the most**. In less developed countries the transportation and logistics infrastructures levels significantly decrease making it difficult to reach those in most need.

8. **Fast and reliable intermodal transport is still a dream**. In general, synchronization is poor and interfaces are so badly designed that intermodal routes are mostly time and cost inefficient and risky. For instance, trucks are much more used than trains while the former emit twenty times more CO_2 than the latter.

9. **Getting products in, through, and out of cities is a nightmare**. Most cities are not designed and equipped for easing freight transportation, handling and storage.

10. **Products unnecessarily move**. Products commonly travel thousands of kilometers that could have been avoided by routing them smartly and/or making them much nearer to their point of use.

11. **Networks are neither secure nor robust**. There is extreme concentration of operations in a limited number of centralized production and distribution facilities, with travel along a narrow set of high-traffic routes. This makes the logistic networks and supply chains of so many businesses, insecure in face of robbery and terrorism acts, and not robust in face of natural disasters and demand crises.

12. **Smart automation and technology are hard to justify**. Vehicles, handling systems, and operational facilities have to deal with so many types of materials, shapes, and unit loads.

13. **Innovation is strangled**. Innovation is bottlenecked, notably by lack of generic standards and protocols, transparency, modularity, and systemic open infrastructure.

2.1.4 The solution proposed

The PI vision can be defined through the following thirteen characteristics.

1. **Encapsulate merchandises in world-standard smart green modular containers**. The DI designed standardized packets embedding information to ease the communication, as well as protocols for their transit and process in different systems across networks [23]. The PI on the other hand, encapsulates physical objects in π-containers, standardized, smart, green and modular containers. See [24, 25] for more details on these smart containers, represented in Figure 2.1.

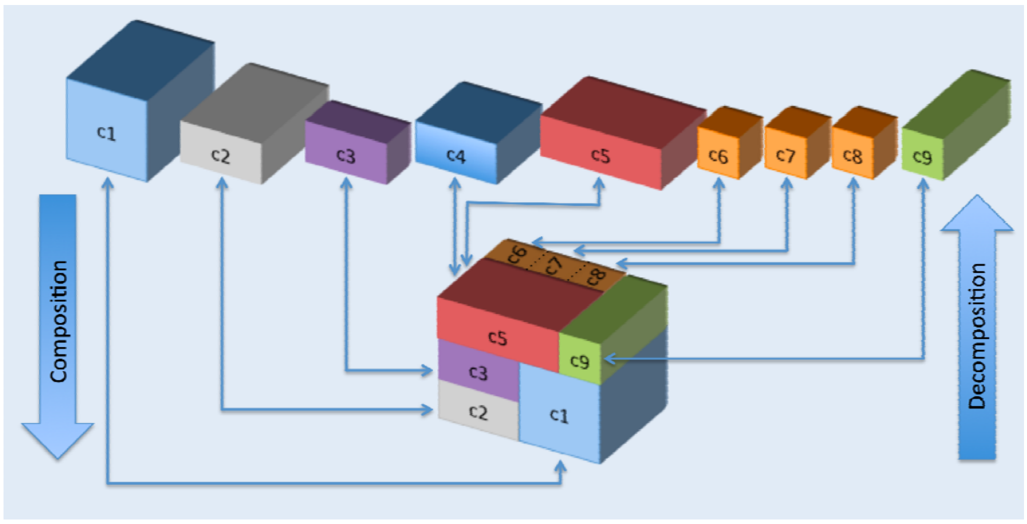


FIGURE 2.1: Modularity of π-containers.

2. **Aiming toward universal interconnectivity**. The operations of unloading, orientation, storage and loading, applied to π-containers in a smart automated and/or human-assisted way should be generalized and functionally standardized.

3. **Evolve from material to π-container handling and storage systems**. In the PI there are only π-container material handling and storage systems to enable their fast, cheap, easy and reliable input, storage, composing, decomposing, monitoring, protection and output through smart, sustainable and seamless automation and human handling. There's a wide terminology for the sites, facilities and systems within the π-nodes, locations expressly designed to perform operations on π-containers, which can be found in [7].

4. **Exploit smart networked container embedding smart objects**. Each π-container has a unique worldwide identifier similar to the MAC address in the DI and a smart tag to insure the identification, integrity, routing, conditioning, monitoring, traceability, and security. Here is where the Internet of Things (IoT) [26] could be introduced, to enable the interconnection of physical objects with smart connective technologies such as RFID and GPS. The schematics of this interrelation are shown in Figure 2.2, which was extracted from [1].

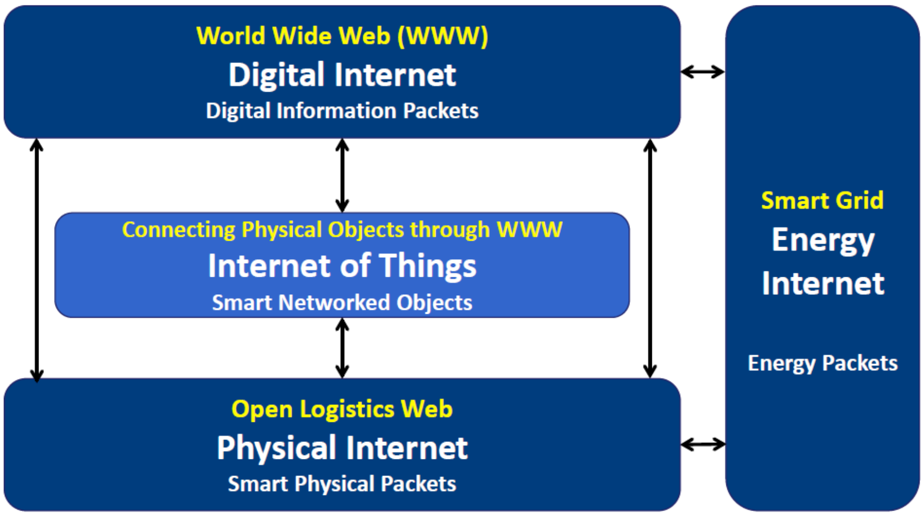


FIGURE 2.2: Contextualizing the physical internet.

5. **Evolve from point-to-point hub-and-spoke transport to distributed multi-segment intermodal transport**. In the DI data packets do not travel directly from origin node A to destination node B, but following the path that routing algorithms tell. Nowadays, if a trailer fully loaded with containers had to be transported a long distance it would be most likely that the task was assigned to a single driver. He would have to perform a multi-day trip, driving all the way to the destination, sleeping and eating in the truck or doing small breaks. Once the load were delivered he would try to pick up the nearest cargo that had to travel to the origin node, to avoid empty travel. However, in the PI this trailer would have been assigned to different drivers working in small shifts, or if possible it would have been transported to the next π-hub so that intermodal transport could be performed, e.g., using trains, ships or planes alongside other π-containers heading to similar destinations.

6. **Embrace a unified multi-tier conceptual framework**. The physical internet is to be based on the same conceptual framework whatever the scale of the involved networks. This can be seen with a set of networks being embedded in wider networks, all of them operating under the PI standards and protocols. An example of two interconnected intra-state inter-city networks can be found in [7], one in the Québec province of Canada coupled to another in the northeastern states of the US. Another road network case study based in Eastern Canada is explored in [27].

7. **Activate and exploit an Open Global Supply Web**. Shift from private supply networks to an Open Global Supply Web enabling the physical equivalents of Intranets, Virtual Private Networks, Cloud Computing and Cloud Storage in the DI. Open supply webs have their nodes accessible to producers, distributors, logistics providers, retailers and users.

8. **Design products fitting containers with minimal space waste**. It is quite obvious that the objects carried in π-containers have to be designed so as to minimize the space waste and optimize their fitting in the different containers standards.

   Another way of managing space would be to design physical objects in a way that only their key components and modules had to be shipped so that they could be easily finished in near points of use.

9. **Minimize physical moves and storages by digitally transmitting knowledge and materializing objects as locally as possible**. With open production centers capable of locally realizing (making, assembling, finishing, personalizing) for clients a wide variety of products from combinations of digitally transmitted specifications.

10. **Deploy open performance monitoring and capability certifications**. For instance, a port would openly post on the Internet its live and historical performance, e.g., ship unloading and loading times, container sojourn times in the port, etc.

11. **Prioritize webbed reliability and resilience of networks**. The webbing of the networks should allow the PI to insure its own robustness and resilience to unforeseen events, just as the DI is able to insure with its protocols and standards.

12. **Stimulate business model innovation**. The DI has created loads of new business and their models, from service providers to electronic retailers. The PI has the potential of attracting new business models and to have a similar impact, for instance with innovative IoT solutions.

13. **Enable open infrastructural innovation**. As an example, we could imagine linking two major cities with a green-energy low-noise very-high-speed container train. The PI would ease the technical design and engineering of the entire infrastructure.

[7] discussed several means of integrating shippers and π-containers within all the PI infrastructure, making special remark to the way in which they should be interconnected and where the decision making should be done:

- "*The smart and connected nature of π-containers would enable decisions to be taken on the spot, given new current information on opportunities and constraints. They would decide on their routing dynamically, adapting their plans in route. They would call back to the shipper or his representative human or virtual logistic agent only in cases of out-of-bound situations where special circumstances make it forecast an improbable arrival on time and on budget, or when their physical or informational integrity and security are in danger*".

- "*Another option leaves minimal decision-making to the π-container, which simply relays information to an agent that takes the decisions in its place, and transmits it to both the π-container and, when appropriate, the physical internet elements involved in the route. The agent either takes the routing decisions on a one-by-one basis or considers a number of π-containers under his/its control. The π-containers and local π-elements* *only take initiative in cases of agent unavailability or incapability to respond in time to urgent decisional need*".

- "*In another option nearer to current ways of doing, the shippers or their logistic agents are securing complete routes prior to departure. As an alternative way, they may impose a set of key intermediary nodes and/or links, leaving the rest to more autonomous decision-making*".

The author called for further research on all topics we've presented, that is, on every characteristic of the PI. "*There needs to be creative design and engineering projects; analytical studies; simulation and serious gaming-based projects; pilot, prototyping, and demonstration projects; as well as optimization studies for decision-making within the new paradigm*". And he also suggested to focus on "*specific application areas such as containers, handling systems, ports, hubs, and so on*". In our case we have focused on the distribution and mobility layers/webs of the presented Logistics web, which somehow could be understood as an optimization study on the distribution routing problem related to the PI.

2.2 Combinatorial optimization problems

Optimization problems can naturally be divided into two different groups according to the type of variables that we are optimizing: *i*) those with continuous variables; and *ii*) those with discrete variables (which are called *combinatorial*). In continuous problems we are generally looking for a set of real numbers or a function, whereas in combinatorial problems, we search for an object from a finite (or countably infinite) set, *e.g.*, an integer set, permutation, graph, etc. These two types of problems are quite different in essence, thus their solving methods are truly divergent [28].

Combinatorial optimization problems (COPs) are a specific type of problems consisting in finding extrema of an objective function on a combinatorial space [29], in other words, it is aimed to to find the global optimal solution among all of them, i.e., the one that optimizes the value (global optimum) of the objective function according to a possibly existent set of constraints. For instance, the traveling salesman problem (TSP), the vehicle routing problem (VRP), the permutation flowshop scheduling problem (PFSP), they are all COPs. If interested, [30] gives a formal mathematical definition of COPs that is beyond the scope of this study.

COPs are usually too complex to be solved using exact methods if the problem is large enough. These methods can guarantee to obtain the global optimal solution, but they are impractical in terms of the computational time needed. So generally speaking, approximate methods are quite the fit in order to solve medium to large-sized problems in a reasonable computational time.

Furthermore, in the operational research literature (OR) it is quite common to define artificial scenarios, deterministic in their nature with a fixed number of static inputs, simplified from reality and modeled to be optimized the usual way. However these scenarios are far from being an accurate representation of real life problems. Randomness might easily arise in many parameters of the problem. Therefore, it is necessary to introduce different variants of COPs such as stochastic COPs (SCOPs), fuzzy COPs (FCOPs), robust COPs (RCOPs) or COPs with dynamic inputs (COPDIs). For instance, in the stochastic version of the vehicle routing problem (VRPSD), customers' demands are stochastic, following a given probability distribution, and cannot be revealed until the vehicle reaches the customer [31]. Other stochastic behaviors can be seen in several more problems.

There are many ways of classifying algorithms and it is always hard to end up with a single taxonomy. The classical way to classify algorithms is the way we learn them, *i.e.*, by its design paradigm. There are many educational and reference books [32, 33, 34, 35] which separate them in the following way: sorting algorithms, graph-related algorithms, greedy algorithms, divide and conquer, dynamic programming, linear programming, network flow, approximation algorithms, parameterization algorithms, local search, randomized algorithms and algorithms in computational geometry.

[36] summarizes the methodologies and algorithms used to solve the vehicle routing problem (VRP) [37]. In their survey, exact and approximate methods are discussed with respect to the class of problems so called rich vehicle routing problems (RVRPs), which are different variants of complex constrained VRPs. A classification of optimization methods is provided in this paper, which we have adapted in Figure 2.3.

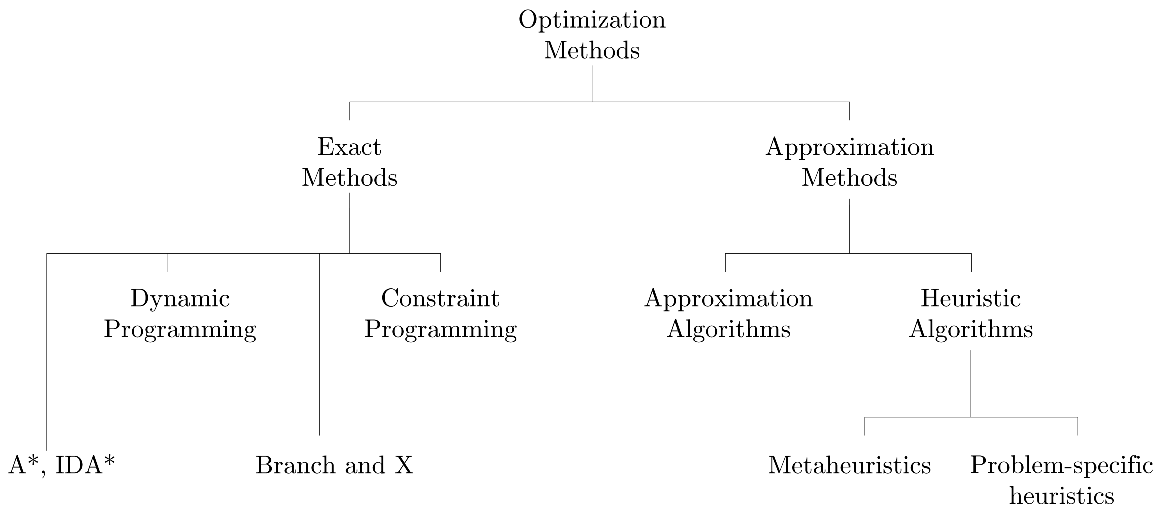


FIGURE 2.3: Classification of classical optimization methods.

According to [38], roughly speaking, the methods employed to solve COPs are mainly divided into two categories: deterministic and probabilistic algorithms. Deterministic algorithms give always the same solution for a given input, i.e., in each execution step it exists exactly one way to proceed. Whereas, probabilistic algorithms, also called randomized algorithms, have a behavior determined not only by its input but also by values produced by a random-number generator [33], i.e., they internally behave randomly. As [32] states, "*Efficient deterministic algorithms that always yield the correct answer are a special case of efficient randomized algorithms that only need to yield the correct answer with high probability*".

A rough taxonomy of global optimization methods, extracted from [38], is shown in Figure 2.4. As mentioned before, the author distinguishes between deterministic and probabilistic approaches.

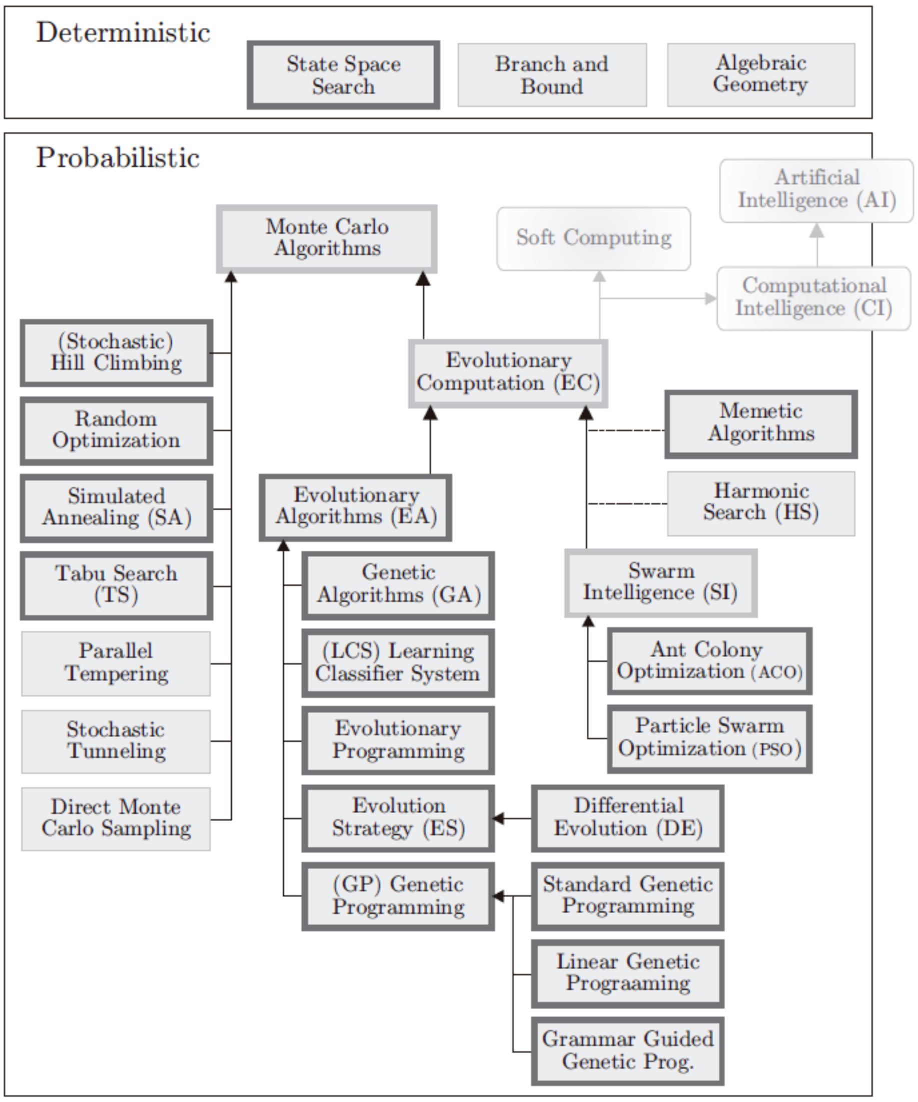

Deterministic
State Space Search
Branch and Bound
Algebraic Geometry
Probabilistic
Monte Carlo Algorithms
Soft Computing
Artificial Intelligence (AI)
Computational Intelligence (CI)
(Stochastic) Hill Climbing
Random Optimization
Simulated Annealing (SA)
Tabu Search (TS)
Parallel Tempering
Stochastic Tunneling
Direct Monte Carlo Sampling
Evolutionary Computation (EC)
Memetic Algorithms
Harmonic Search (HS)
Evolutionary Algorithms (EA)
Swarm Intelligence (SI)
Genetic Algorithms (GA)
(LCS) Learning Classifier System
Evolutionary Programming
Evolution Strategy (ES)
(GP) Genetic Programming
Ant Colony Optimization (ACO)
Particle Swarm Optimization (PSO)
Differential Evolution (DE)
Standard Genetic Programming
Linear Genetic Prograaming
Grammar Guided Genetic Prog.

Figure 2.4: Classification of global optimization methods.

As [34] states, "*Optimization problems have been not classified in a satisfactory way within the theory of P and NP; it is these problems that motivate the immediate extensions of this theory beyond NP*".

2.3 Optimization methods

In this section we are going to explain the optimization methods used for our problem in more detail. Section 2.2 widely described what COPs are and mentioned some of the most known techniques to solve them, now we are going to focus in heuristics, metaheuristics and precisely to the variable neighborhood search (VNS) metaheuristic.

2.3.1 Heuristics

"*Heuristics find good solutions on large-size problem instances. They allow to obtain acceptable performance at acceptable costs in a wide range of problems. In general, heuristics do not have an approximation guarantee on the obtained solutions.*" [39]. Approximation algorithms provide provable solution costs (or qualities) and computational times bounds, unlike heuristics. An (approximate) heuristic is ϵ-approximate if, intuitively, the relative error of the solution found is at most ϵ with respect to the optimal [34]. Anyhow, problem-specific heuristics are widely used to solve COPs, as they are fast simple problem-dependent techniques that lead to the design of deterministic algorithms. On the contrary, they cannot be seen as a black box that can solve several problems, as their characteristics are problem-dependent, they are tailored and designed to solve a specific problem and/or instance. The heuristic concept in solving optimization problems was first introduced by Polya in 1945 [40].

2.3.2 Metaheuristics

Metaheuristics are one of the most used techniques to reach near-optimal solutions in relatively short time [36]. They are advanced problem-independent procedures that lead to the design of randomized algorithms, i.e., algorithms that give different solutions for the same input. Somehow they can be seen as a black box that can be used to solve many different problems. Metaheuristics serve three main purposes: solve problems faster, solve larger problems and obtain robust algorithms. However, there is no guarantee to find global optimal solutions, nor there's any bound on the quality of the solution obtained. Nonetheless, near-optimal solutions are usually obtained, and expected to be found.

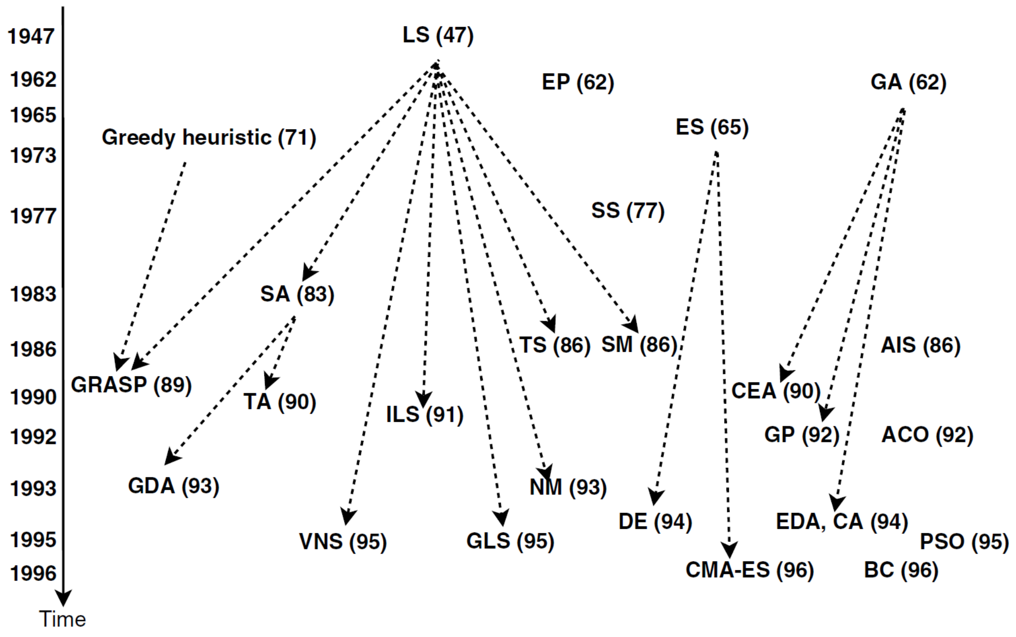


FIGURE 2.5: Genealogy of metaheuristics.

Figure 2.5 shows the genealogy of metaheuristics, extracted from [39]. The most popular metaheuristics according to the number of Google Scholar indexed articles from 2006 to 2015 are: ant colony optimization (ACO), artificial immune systems (AIS), greedy randomized adaptive search procedure (GRASP), iterated local search (ILS), genetic algorithms (GA), particle swarm optimization (PSO), simulated annealing (SA), scatter search (SS), tabu search (TS), variable neighborhood search (VNS). Figure 2.6, extracted from [41], classifies them according to whether they are: *i*) single solution-based (SMs) or population-based (PMs); *ii*) use memory; and *iii*) are nature-inspired. The bigger the circumference in the figure, the greater the number of articles found in Google Scholar. A wider classification of metaheuristics is shown in Figure 2.7.

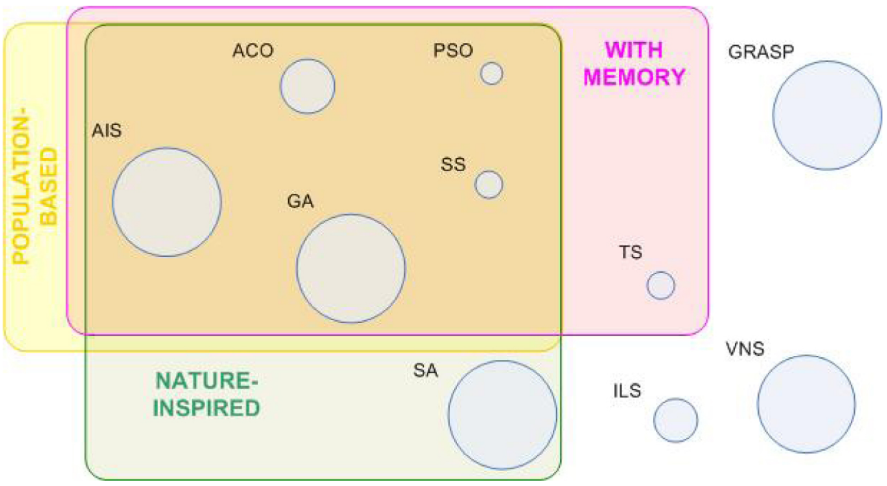


FIGURE 2.6: Classification of most known metaheuristics.

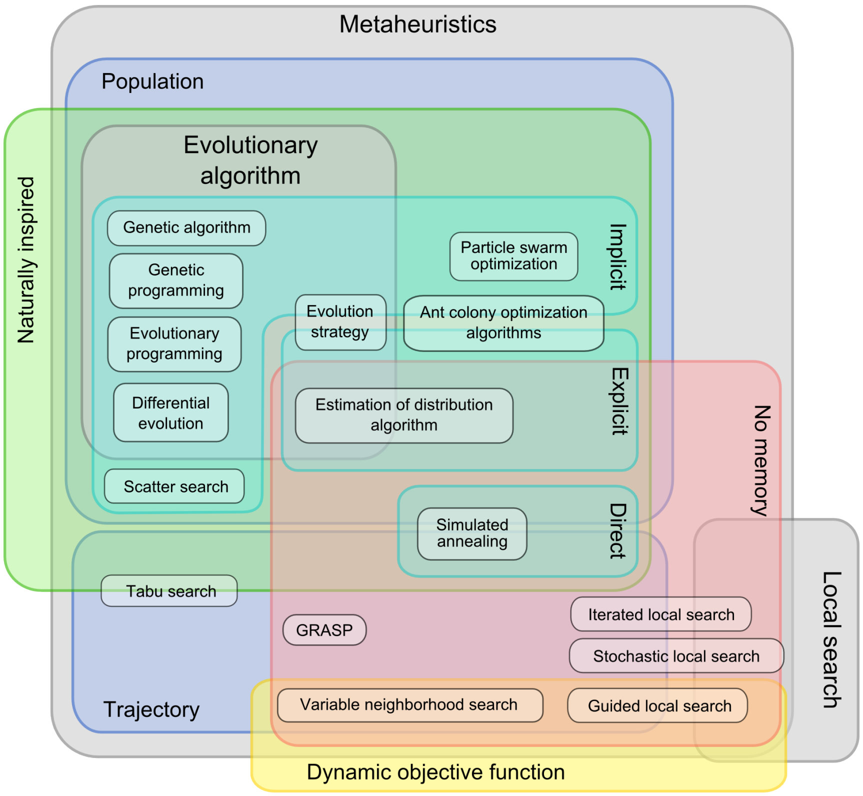


FIGURE 2.7: Metaheuristics grouped by different criteria.

2.3.3 The variable neighborhood search metaheuristic (VNS)

The variable neighborhood search is a single-solution based metaheuristic, memoryless, *i.e.*, no information dynamically extracted is used during the search, and not nature-inspired. It was introduced by P. Hansen and N. Mladenovic in 1997 [42]. The main idea of this metaheuristic, and the reason why it is widely used still nowadays, is that it lets escape from local optima exploring successively or at random different neighborhoods structures. This algorithm exploits the fact that using various neighborhoods in local search may generate different local optima, as different neighborhoods generate different landscapes, and that the global optima is a local optima for a given neighborhood (see Figure 2.8).

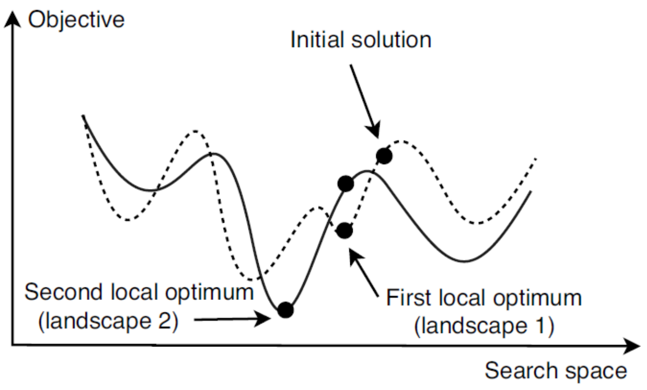

Objective
Initial solution
Second local optimum (landscape 2)
First local optimum (landscape 1)
Search space

FIGURE 2.8: Variable neighborhoods search using two neighborhoods.

The general variable neighborhood search algorithm iterates through a set of neighborhood structures N_k $(k = 1, ..., n)$. Pseudo-code 1 illustrates its main idea, adapted from [39].

Algorithm 1 Basic variable neighborhood search algorithm.

```
1: procedure VNS(neighborhoods)
2:     % neighborhoods: $N_k$ for $k = 1, ..., k_{max}$
3:     $x \leftarrow x_0$                                    ▷ Initial solution
4:     while stopping criteria not met do
5:         $k \leftarrow 1$
6:         repeat
7:             $x' \leftarrow shaking(N_k(x))$       ▷ Pick random solution from $N_k(x)$
8:             $x'' \leftarrow localsearch(x')$
9:             if $f(x'') < f(x')$ then                  ▷ Improvement
10:                $x \leftarrow x''$
11:                $k \leftarrow 1$                      ▷ Continue with $N_1(x)$
12:            else $k \leftarrow k + 1$                 ▷ Continue with $N_{k+1}(x)$
13:            end if
14:        until $k = k_{max}$
15:    end while
16:    return best solution found
17: end procedure
```

This algorithm performs three steps at each iteration: *i*) shaking; *ii*) local search; and *iii*) move from neighborhood. The local search is intended to be a fast small variation from the previous solution obtained, so that the algorithm doesn't get stuck there for long computational times. The stopping criteria might differ depending on the problem we are dealing with, although a limit in the computational time or number of iterations is widely used. Many variations from this general scheme of the algorithm can be generated, *e.g.*, considering intermediate worse solutions than the current we have, modifying the *shaking* step to pick good-quality solutions at first.

2.3.4 Biased randomization of heuristics (BR)

Heuristic methods generally follow an iterative process to build a solution for a given problem. Thus, at each iteration, the next step, decision or constructive movement is made from a list of different possibilities. This selection from a list of potential candidates can be done in many ways. A totally uninformed heuristic would randomly choose the next step, hence it wouldn't be much of a help. A greedy heuristic would sort the list of candidates in order to choose the best one, *i.e.*, it would always perform the best step in the sort run. This last kind of heuristic has a huge disadvantage, which is its implicit determinism, as the same choices will be made again and again for a given input, thus it will always obtain the same solution. And also seems quite inconvenient not to look farther than the current iteration, *i.e.*, not taking into account the subsequent possibilities once a decision is made.

To escape from local optima it is necessary to add a source of randomness in this construction phase of the algorithm. Although, we don't want to take this decision completely random (uniform distribution), nor imitate a greedy behavior by giving zero probabilities to all candidates except for the first one. Instead we will bias the selection giving stronger probabilities to better *a priori* candidates. This technique is called biased randomization (BR) and the methods to bias the randomization are called biased-randomized procedures (BRPs) [43]. Figure 2.9 illustrates the main idea of biased randomization techniques, which was extracted from [44].

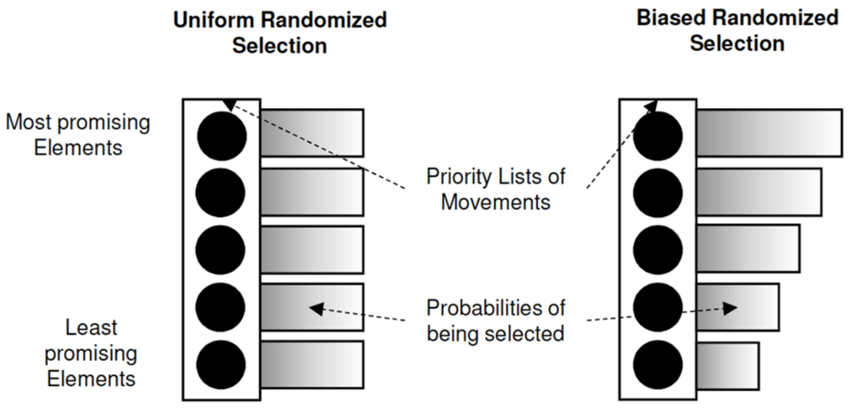

Uniform Randomized Selection
Biased Randomized Selection
Most promising Elements
Least promising Elements
Priority Lists of Movements
Probabilities of being selected

FIGURE 2.9: Uniform randomization and biased randomization.

Randomization can also be included while exploring a neighborhood in a local search method. However, for the case of our problem, in the context of the PI, we are using a specific type of procedures called BRPs with skewed theoretical probability distributions (STPDs), which doesn't require to empirically build a probability distribution according to an heuristic or some kind of ranking. Exmaples of such STPDs which are non-symmetric by definition are the geometric, the triangular or the log-normal. Figure 2.10 shows the difference on the probabilities for each candidate for the geometric and the triangular distributions, extracted from [43].

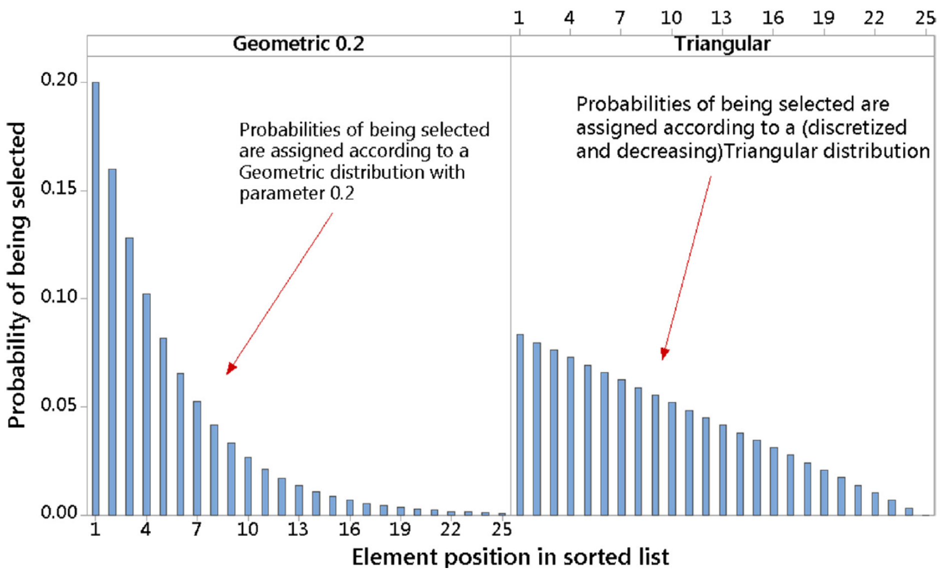


FIGURE 2.10: Geometric and triangular STPDs for BRPs.

2.3.5 A multi-start framework (MS)

Randomized heuristics, and in general, any type of randomized or stochastic algorithms can easily be embedded into a multi-start framework (MS) [45]. The basic idea under these techniques is quite logic in fact, as the main strategy consists in create numerous solutions with a randomized algorithm until a stopping criteria is met, then the best solution is returned. Pseudo-code 2 summarizes the main scheme of a multi-start algorithm.

Algorithm 2 Basic multi-start algorithm.

```
1: procedure MULTISTART
2:     $x \leftarrow x_0$                                  ▷ Initial solution
3:     while stopping criteria not met do
4:         $x' \leftarrow buildSolution()$                 ▷ Randomized heuristic
5:         if $f(x') < f(x)$ then                          ▷ Improvement
6:             $x \leftarrow x'$
7:         end if
8:     end while
9:     return x
10: end procedure
```

There's always a trade off between diversification (structural variation) and intensification (solution improvement) when searching for a near-optimal solution, no matter what kind of problem we are tackling. In this case, the multi-start algorithm tries to diversify this search, however it is done in an uninformed way which might seem naive. Despite that fact, these methods obtain high-quality solutions for some complex problems.

2.3.6 The geometric distribution

The geometric distribution is the probability distribution of the number k of Bernouilli trials needed to get a success. Given p the probability of success and q, of failure we have the following two distribution functions:

Probability distribution function (**PDF**).
Probability that the kth trial (out of k trials) is the first success.

$$P(X = k) = p_x(k) = q \cdot q \cdot ...^{(k-1)} \cdot q \cdot p = q^{k-1}p = (1-p)^{k-1}p$$

Cumulative distribution function (**CDF**).
Probability that the first success is before or at the kth trial.

$$P(X \leq k) = F_x(k) = \textstyle\sum_{i=1}^{k}(1-p)^{i-1}p = p(1 + (1-p) + (1-p)^2 + ... + (1-p)^{k-1})$$

Now let's see how to make the candidate's selection real fast when implemented in Java.

$$S_k = 1 + (1-p) + (1-p)^2 + ... + (1-p)^{k-1} = 1\frac{(1-p)^k - 1}{(1-p) - 1} = \frac{1-(1-p)^k}{p}$$

$$\text{Given that } S_k = 1\frac{a_{k+1} - a_1}{r-1} = \frac{a_1 r^k - a_1}{r-1} = a_1\frac{r^k - 1}{r-1} \text{ (geometric progression)}$$

$$F_x(k) = p\, S_k = 1 - (1-p)^k, \quad \text{where } F_x(k) \in [0,1]$$

$$(1-p)^k = 1 - F_x(k) \in [0,1]$$

$$k \log(1-p) = \log(1 - F_x(k))$$

$$k = \frac{\log(1 - F_x(k))}{\log(1-p)}, \text{ where p stands for our } \beta \text{ parameter.}$$

With this formula we are able to rapidly select the next candidate by generating one pseudo-random number and performing one subtraction, two logarithms, one division and a modulo operation.

Chapter 3

Literature review

This chapter is structured as follows: first part will review the idea of interconnected logistics network supported by the physical internet concept and IoT solutions, whereas the second part will discuss related articles to the proposed container distribution system.

3.1 Physical internet

[7] claims that the current world-wide logistics processes are inefficient by reviewing thirteen unsustainability symptoms of the way physical objects are transported, handled, stored, realized, supplied, and used. For instance, the issue of driver shortage where one of the reasons is low job satisfaction due to long periods of time in which drivers have to be away from home. Taking into account the current trends, driver shortage in the U.S. may balloon to almost 175,000 by 2024 [46]. The proposed optimization model aims to solve the problem by introducing work-life balance friendly working schedules for the drivers.

In order to deal with the logistics sustainability grand challenge, [7] proposed to strive towards the PI where logistics networks would be interconnected and goods would be encapsulated in world-standardized, green, networked, and smart containers that can be distributed across fast, reliable and eco-friendly transportation systems. Explicit state-of-the-art on PI has been gathered by [47].

[25] reviews recent research papers on smart containers and explains various projects which aim to achieve different levels of container intelligence in their communication with supply chain management systems by using IoT solutions.

IoT comprises many interconnected technologies like radio frequency identification (RFID) and wireless sensor and actor networks (WSAN) in order to exchange information between objects, systems and its users. While there is wide range of literature available, [26] provides essence of the concept, architectural elements and future developments, whereas [48] tackles challenges of IoT implementation, data storage, analytics and security.

It is clear that RFID and WSAN prevalence in logistics systems is increasing. Using real-time data acquired from responsive containers is also a future objective of the proposed optimization model. Dynamic traffic information will affect freight movement, allow better planning and improved scheduling [26]. Online monitoring of origin-destination routes and container travel times would enable to make efficient decisions depending on the state of the system, *i.e.*, to calculate routes and assign the available drivers to containers according to priority as the containers enter and/or progress within the distribution network.

3.2 Container distribution model

Up to now, a tremendous attention has been given to optimization of diverse logistics networks. There is a wide range of literature available on optimization in different areas, such as road logistics, marine networks, or cargo industry in general. Just to point out some recent examples, [49] builds a model for multi-size inland container transportation problem which aims to minimize total travel distance and operation time of the trucks. [50] offers modeling and an optimization model for a multi-echelon container supply chain, whereas [51] optimizes multi-modal transportation by combining routes.

In the context of hub-and-spoke systems, [52] optimizes a port logistics network of dynamic hinterland, and [53] studies routing optimization for multi-type containerships with time deadline to minimize service, waiting and traveling cost. [54] combine two hub-and-spoke networks into a regional port cluster. [55] in their work concentrate on decision-making, calculating cost savings of developing a basic hub-and-spoke or a hybrid version and applying the model to the Australian parcel service provider, whereas [56] along with routing decisions, considers inventory levels in pursuit of minimizing shipping and inventory costs.

As this thesis proposes an innovative distribution approach, related literature is very limited. However, combinatorial problems in transportation and logistics are quite often particular cases and variations of a network design problem [57], where some sort of discrete choice with regard to the network structure is involved. Seems relevant to the problem at issue to consider the multicommodity network design problem (MCND) [58], as it has many applications in transportation planning. This problem is naturally related to flow routing so that certain commodity demands are satisfied, but also to network design as several facilities can be installed in different arcs. With respect to the problem we are dealing with, the similarities reside in the container routing part, without taking into account driver assignments, as containers can be understood as integer commodities that need to be routed with specific origin and destination nodes. In essence, there's an implicit design problem that has a direct effect on the operations of the transportation network.

This scenario was most likely introduced by [7] in order to switch from the traditional point A to point B delivery of drivers going on a multi-day journey and possibly returning empty handed, to a PI enabled model where the driver takes load to a hub two to six hours away and then returns home with another trailer. A simple case study of the route from Quebec to Los Angeles suggests that instead of the 240-hour journey made by a single driver, a load would reach the destination in 120 hours with the help of 17 drivers who would get home the same day.

While the method speeds up the delivery and facilitates short-haul truck driving, which makes the system socially responsible, it is important to consider trade-offs. [59] compares the current logistics process performance of France road network with the PI enabled distribution introducing hubs and then implementing multi-modal transportation in order to reduce CO_2

emissions. Results claim that the PI concept could substantially improve logistics efficiency and sustainability, *e.g.*, reducing carbon footprint by 60% without jeopardizing operational costs or accommodating lead times.

[27] confirms that the PI distribution approach in comparison with conventional methods reduces driving distance, gas emissions and social burden of traffic using Monte-Carlo simulation (MCS) in an Eastern Canada road network case study. It also verifies that the number of drivers who return home at the same day stays comparatively high despite traffic intensity.

While the forehand articles mentioned employ some of the PI distribution ideas, the mechanism of our model is different. Multiple drivers could be assigned to a container for each edge of the path according to the priority and deadline of the container, and spare working hours of the available drivers. Re-scheduling would occur at each hub, thus ensuring efficient real-time decision-making. This model is also flexible in terms of routing optimization as there are no driver-container pairs assigned which have to start and return at home-base together as in [27]; paths of containers and drivers are rather calculated independently.

Chapter 4

Problem statement

The spoke-hub network problem studied can be defined as a network $N = (L \cup H \cup D,\, E)$, where the set of nodes $L \cup H \cup D$ respectively represents container origin and destination locations, also called endpoint nodes, hubs/facilities, and drivers' settlements/depots; and the set of edges E represents links connecting these nodes. Each edge $e \in E$ is characterized by a traveling time t. Notice that the graph underneath this network N is weighted, undirected and not necessarily complete.

A set of containers C has to be transported through this network. Each container $c \in C$ has an associated origin and destination locations, l_o, $l_d \in L$ and a due time d. Containers can be dropped temporarily at hubs on their way from origin to destination. Depots and endpoints are passable by drivers, but only the latter can store containers if they are their origin or destination.

A limited number of drivers start and end their working shift at their associated depots. Notice that a driver is only able to drive one single truck and can't change it during his shift. Somehow, trucks and drivers are interchangeable concepts in this problem. Also notice that we are only considering that a driver is able to perform one shift. It is assumed that hubs are incapacitated. The goal is to minimize the cost of the solution, which from now on will primarily be the total time required to deliver all containers and to return all drivers to their depots, subject to: (i) each truck can load one container at a time; (ii) all containers must reach their destination on or before the corresponding due time; and (iii) there is a maximum number of working hours per driver.

Figure 4.1 shows a representation for a small network with its weights, figure 4.2 shows a feasible solution for a problem with 2 containers and 3 drivers and figure 4.3 shows a slightly different solution for the same problem with a timescale. Different representations have been used for the last two figures in order to remark the routes in one and the timings in the other.

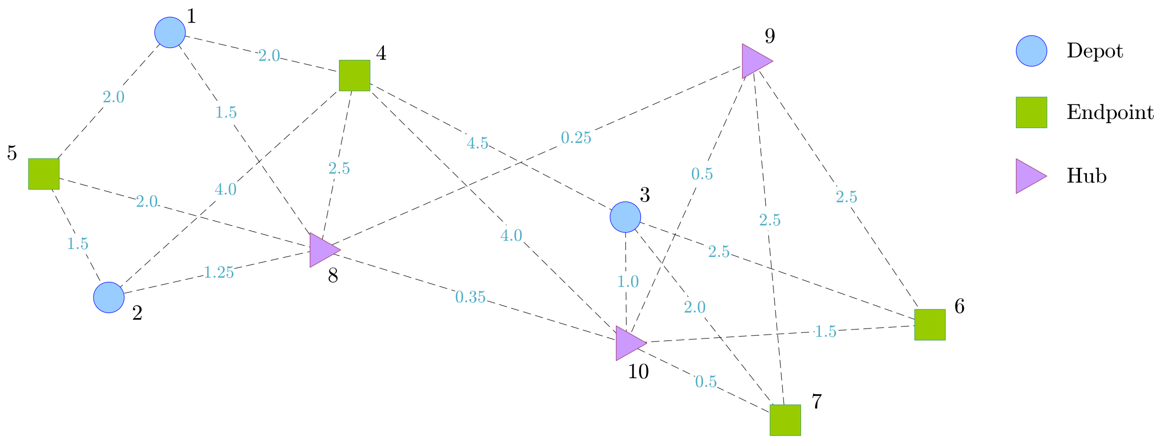


FIGURE 4.1: Representation of the container transportation network.

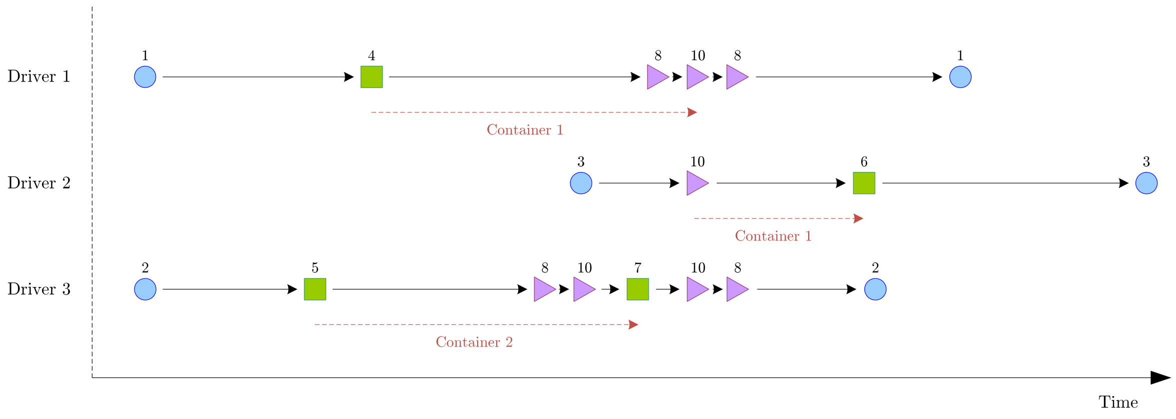


FIGURE 4.2: Representation of a solution for the problem.

It is easy to see in figure 4.2 that the second driver is anticipating the need to transport the first container reaching the node number 10. This behavior seems difficult to be simulated by an algorithm, nevertheless with small instances is easy to be grasped by a human eye while solving the problem manually with a paper and a pencil.

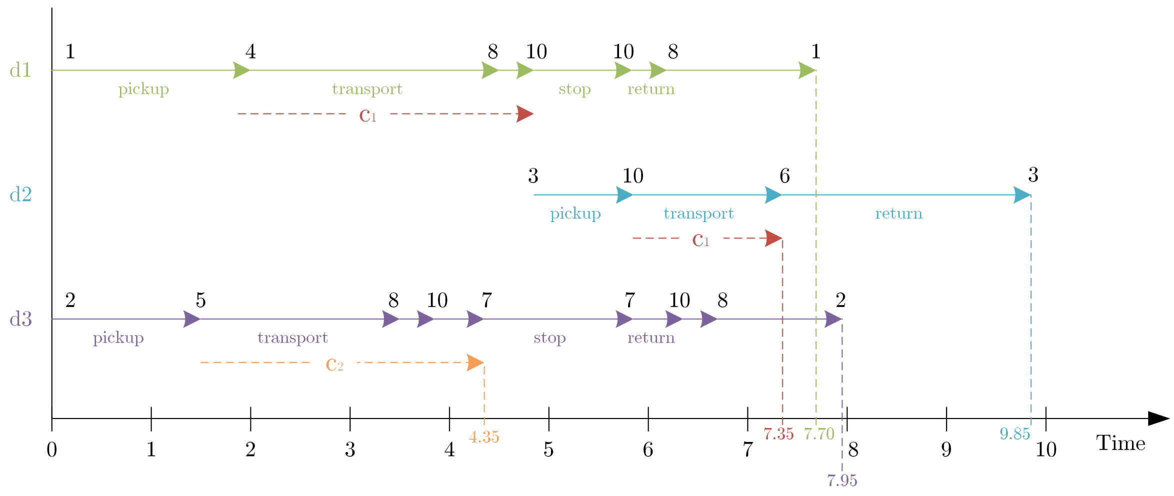


Figure 4.3: Representation of a slightly different solution with timescale.

Figure 4.3 also shows the end time for containers and drivers as well as the different actions each driver is doing alongside his route: pickups, transports, stops or returns. In this solution, though, the second driver is activated once the first has arrived to the node number 10 and has realized that can't keep transporting the first container because otherwise he wouldn't arrive to his depot on time. Thus, the first container is left at that node with the need of an idle driver to transport it towards its destination.

Chapter 5

Methodology

5.1 Solving approaches

This section describes three approaches to solve the aforementioned problem. First, a heuristic relying on a discrete-event list is presented, which makes deterministic decisions based on the best selections in the short run related to containers, drivers and paths. Later, a multi-start framework is proposed, which is based on the combination of the previous heuristic and biased randomization techniques. While this last approach is going to take more computational time, it is expected to provide higher-quality solutions. Finally, a VNS is designed to widely explore the search space in a more efficient manner.

5.1.1 Heuristic

Because the underlying methodology of the heuristic proposed is based on discrete events it is important to understand that, as it is the case in simulation, an event is triggered once a decision taken in a previous moment of time is finished. For instance, we can decide that certain driver is going to transport a container from his current location to the next node in the shortest path towards the container's destination. Then an event would be triggered right when the transport ends. This event would change the state of the system, thus new decisions would be made upon it, and more events would be scheduled and appropriately triggered. However, we need to store all the relevant information related to an event, *i.e.*, the driver involved, the origin and end nodes, the time when was triggered and the container being transported, if any. Even though an event is discrete in the sense that is triggered in a precise moment and does not continuously happen in time, an starting time is linked to it, as there was a moment when the decision was taken. So its starting time is implicitly set to the time when it was scheduled, and its length is defined by the traveling time between the origin and end nodes. We have distinguished 3 different types of events related to different decisions and actions: pickups, transports and returns.

- Pickups consist in moving an empty driver towards an unattended container. However, it is not implied that the driver will transport that container afterwards, but it is likely to happen. These transports can be multi-hop trips, *i.e.*, more than one node could be traversed without divesting the container of the driver until the latter arrives to the container's location.

- Transports are always between hubs and/or endpoints and consist in the transportation of a container from its current node to the next one towards its destination. They can also be multi-hop trips if the next node is a depot or an endpoint different from its destination.
- Returns, on the other hand, are always 1-hop trips (node-to-node) and they are only scheduled when necessary, *i.e.*, when a driver needs to get back to its depot so that its working shift isn't exceeded.

Our heuristic is described in detail below, summarized in Figure 5.1 and outlined in Pseudocode 3:

1. Compute the shortest path between any pair of nodes in the network using the Floyd-Warshall algorithm [60].
2. Make an initial assignment of drivers to containers so that all containers are attended, or at least, as many as available (feasible) drivers. These assignments are made the same way we make new decisions and schedule events: iteratively assigning the most urgent container to its nearest driver. The first set of events generated will be added to an event list increasingly sorted by the end time of the event, *i.e.*, the time when the event will be triggered. So far we have only scheduled pickups.
3. The following 4 - 7 steps are iteratively executed until all events have been processed, *i.e.*, until all drivers have arrived to their depot and a solution is found (feasible or not).
4. At each iteration the first event is triggered and removed from the event list. The current time is set to the time of this event, as if we were advancing the time from the last processed event. However, we trigger as many events as actions end at the same time, because there might be many events to be triggered at the same time, as there can be collisions. When we process an event the container is divested of the driver and its related action is finished, whereas when we schedule an event its related action starts to happen in the current time. Thus, we are not scheduling actions that will start to happen in the future.
5. Update the system with the arrival of the driver(s) and container(s), if any. Here is where the distinction of events is needed, as different actions have different consequences.
6. Schedule container-related events, step summarized in Figure 5.2 and detailed in Pseudocode 4, following these steps:

   6.1. Sort unattended containers by urgency, *i.e.*, by the difference between the actual traveling time of the container towards its destination and the remaining time of its deadline.

   6.2. For each container in that order, the following steps are taken: *i*) sort idle drivers by proximity; *ii*) select the next node of the container using the shortest path;

iii) select the nearest feasible driver; and *iv*) schedule the event, which will be appropriately added to the event list according to the sorting criteria mentioned.

If the selected driver is in a different node than the container, a pickup event will be scheduled, instead of a transport, and the destination node will be the one where the container resides.

A feasible driver will be able to: *i*) reach the container; *ii*) transport it to the next hub in the shortest path to its destination (or directly to its destination, if there wasn't any); and *iii*) return to his depot on time.

Idle drivers remain in the same node until they are assigned to a container or they need to return to their depot. Notice that a decision could be that the driver that just arrived leaves a non-urgent container at the current node and goes empty to pickup a more urgent container.

Also notice that some hops are mandatory, for instance, when a driver transporting a container passes through a depot or an endpoint that is not the container's destination, he has to keep going because he can't leave the container there. Therefore, all transport events are hub-to-hub, except for the first and last events of a delivered container, which involve endpoints.

7. Schedule driver-related return events, step summarized in Figure 5.3 and detailed in Pseudo-code 5, if necessary. For all idle drivers, if they need to start returning in the current time because they would not be on time returning on the next event, we schedule a return event.

   This way we ensure that working shift constraints will not ever be violated using this heuristic, but containers' might.

   Notice that once a driver is returning to his depot it can still transport a container in his way home, either because it's in his path, or because has enough margin to deviate from it.

   It is also important to remark that there is no driver finished state until he has reached his depot and ended his working shift. Once a driver that is returning arrives to his depot it could still be assigned to a container if he has margin to transport it and return to his depot on time.

8. Once all events in the list have been triggered and processed, *i.e.*, empty event list, we have a solution that, by construction, is likely to be feasible. Compute the associated cost, *i.e.*, the time of the last event, that is, the time when the last driver reached his depot, and check its feasibility.

This way we are trying to maximize the number of containers being attended at all time, while trying to minimize the traveling time of all drivers.

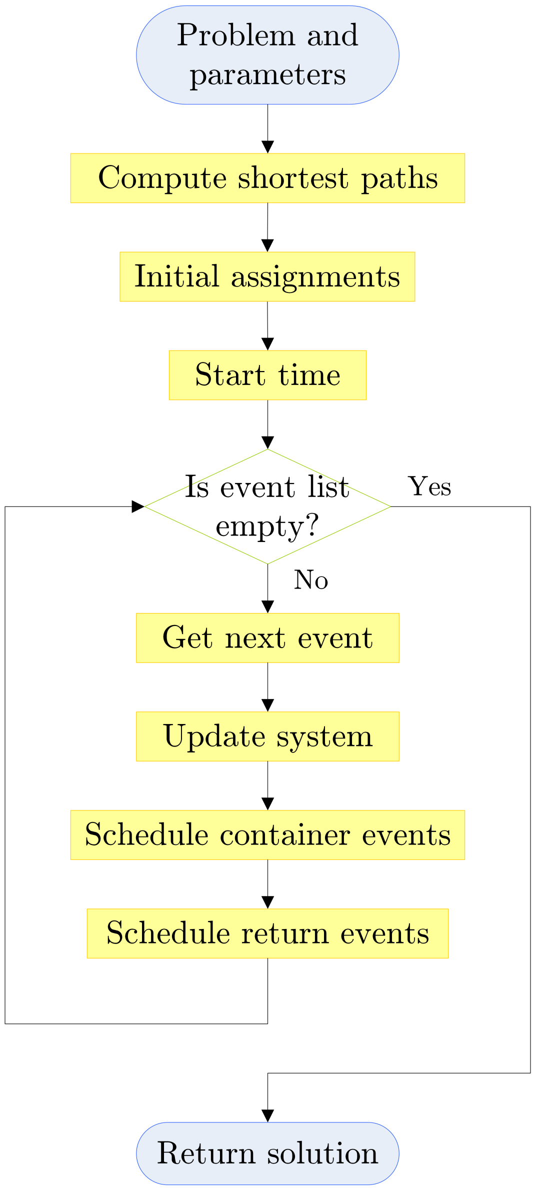


FIGURE 5.1: Scheme of the deterministic heuristic proposed.

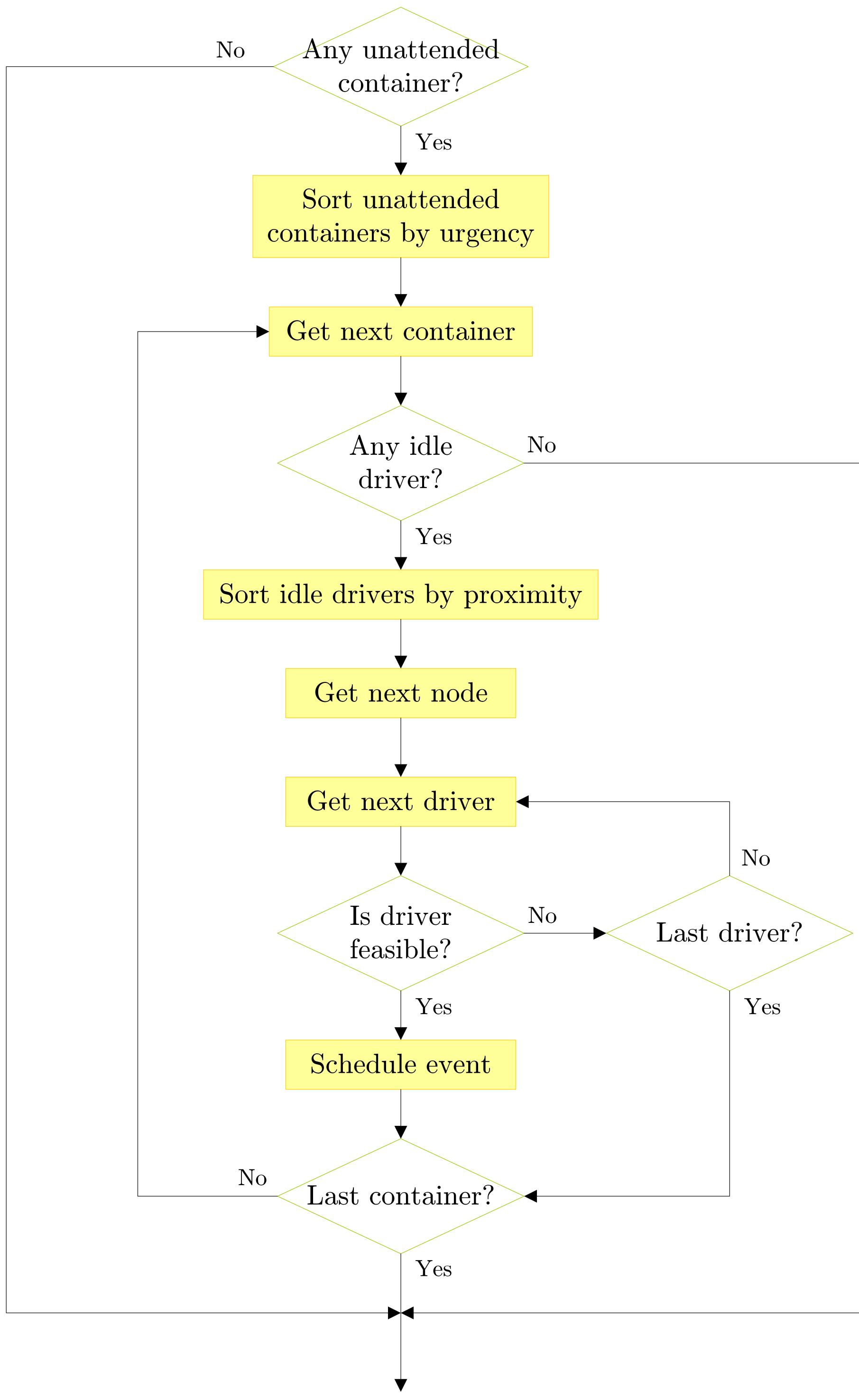


FIGURE 5.2: Scheme of the container-related events step.

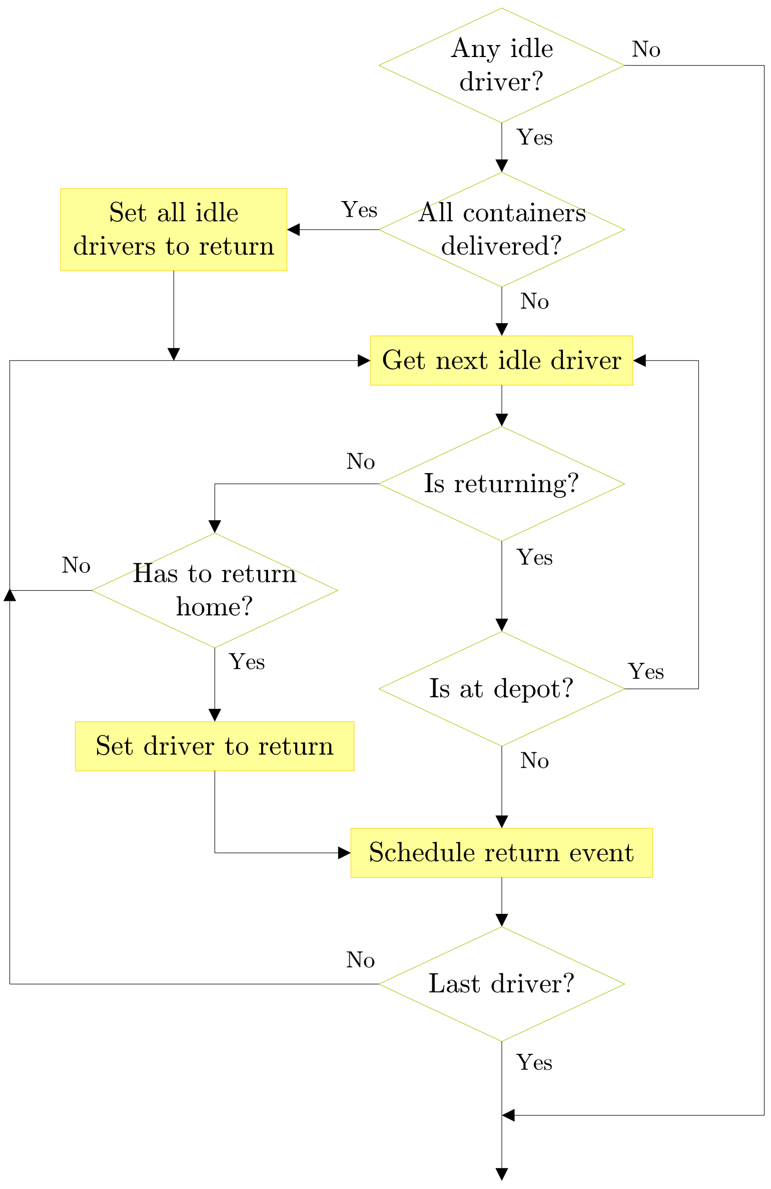


FIGURE 5.3: Scheme of the driver-related events step.

Algorithm 3 Main procedure of the proposed heuristic.

```
 1: procedure SOLVE(network, containers, drivers)
 2:
 3:     shortestPaths ← Floyd-Warshall(network)          ▷ Find shortest path between all nodes
 4:     solution ← emptySolution
 5:     eventList ← scheduleContainerEvents(containers, drivers, shortestPaths, solution)
 6:
 7:     while eventList is not empty do
 8:         event ← get next event from eventList
 9:         updateSystem(event, network, containers, drivers, solution)
10:         scheduleContainerEvents(containers, drivers, shortestPaths, solution)
11:         scheduleReturnEvents(drivers, shortestPaths, solution)
12:     end while
13:     return solution
14: end procedure
```

Algorithm 4 Schedule container events step of the heuristic.

```
 1: procedure SCHEDULECONTAINEREVENTS(containers, drivers, shortestPaths, solution)
 2:
 3:     unattendedContainers ← getUnattendedContainers(containers)
 4:     if unattendedContainers is not empty then
 5:         sort unattendedContainers                                  ▷ By urgency
 6:         idleDrivers ← getIdleDrivers(drivers)
 7:
 8:         for all container in unattendedContainers do
 9:             if idleDrivers is empty then exit for
10:             else
11:                 node ← getNextNode(container, shortestPaths)       ▷ By shortest path
12:                 sort idleDrivers                                   ▷ By distance to container
13:                 found ← false
14:                 k ← 0
15:                 while k < size of idleDrivers and not found do
16:                     if areFeasible(driver, container) then
17:                         scheduleEvent(driver, container, node)     ▷ Add event to eventList
18:                         update solution
19:                         remove driver from idleDrivers
20:                         found ← true
21:                     end if
22:                     k ← k + 1
23:                 end while
24:             end if
25:         end for
26:
27:     end if
28: end procedure
```

Algorithm 5 Schedule return events step of the heuristic.

```
procedure ScheduleReturnEvents(drivers, shortestPaths, solution)

    idleDrivers ← getIdleDrivers(drivers)
    if idleDrivers is not empty then

        if all containers delivered then                          ▷ Set all drivers to return
            for all driver in idleDrivers do
                if driver is not returning then
                    set driver to return
                end if
            end for
        end if

        for all driver in idleDrivers do
            if driver is returning then
                if driver is not at depot then                    ▷ Schedule next return event
                    scheduleReturnEvent(driver, shortestPaths)    ▷ Adds event to eventList
                    update solution
                end if
            else if driver has to return then                     ▷ Can't be on time in the next event
                set driver to return
                scheduleReturnEvent(driver, shortestPaths)
                update solution
            end if
        end for

    end if
end procedure
```

5.1.2 Biased-randomized multi-start

As noted before, the decisions of this heuristic are greedy, *i.e.*, the best choice in the short term is always chosen and the same solution is always obtained for a given instance.

In order to overcome this disadvantage and explore a wider search area, a biased-randomized heuristic (BRH) is built employing a multi-start framework (BR-MS) [45], detailed in Pseudo-code 6. In particular, each decision related to the selection of a container, path (or node) and driver is randomized by using a geometric probability distribution g_c, g_p, g_d with the respective β_c, β_p, β_d parameters. Numerous solutions are built until a stopping criteria based on the number of iterations is met and the best solution (feasible or not) is returned.

The scheme of the algorithm is exactly the same as the one for the heuristic. However, the way to schedule container events, shown in Pseudo-code 7, differ in: *i*) the order in which containers are iterated, not always the most urgent is attended first; *ii*) the next container's node in the path towards its destination, not always the shortest path is taken; and *iii*) the driver to transport/pickup the container, not always the closest one is chosen.

Algorithm 6 Main procedure of the proposed BR-MS.

```
procedure SolveBRMS(network, containers, drivers, $iter_{max}$, $\beta_{min}$, $\beta_{max}$)

    bestSol ← emptySol                                        ▷ Initialize bestSol
    $iter \leftarrow 0$

    while $iter < iter_{max}$ do
        $\beta_C \leftarrow random(\beta_{min}, \beta_{max})$            ▷ Parameter for container selection
        $\beta_P \leftarrow random(\beta_{min}, \beta_{max})$            ▷ Parameter for path selection
        $\beta_D \leftarrow random(\beta_{min}, \beta_{max})$            ▷ Parameter for driver selection
        newSol ← solveBRH(network, containers, drivers, $\beta_C$, $\beta_P$, $\beta_D$)
        if $totalTime$(newSol) < $totalTime$(bestSol) then
            bestSol ← newSol
        end if
        $iter \leftarrow iter + 1$
    end while

    return bestSol
end procedure
```

The path that each container will follow is not set in the beginning, but decided in each step of the algorithm choosing the next node they will visit. As mentioned before, transport events are hub to hub, so in order to select the next node (a hub or the container's destination) the following steps are taken:

1. Obtain a list of the adjacent hubs (or its destination) to the current container's node.

2. Sort this list by the traveling time from that hub to the container's destination following the shortest path and taking into account the edge from the container's current node to the hub.

3. Select the next node from the sorted list according to the geometric distribution.

The idea behind this approach is that the best option on the short run does not necessarily lead to the best global solution, not even to a feasible one.

Algorithm 7 Schedule container events step of the BRH.

```
procedure ScheduleContainerEventsBRH($\beta_c$, $\beta_p$, $\beta_d$, containers, drivers, shortestPaths)

    unattendedContainers ← getUnattendedContainers(containers)
    if unattendedContainers is not empty then
        sort unattendedContainers                                  ▷ by urgency
        idleDrivers ← getIdleDrivers(drivers)
        k ← 0

        while k < size of unattendedContainers and idleDrivers is not empty do
            container ← selectContainerBRH($\beta_c$, unattendedContainers)
            node ← selectPathBRH($\beta_p$, container, shortestPaths)
            driversList ← sort idleDrivers                         ▷ by distance
            found ← false
            while driversList is not empty and not found do
                driver ← selectDriverBRH($\beta_d$, driversList)
                if areFeasible(driver, container) then
                    scheduleEvent(driver, container, node)
                    remove driver from idleDrivers
                    remove container from unattendedContainers
                    found ← true
                else
                    remove driver from driversList
                end if
            end while
            k ← k + 1
        end while

    end if
end procedure
```

5.1.3 Biased-randomized variable neighborhood search

The biased-randomized multi-start approach randomizes the heuristic proposed and serves to explore different routing possibilities during the search for higher-quality solutions. Nonetheless, this search is uninformed in the sense that there's no heuristic being used to address the direction we should follow, *i.e.*, given a good solution as a starting point there's no path towards a better solution besides the construction of a new solution from square one. Therefore, a biased-randomized variable neighborhood search (BR-VNS) [61] is presented as a natural extension to the BR-MS.

The VNS metaheuristic aims to escape from local optima exploring systematically or at random different neighborhoods. In this specific problem, the exploration is done randomly without any predefined set of neighborhood structures. A neighborhood $N_p(x)$ is defined as the set of possible solutions that can be obtained from resolving a partial solution x_p, which is the result of undoing the last p% of decisions taken in the solution x. So the shaking step consists in *destroying* by some percentage p a base solution and *resolving* it once by using the randomized heuristic, *i.e.*, instead of picking a random solution from the current neighborhood, as it is usually done, a solution is built by bringing the base solution back into a past state, like some sort of backtracking, and then the partial solution is resolved. If the solution obtained is better than the base solution, the former becomes the new base solution and the percentage of destruction p is reset to a minimum percentage. Otherwise, p is incremented by a certain *step* and it goes on again. This algorithm is described in detail below and summarized in Pseudo-code 8:

1. Initialize the base solution using the deterministic heuristic. The randomized heuristic could also be used, but an unfeasible solution could be obtained and it wouldn't be a good starting base solution.

2. Set the current percentage of destruction p to the minimum.

3. While a stopping criteria based on the number of iterations is not met we proceed to iteratively try to improve our solution by the following process:

   - Shaking or construction/destruction phase: the base solution is rolled back to an intermediate state where the last p% of decisions are reversed. A partial solution is obtained with a specific current time and state of drivers and containers. This solution is then resolved by means of the BR heuristic.
   - If the new solution is better than the base solution, *i.e.*, if the cost function of the new solution is lower than the one of the base solution, the former becomes the latter and the percentage p is reset to the minimum. However, sometimes worse solutions become the base solution if an acceptance criterion is met, *e.g.*, the demon acceptance function [62].
   - Otherwise, p is incremented by *step*.

4. In the end, the best solution found is returned, which might vary from the base solution. The former is only updated if there's an improvement in the cost, whereas the latter is updated every time a new solution is accepted (better or worse).

This way we ensure that good intermediate solution states are thoroughly explored and more routing terminations are built on good base solutions.

Algorithm 8 Main procedure of the proposed BR-VNS algorithm.

```
procedure SolveBRVNS(network, containers, drivers, $iter_{max}$, $p_{min}$, $p_{max}$, $step$)
    % $p_{min}$: minimum percentage of solution destruction
    % $p_{max}$: maximum percentage of solution destruction
    % $step$: increment of percentage on each iteration

    baseSol ← SolveSH(network, containers, drivers)          ▷ Using the fast heuristic
    $demon \leftarrow 0$
    bestSol ← baseSol
    $p \leftarrow p_{min}$
    $iter \leftarrow 0$
    while $iter < iter_{max}$ do
        newSol ← shaking(p, baseSol)          ▷ Destruction/construction phase
        $\Delta \leftarrow totalTime$(newSol) $-$ $totalTime$(baseSol)
        if $\Delta < 0$ then          ▷ Improvement
            baseSol ← newSol
            $p \leftarrow p_{min} - step$
            $demon \leftarrow -\Delta$
            if $totalTime$(newSol) $<$ $totalTime$(bestSol) then
                bestSol ← newSol
            end if
        else
            if $\Delta < demon$ then          ▷ Worsening
                baseSol ← newSol
                $p \leftarrow p_{min} - step$
                $demon \leftarrow 0$
            end if
        end if
        $p \leftarrow min(p + step,\ p_{max})$          ▷ Visiting next neighborhood $N_p(baseSol)$
        $iter \leftarrow iter + 1$
    end while
    return bestSol
end procedure
```

5.2 Creation of standardized instances

Since there are not standardized benchmark instances in the literature for this problem we are introducing our own ones. However we are not creating artificially generated networks, but adapting the ones introduced by [63] for the Time Capacitated Arc Routing Problem that can be found in the *TCARP large rural datasets*. In these datasets we can find 5 different networks that represent a simplified version of Irish roads drawn from a Geographic Information System. Table 5.1 shows their main characteristics specifying no. of nodes, no. of edges, density (or sparsity) $d \in (0,1)$, and basic stats on the weights and the degree of the nodes. Notice that original weights represented distances and ranged from 1 to 1394 meters, but we have scaled them for each instance to a reasonable range for our problem and will be interpreted as traveling times in hours.

	Network			Weights			Degree	
Instance	Nodes	Edges	Density	Mean	Min	Max	Mean	Max
TCARP-R1	221	245	0.01	0.87	0.069	1.786	2.22	5
TCARP-R2	382	424	0.006	0.938	0.048	2.125	2.22	4
TCARP-R3	508	550	0.004	0.916	0.062	2.893	2.17	5
TCARP-R4	805	872	0.003	0.788	0.022	3.098	2.17	6
TCARP-R5	599	647	0.004	0.964	0.002	3.062	2.16	5

TABLE 5.1: Networks' characteristics summary.

For each network 4 different instances have been created with a different set of containers, fleet of drivers and structure of the network in terms of the assignment of each kind of node, *i.e*, we have specifically considered scenarios with 25, 50, 100 and 200 containers and the number of drivers per depot ranges in different upper and lower bounds, as well as their shifts. The set of containers has different origins, destinations and margins in their deadlines. But also each instance might have a different number of depots, hubs and endpoints, as they can be located in different nodes, despite the fact that the underlying graph is the same, *i.e.*, the same number of nodes, their coordinates and adjacency matrix.

So in order to adapt and create an instance for the PI container transportation problem the following steps have been taken:

- Make the node's assignment: each node in the network has been assigned to be a depot, hub or endpoint following a simple yet reasonable criteria according to its connectivity, *i.e.*, according to its degree d:

  1. If the node is a leaf ($d == 1$) it can either be an endpoint or a depot, because a hub wouldn't normally be useful there. So these nodes are endpoints with 0.5 probability or depots with the same probability.
  2. If the node has relatively high connectivity ($d >= 4$) it must be a hub so that other endpoints and depots can be connected through them. However we need to avoid big clusters of close hubs, so these nodes are hubs with 0.75 probability or depots with 0.25.

3. Otherwise, the node has a degree $2 <= d <= 3$ and we just try to allocate all types of nodes with different probabilities. A depot is assigned with 0.3 probability, an endpoint with 0.2 and a hub, 0.5. In this case we hope not to have isolated depots or not reachable endpoints by also adding many hubs. However these properties are hard to be preserved across the whole network for big instances using this simple assignment.

- Define the set of containers to be delivered: origin and destination endpoints are randomly chosen but taking into account that the shortest path between both nodes can't exceed 24h (for instances 1 to 10) and 12h, for the rest. Otherwise the solutions can take a total time up to 72h if the network is big enough. The due times for the containers are set to a multiple of the shortest path's length between their origin and destination endpoints. For instance, if the length of the shortest path between origin and destination is l, we normally set a margin of $k \cdot l$ hours, where k usually takes a value between 2 and 3, depending on the instance.
- Assign drivers to depots and set their shifts: there's no prior number of drivers known when creating an instance. Instead we assign a random number of drivers per depot within a lower and upper bound. The working shifts range from 8 to 10 hours, as the physical internet aims to reduce the maximum working hours per driver in a day shift, which right now can't exceed the 11-hour driving period. These shifts are randomly assigned to each driver. Notice that because the containers aren't equally distributed across the whole network, but randomly placed in different endpoints that might serve as origin and/or destination for more than a container, we definitely need to allocate many available drivers per depot even though most of them won't be moved.

As one could imagine, the way in which we are creating our instances doesn't guarantee at all to obtain feasible solutions out of them, because scenarios where a container isn't reachable during the simulation of the solution are likely to happen. So our approach has been to generate several hundreds of instances for each original network (1 to 5) and set of containers (25, 50, 100 and 200), solve them using the deterministic heuristic and choose the ones in which a feasible solution was obtained.

This manipulation of inputs and creation of new instances has been done using Python 3.6.3 and specific libraries such as NetworkX for graph manipulation or Matplotlib and Pylab for visualization purposes.

5.3 Verification and validation (V&V)

According to [64] the process of verification aims to test that an algorithm "*complies with the intent of its designers and developers*", that is, that our code does exactly what we expect it to do in the precise way we designed it to do it. In other words, "*the accuracy of transforming a created model into the computer program is tested in the verification phase*" [65]. In order to assert that this condition is preserved, the usual way of doing in OOP is to perform unit testing in methods and classes. This testing phase tries to cover the most common scenarios as well as some known particular and extreme cases to confirm that the algorithm behaves as expected and outputs the right results. From that point on, it has to be assumed that the algorithm has been verified, as we can't exhaustively test all inputs and cases. Complex algorithms are hard to be mathematically verified, thus they are usually tested using specific software frameworks such as JUnit, even though there's a lot of research on formal verification (see [66]). On the other hand, validation ensures that "*the built model is an accurate representation of the modeled phenomena to simulate*" [65], *i.e.*, in our case it would be the same as checking that the solution given by our algorithms is consistent and coherent with respect to the constraints we defined in our problem.

With regard to the verification of the algorithms developed, we have tested critical methods of our classes with JUnit and debugged most parts of the code. However, exhaustive unit testing hasn't been possible to be done in a project with this time frame, as more than a total of 6000 lines were written, despite it being a poor metric.

In respect of the validation process, once a solution has been obtained using any of the approaches explained in section 5.1, two different types of checks are made: correctness and consistency assertions (validation) to ensure that the algorithm is returning a valid solution, but also feasibility assertions in order to determine whether the solution is feasible or not. A solution will be considered feasible if the following constraints are met:

1. All containers were delivered.
2. All containers arrived to their destination no later than their due itme.
3. All drivers returned and reached their depot within their working shift, i.e., they worked at most their maximum number of hours, or didn't work at all.
4. All drivers transported one container at a time (implicit in the algorithm, thus, it is not checked).

Constraints 1 and 2 could be merged into a single one, but it seems more informative to distinguish whether a solution was not feasible because a container wasn't delivered or because a container violated its due time.
The following correctness and consistency checks are made:

1. Drivers correctness.
   i) All drivers returned to their depot.
   ii) All driver routes start and end in their correspondent depot.
2. Containers correctness.
   i) All container routes start from the container's origin endpoint and end in the last hub they were left or its destination endpoint.
3. Routes correctness.
   i) All edges for each route exist in the network, i.e., in the undirected weighted graph.

   ii) All routes are traversable or valid, i.e, consecutive edges have the same node in the end of the first edge and the beginning of the second edge.

4. Transports consistency.

   i) All containers that appear to be transported by a driver in his route are actually being transported by him and not anyone else, appearing as well in all of the transported containers routes. And vice versa, all drivers that appear to be transporting a container in its route are actually transporting it, and not a different one, appearing also in all of the driver routes.

   ii) All transports recorded in a driver route happen in the same exact time in the containers routes and vice versa.

5.4 Implementation

As mentioned earlier, our algorithms have been implemented in Java following standard OOP practices. For that reason in this section we will discuss the main ideas concerning the implementation of the solving approaches.

5.4.1 Project structure

First of all, seems relevant to show the project structure with the classes defined and a short description for each of them. For a detailed description of classes and their methods, constructors, etc. see the documentation provided in javadoc format, delivered alongside this thesis. The following classes have been implemented using this packages' structure:

- Algorithms
  - *Heuristic*: The deterministic greedy heuristic.
  - *BRHeuristic*: The biased-randomized version of the heuristic (BRH), which extends the Heuristic class.
  - *MultiStart*: The biased-randomized multi-start framework (BR-MS).
  - *VNS*: The variable neighborhood search metaheuristic (abstract class).
  - *VNSPI*: The biased-randomized variable neighborhood search for this PI problem (BR-VNS), which extens the VNS class.
- Experiments
  - *Convergence*: Convergence experiments to compare the BR-MS with the BR-VNS approaches.
  - *Experiments*: The results tables are obtained using this class, which executes the three approaches for a given number of seeds for each input instance.
  - *ParameterFineTuning*: Given an instance this class tries to find the most promising interval for the beta parameters with a given size.

  - *Radar*: Obtains the results to create the radar plots.

- IO
  - *Inputs*: Reads instances from files and saves the inputs in private structures that will be used by different algorithms. It is also able to read an outputted solution from a file and rebuild it to be manipulated within the code.
  - *Outputs*: Writes solutions to files and result tables in *csv* format.

- Parameters
  - *Bounds*: Upper and lower bounds for the beta parameter of each geometric distribution.
  - *BRParameters*: Parameters for the biased-randomized heuristic. Only stores the three beta parameters.
  - *MSParameters*: Parameters for the multi-start framework such as the seed, no. of iterations and the bounds for the beta parameters.
  - *VNSParameters*: Parameters for the variable neighborhood search such as the seed, minimum and maximum percentages of destruction, parameter step, no. of iterations and the bounds for the beta parameters needed when resolving a partial solution.

- Problem
  - *Container*: Represents a container to be shipped. Stores information such as id, origin, destination and current nodes, due time or the start time when it was first moved.
  - *ConteinerNode*: Abstract class representing nodes that can store containers.
  - *ContainerRoute*: Represents the route of a container. It extends the class Route.
  - *DepotNode*: Represents a depot from where drivers depart in the beginning of the problem.
  - *Driver*: Represents a driver that transports containers. Stores information such as id, depot and current nodes, start time and max time.
  - *DriverRoute*: Represents the route of a driver. It extends the class Route.
  - *Edge*: Edge between two nodes in the network. It has a weight related to the traveling time, and origin and end nodes. A reference to the reversed edge is stored, so that all edges objects are reused in driver and container routes. The graph is undirected, but the routes have to be stored in such a way that the direction in which the edge was traversed is known.
  - *EndpointNode*: Represents an endpoint. Extends the ContainerNode class.
  - *Event*: Represents an event, which stores the driver that was moved, the container transported (if any), the edge traversed, the start and end times.

  - *HubNode*: Represents a hub. Extends the ContainerNode class.
  - Metrics: Stores metrics of the solution such as drivers elapsed, traveling, stopped and empty times and containers elapsed, traveling and not moving time. It also saves how many containers followed (one of) the shortest path(s).
  - *Network*: Represents the network of an instance. Implements a weighted, undirected graph from the JGraphT library.
  - *Node*: Abstract class that represent a node in the network with id, and x, y coordinates.
  - *PartialSolution*: Represents a partial solution, that is, a solution in a intermediate state, which can be resolved. Extends the class Solution.
  - *Route*: Abstract class that represents a generic route. Stores a list of the edges traversed in the route, its elapsed time, and start time in the solution.
  - *Solution*: Represents a solution in this PI problem. Numerous metrics and structures are stored in this class that will be discussed in Section 5.4.3.

- Useful

  - *Time*: Time-related methods to measure computational times.
  - *Manager*: Useful methods and constants used throughout the project.
  - *Pair*: A generic pair class used in Experiments class.

Figures 5.4 to 5.6 respectively show the inheritance structure of classes Solution-PartialSolution, Heuristic-BRHeuristic, VNS-VNSPI, the ContainerRoute and DriverRoute inheriting from Route and the more complex structure of Node, ContainerNode, DepotNode, EndpointNode and HubNode. Notice that some of their properties are listed below.

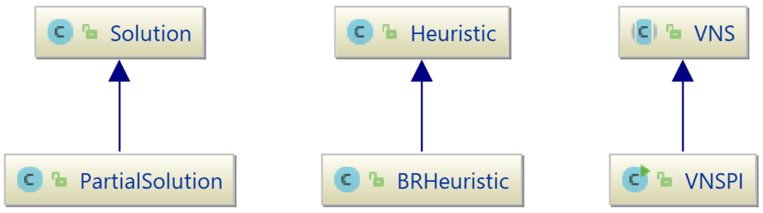


Figure 5.4: PartialSolution, BRHeuristic and VNSPI classes respectively inheriting from Solution, Heuristic and VNS.

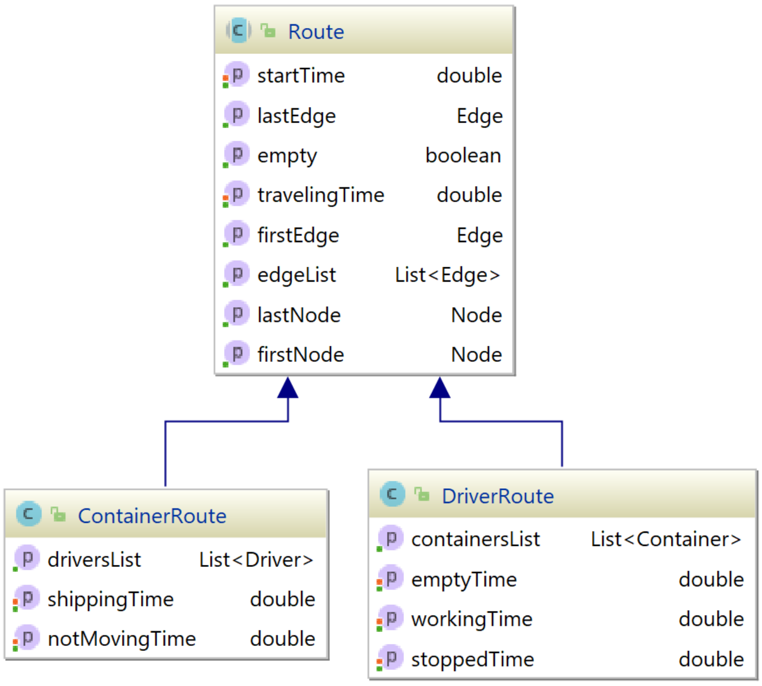


FIGURE 5.5: Route's inheritance structure.

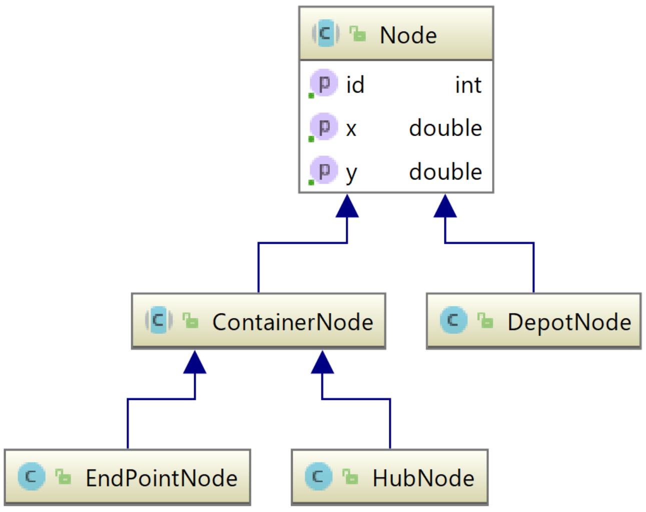


FIGURE 5.6: Node's inheritance structure.

5.4.2 Shortest paths computation

There are two usual choices that we could think of in order to compute the shortest path between any pair of nodes in a sparse undirected weighted graph. The first approach is to run the Johnson's algorithm [67] with worst-case performance $\mathcal{O}(|V|^2 \log |V| + |V||E|)$, which for sparse graphs, $|E| = \mathcal{O}(|V|)$, turns out to be $\mathcal{O}(|V|^2 \log |V| + |V|^2) = \mathcal{O}(|V|^2 \log |V|)$. Because this algorithm is used in directed weighted graphs, a small transformation in an undirected weighted graph would have to be done first, replacing each edge for two directed opposite edges. A second approach would be to run the Dijkstra's algorithm [68] with worst-case performance $\mathcal{O}(|E| + |V| \log |V|)$ which in our case would be $\mathcal{O}(|V| \log |V|)$. Now this algorithm should be executed from each node giving a complexity of $\mathcal{O}(|V|^2 \log |V|)$, the same performance as the previous. Nevertheless, we opted to use the Floyd-Warshall algorithm [60] with worst, best-case and average performance $\mathcal{O}(|V|^3)$. It is slower than the previous two approaches but way easier to be used and manipulated using the JGraphT library, which already has it implemented.

5.4.3 Solution metrics

While we solve an instance using the heuristic, no matter if the biased-randomized version or the deterministic one, the following informational metric fields are stored in the solution:

- *Total time*: the time of the last event triggered.
- *Shipping time*: the elapsed time to deliver all containers, if all were delivered, or the end time of the last transport event, otherwise.
- *Computational time*: computational time to find the solution (in seconds).
- *Elapsed search time*: total elapsed computational time to execute the algorithm (in seconds).
- *No. of delivered containers*: number of containers delivered.
- *No. of containers on time*: number of containers delivered on time.
- *No. of drivers used*: number of drivers mobilized.
- *No. of drivers on time*: number of drivers that returned on time.
- *No. of events*: total amount of events triggered
- *Container routes*: list of container routes by id.
- *Driver routes*: list of driver routes by id.
- *Constraints violated*: list of constraint violated.
- *Event history*: sorted list with all scheduled events in the solution.

Information stored in Metrics class, which is referenced by Solution class, thus, the latter indirectly has stored the former fields, which are:

- *Drivers elapsed time*: total amount of hours worked by all drivers, that is, the sum of all driver shift hours, including stops in their routes.
- *Drivers traveling time*: total amount of hours traveled by all drivers, excluding stops.
- *Drivers stopped time*: total amount of hours spent in stops by all drivers during their routes.
- *Drivers empty time*: total amount of hours in which the trucks were empty during their routes, including stops.
- *Containers elapsed time*: the sum of shipping times of all container routes, including stops.
- *Containers traveling time*: total amount of hours in which the containers where being transported, excluding stops.
- *Containers not moving time*: total amount of hours spent by all containers being stopped, including the elapsed time to be reached by a driver.
- *Use of shortest paths:* number of containers that followed the shortest path.

5.4.4 Candidate's selection in BR

With the formula obtained in Subsection 2.3.6, to select the next candidate out of a sorted list of potential elements, we only need to generate a pseudo-random number in the (0, 1) interval, compute its logarithm, divide it by $\log(1-\beta)$ and perform the modulo with the number of candidates. The *Java* Code 9 shows an easy implementation of this method.

Algorithm 9 Java method to select next candidate in BRH.

```java
private int getNextCandidate(double beta, int size, Random rng) {
        assert beta > 0;
        int index = (int) (Math.log(rng.nextDouble()) / Math.log(1 - beta));
        return Math.floorMod(index, size);
}
```

Notice that is the same, calculating $1 - rng.nextDouble()$ than directly generating a random number between 0 and 1.

Chapter 6

Computational experiments

6.1 Small test instances

In order to test and debug our algorithms we have designed a set of 10 relatively small and simple instances. The number of containers and drivers that comprise these instances is quite low. Some of them have been designed to be unfeasible for the deterministic heuristic on purpose, as our algorithms will give a solution either way. Figure 6.1 shows a clear example of a highly connected hub-and-spoke network with similar size to the ones in the set of instances. Table 6.1 summarizes the main characteristics of these instances giving the size of the network, the no. of depots, hubs and endpoints and no. of containers and drivers.

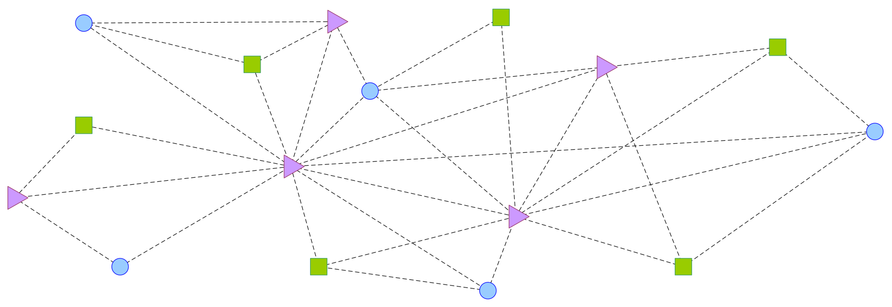


FIGURE 6.1: Example of a small hub-and-spoke network.

Network			Problem	
Instance	Size	D/H/E	#C	#D
instance-01	10	3/3/4	2	3
instance-02	9	3/3/3	3	3
instance-03	12	3/4/5	6	6
instance-04	10	4/3/3	2	5
instance-05	13	4/3/6	9	12
instance-06	11	1/6/4	8	6
instance-07	12	4/4/4	24	40
instance-08	10	3/3/4	2	3
instance-09	13	4/3/6	4	8
instance-10	10	2/3/5	14	16

TABLE 6.1: Small test instances' characteristics.

6.2 Benchmark standardized instances

Table 6.2 presents the characteristics of the 20 benchmark instances that we have created, where we show the size of the network, the no. of depots, hubs and endpoints, the no. of containers and drivers, the parameters used to create each instance, *i.e.*, minimum and maximum no. of drivers per depot, margin for container due times and maximum traveling time of the shortest path between origin and destination, as well as a small summary of the containers shipments including longest, shortest and average path length taking into account only shortest paths.

Network			Problem		Parameters				Shipments		
Instance	Size	D/H/E	#C	#D	D_{min}	D_{max}	M	L	Long	Short	Mean
pi-01	221	82/83/56	25	210	2	5	2	24	19.947	4.131	9.942
pi-02	382	136/167/79	25	605	3	6	2	24	23.241	2.502	12.94
pi-03	508	192/191/125	25	871	3	6	2	24	23.317	3.169	15.858
pi-04	805	281/316/208	25	1159	2	6	2	24	23.006	2.644	16.173
pi-05	599	200/257/142	25	1383	6	8	2.5	24	23.794	7.304	17.013
pi-06	221	82/89/50	50	1228	10	20	2.5	24	21.159	1.435	12.157
pi-07	382	128/168/86	50	1994	10	20	2	24	23.149	1.98	14.452
pi-08	508	182/194/132	50	2795	10	20	2	24	23.757	2.542	14.182
pi-09	805	272/323/210	50	4756	15	20	3	24	23.77	1.835	14.099
pi-10	599	195/267/137	50	3424	15	20	3	24	23.85	2.51	15.731
pi-11	221	85/88/48	100	1258	10	20	2	12	11.948	2.786	8.124
pi-12	382	131/174/77	100	2279	15	20	3	12	11.997	0.593	7.887
pi-13	508	186/198/124	100	3200	15	20	3	12	11.895	0.296	7.618
pi-14	805	252/322/231	100	3818	10	20	2	12	11.974	1.194	8.136
pi-15	599	214/249/136	100	3736	15	20	3	12	11.967	0.782	8.183
pi-16	221	73/86/62	200	1864	20	30	3	12	11.996	0.548	7.928
pi-17	382	133/167/82	200	3322	20	30	3	12	11.993	0.318	8.047
pi-18	508	174/201/133	200	4333	20	30	3	12	11.962	0.58	7.934
pi-19	805	279/327/199	200	7679	25	30	3	12	11.97	0.216	8.206
pi-20	599	190/279/130	200	4830	20	30	3	12	11.963	0.142	7.901

Table 6.2: Benchmark instances' characteristics.

D: depots. H: hubs. E: endpoints. #C: no. containers; #D: no. drivers.
M: margin for container's due time.
L: maximum length of the shortest path between origin and destination nodes for all containers' shipments (in hours).

Figure 6.2 shows the structure of the network from instance pi-07, where circle blue nodes represent depots, triangular purple nodes represent hubs and square green nodes represent endpoints, which is the same convention we used in the example network (Figure 4.1). Representation of the networks from instances 1 to 5 can be found in Appendix A (Figures A.11 to A.15).

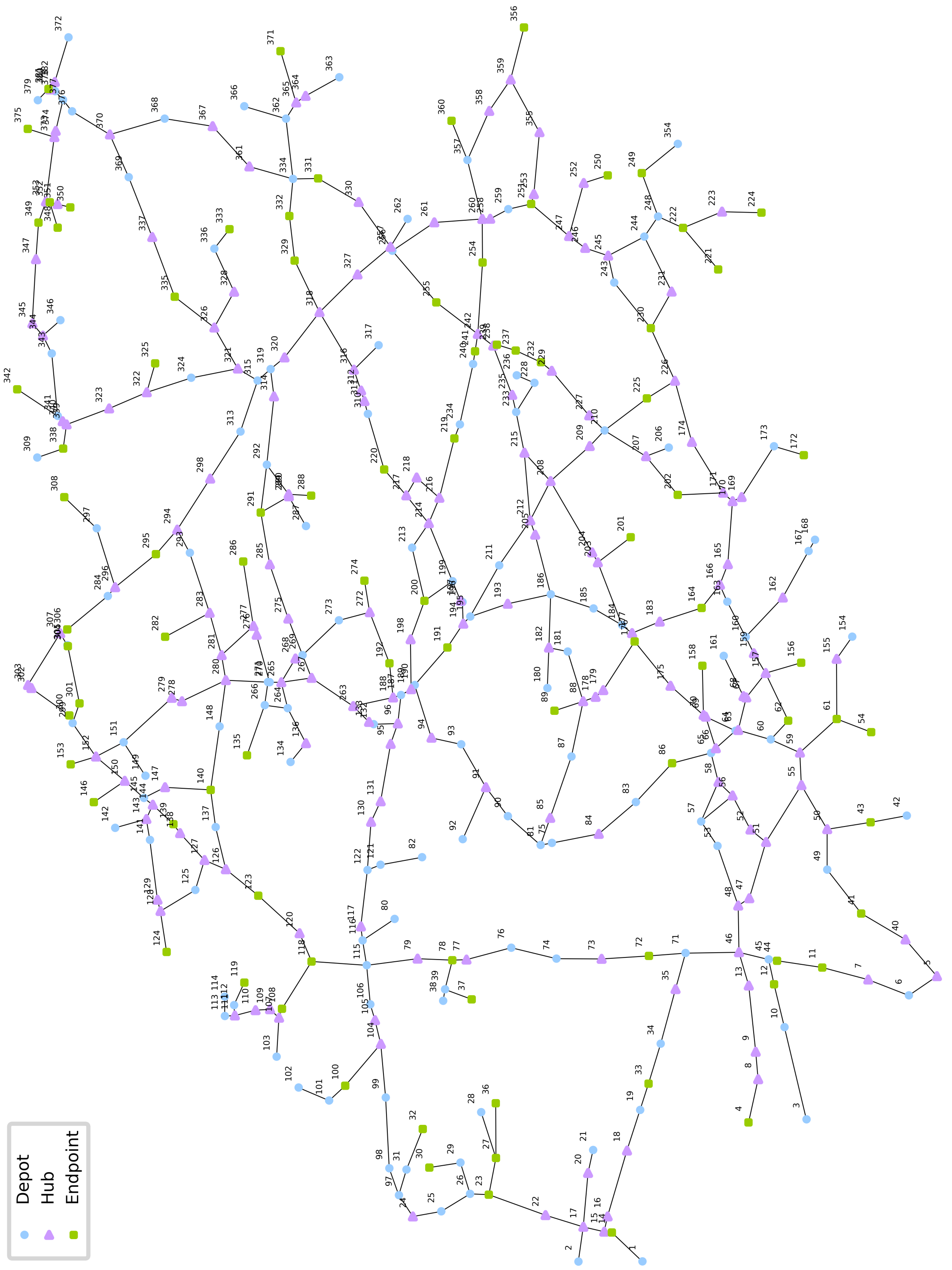


FIGURE 6.2: Network representation of instance pi-07 adapted from TCARP-R2.

With regard to the parameters settings, all beta parameters for the geometric distributions are taking random values in the (0.95, 1.00) interval, as some trial-and-error tests let us determine significant better results for higher beta values, as we would normally expect. Even though they may seem really close to 1.00, thus not diversifying the search as much as possible, the results obtained this way were better in terms of the cost of the solution (total time), but also in terms of the computational time taken. For both solving approaches – BR-MS and BR-VNS – the number of iterations is set to a relatively low value, precisely 100, as we wanted to see the best solutions obtained in a reasonable computational time. In the latter approach, the parameters chosen for the BR-VNS are the following ones: $p_{min} = 20$, $p_{max} = 100$, $step = 1$. These parameters were also set by trial-and-error in *reasonable* ranges until the best results were obtained. Notice that, when the percentage of destruction of the BR-VNS reaches 100, we have the particular case of the BR-MS, *i.e.*, new solutions are built from the beginning. The demon acceptance criterion is used to consider worse solutions and the initial solution is obtained using the deterministic heuristic.

Chapter 7

Results

In order to establish a fair comparison between the proposed algorithms, the deterministic heuristic has to be considered as the solution that a human expert would give taking the best (local) choices in each moment, *i.e.*, it is an upper bound for the other two approaches, which are expected to outperform it in most of the scenarios presented.

Our algorithms have been implemented using Java SE 8.0_151, tested with JUnit 4.12 and all the experiments run in a Dell Workstation Precision Tower Serie 7000, Intel Xeon E5-2650 v4 with 32GB RAM. All instances have been executed 10 times, each with a different seed, and the best results are shown in Table 7.1, where we specify the total time (TT), *i.e.*, the elapsed time to deliver all containers and return all drivers to their depots, the shipping time (ST), which doesn't take into account the time required to make all drivers return, and the computational time (CT) to find the best solution, for each approach and instance. We also show the gaps of the total time between the three approaches.

In addition, we also consider the shipping time as the objective function to be optimized. The results are shown in Table 7.2, but now the gaps correspond to the shipping times of the solutions obtained by each approach. The results for the heuristic are the same though, as it is deterministic. And the comparison is done between the three approaches but also between the results obtained in Table 7.1 to illustrate the differences when optimizing one or the other (total time or shipping time).

More experiments were done with greater number of iterations for both algorithms in order to determine whether this parameter could significantly improve the quality of the solutions obtained. Increasing the number of iterations is the same as giving an extra computational time to the algorithm, so by doing that we should expect slightly better solutions. The results are reported in Table 7.3 with 200 and 300 iterations for the BR-MS and BR-VNS, respectively.

Table 7.1: Performance comparison optimizing the total time.

	Heuristic (1)			**BR-MS (2)**			**BR-VNS (3)**			**Gap in TT (%)**		
Instance	TT (h)	ST (h)	CT (s)	TT (h)	ST (h)	CT (s)	TT (h)	ST (h)	CT (s)	(1)-(2)	(1)-(3)	(2)-(3)
pi-01	33.433	29.167	0.2	30.175	26.418	6.7	30.175	26.418	2.5	9.74	9.74	0.00
pi-02	37.469	35.684	0.3	36.248	34.463	20.6	34.403	31.558	1.6	3.26	8.18	5.09
pi-03	43.310	39.726	0.7	36.602	33.018	13.2	36.602	33.018	48.1	15.49	15.49	0.00
pi-04	40.207	36.329	3.2	36.189	32.311	192.1	39.165	36.658	162.0	9.99	2.59	-8.22
pi-05	52.394	48.199	1.2	42.575	40.331	108.7	43.048	40.804	85.7	18.74	17.84	-1.11
pi-06	37.963	34.191	0.5	32.801	29.899	33.8	33.088	29.316	12.2	13.60	12.84	-0.87
pi-07	40.558	38.195	0.7	39.856	37.485	20.1	39.856	37.485	35.4	1.73	1.73	0.00
pi-08	39.255	36.345	1.2	36.637	33.727	31.5	36.692	33.727	82.1	6.67	6.53	-0.15
pi-09	38.091	35.238	3.3	35.757	32.904	198.2	35.600	33.165	82.6	6.13	6.54	0.44
pi-10	38.362	35.094	2.6	37.648	34.755	217.6	37.530	34.973	75.9	1.86	2.17	0.31
pi-11	24.452	21.296	0.4	21.677	18.602	24.6	21.492	18.439	33.2	11.35	12.11	0.85
pi-12	25.291	22.515	1.0	25.143	22.453	97.9	25.143	22.453	30.0	0.59	0.59	0.00
pi-13	23.382	19.669	1.2	22.452	19.669	110.1	22.925	21.160	122.2	3.98	1.95	-2.11
pi-14	20.704	17.616	5.0	20.545	17.616	263.4	20.215	17.104	126.5	0.77	2.36	1.61
pi-15	25.792	23.146	1.7	24.099	21.511	241.2	25.123	23.146	74.3	6.56	2.59	-4.25
pi-16	29.708	25.208	0.8	25.231	20.474	6.8	24.857	21.448	37.9	15.07	16.33	1.48
pi-17	22.976	19.884	1.6	22.529	19.622	186.8	22.573	19.587	49.0	1.95	1.75	-0.20
pi-18	27.422	23.020	2.2	25.385	22.038	260.9	25.562	22.177	236.6	7.43	6.78	-0.70
pi-19	23.134	18.642	5.4	22.340	19.091	577.0	22.741	18.642	113.6	3.43	1.70	-1.79
pi-20	27.332	24.750	2.6	25.026	21.706	107.9	24.884	21.886	44.7	8.44	8.96	0.57
Avg.			**1.8**			**136.0**			**72.8**	**7.34**	**6.94**	**-0.45**

TT: total time; ST: shipping time; CT: computational time.

TABLE 7.2: Performance comparison optimizing the shipping time.

	Heuristic (1)			**BR-MS (4)**			**BR-VNS (5)**			**Gap in ST (%)**				
Instance	ST (h)	TT (h)	CT (s)	ST (h)	TT (h)	CT (s)	ST (h)	TT (h)	CT(s)	(1)-(4)	(1)-(5)	(4)-(5)	(2)-(4)	(3)-(5)
pi-01	29.167	33.433	0.2	26.927	31.193	1.6	26.418	30.175	2.2	7.68	9.43	1.89	-1.93	0.00
pi-02	35.684	37.469	0.3	30.753(*)	34.425	36.4	31.558	34.486	2.8	13.82	11.56	-2.62	10.77	0.00
pi-03	39.726	43.31	0.7	33.018	36.602	26.4	33.018	36.602	27.9	16.89	16.89	0.00	0.00	0.00
pi-04	36.329	40.207	2.7	33.684	37.562	31.5	35.305(†)	39.183	115.4	7.28	2.82	-4.81	-4.25	3.69
pi-05	48.199	52.394	1.1	39.086(*)	41.395	20.6	40.832	43.076	87.7	18.91	15.28	-4.47	3.09	-0.07
pi-06	34.191	37.963	0.3	28.646(*)	32.418	31.8	29.342	33.114	4.4	16.22	14.18	-2.43	4.19	-0.09
pi-07	38.195	40.558	0.7	37.485	39.856	44.6	37.485	39.856	35.6	1.86	1.86	0.00	0.00	0.00
pi-08	36.345	39.255	1.1	33.727	36.637	15	33.727	36.637	63.2	7.20	7.20	0.00	0.00	0.00
pi-09	35.238	38.091	2.9	32.904	35.757	183.3	33.209	36.097	161.8	6.62	5.76	-0.93	0.00	-0.13
pi-10	35.094	38.362	1.7	34.713(†)	38.358	29.2	33.237(*)	37.354	134.5	1.09	5.29	4.25	0.12	4.96
pi-11	21.296	24.452	0.3	18.694	21.866	19.3	18.975	21.968	11.7	12.22	10.90	-1.50	-0.49	-2.91
pi-12	22.515	25.291	0.7	21.107(*)	24.431	10.1	22.453	25.143	28.8	6.25	0.28	-6.38	5.99	0.00
pi-13	19.669	23.382	1.2	19.372(†)	23.508	181.4	19.386(†)	23.099	32.5	1.51	1.44	-0.07	1.51	8.38
pi-14	17.616	20.704	3.2	17.454(†)	21.02	165.4	17.509	21.424	128.8	0.92	0.61	-0.32	0.92	-2.37
pi-15	23.146	25.792	1.7	21.601	25.625	221.6	22.373(†)	26.603	6.8	6.68	3.34	-3.57	-0.42	3.34
pi-16	25.208	29.708	0.8	20.306(*)	24.806	68.7	21.813	26.756	94.9	19.45	13.47	-7.42	0.82	-1.70
pi-17	19.884	22.976	2.8	19.511(†)	23.163	248.3	19.51(†)	22.823	63.5	1.88	1.88	0.01	0.57	0.39
pi-18	23.02	27.422	2.4	22.038	25.422	193.1	22.138(†)	26.834	284.3	4.27	3.83	-0.45	0.00	0.18
pi-19	18.642	23.134	5	18.564(†)	23.072	574.3	18.564	22.741	281.4	0.42	0.42	0.00	2.76	0.42
pi-20	24.75	27.332	2.7	21.706	25.026	146.3	21.912	24.988	97.2	12.30	11.47	-0.95	0.00	-0.12
Avg.			**1.6**			**112.4**			**83.3**	**8.2**	**6.9**	**-1.5**	**1.2**	**0.7**

Marked with (†) are the shipping times (ST) that are lower than the ones obtained in Table 7.1 but with greater total times (TT).
Results marked with (*) indicate that both total and shipping times are by chance lower than the ones in the previous table.
Even though the methods (2), (4) and (3), (5) are respectively the same, we have differentiated them because the former (2) and (3) were optimizing the total time, whereas the others were optimizing the shipping time.

Table 7.3: Performance comparison of the search with a greater number of iterations.

	Heuristic (1)			**BR-MS (2)**			**BR-VNS (3)**			**Gap (%)**		
Instance	TT (h)	ST (h)	CT (s)	TT (h)	ST (h)	CT (s)	TT (h)	ST (h)	CT (s)	(1)-(2)	(1)-(3)	(2)-(3)
pi-01	33.433	29.167	0.2	30.175	26.418	8	30.175	26.418	15.8	9.74	9.74	0.00
pi-02	37.469	35.684	0.3	34.815	33.03	38.1	34.403	31.558	7.1	7.08	8.18	1.18
pi-03	43.31	39.726	0.7	36.602	33.018	111.1	36.602	33.018	91.6	15.49	15.49	0.00
pi-04	40.207	36.329	2.1	35.714	32.86	138.1	38.426	34.548	560.1	11.17	4.43	-7.59
pi-05	52.394	48.199	1.1	42.201	39.957	30.2	42.575	40.331	226.7	19.45	18.74	-0.89
pi-06	37.963	34.191	0.3	31.629	28.727	53	31.451	28.479	8.4	16.68	17.15	0.56
pi-07	40.558	38.195	0.7	38.064	35.665	66.1	39.856	37.485	36.2	6.15	1.73	-4.71
pi-08	39.255	36.345	1.1	36.692	33.727	160	36.637	33.727	62	6.53	6.67	0.15
pi-09	38.091	35.238	2.8	35.302	32.349	274.2	35.082	32.651	776.5	7.32	7.90	0.62
pi-10	38.362	35.094	1.6	37.612	34.755	93.8	36.368	33.55	322.8	1.96	5.20	3.31
pi-11	24.452	21.296	0.3	21.93	18.759	61.1	21.257	18.23	36.4	10.31	13.07	3.07
pi-12	25.291	22.515	0.7	24.397	21.03	122.5	24.917	21.55	127.6	3.53	1.48	-2.13
pi-13	23.382	19.669	1.2	22.367	19.825	153.5	21.768	19.386	281.8	4.34	6.90	2.68
pi-14	20.704	17.616	2.9	20.336	17.225	609.8	20.501	17.616	562.1	1.78	0.98	-0.81
pi-15	25.792	23.146	1.6	24.12	21.93	294.4	24.638	23.146	66.7	6.48	4.47	-2.15
pi-16	29.708	25.208	0.7	25.437	21.123	49	24.006	20.938	255.2	14.38	19.19	5.63
pi-17	22.976	19.884	1.5	22.458	19.622	534.8	22.345	19.51	40.6	2.25	2.75	0.50
pi-18	27.422	23.02	2.1	25.385	22.038	373.9	25.472	22.038	303.2	7.43	7.11	-0.34
pi-19	23.134	18.642	4.7	22.309	18.564	1025.2	21.941	19.204	726.6	3.57	5.16	1.65
pi-20	27.332	24.75	2.5	24.62	21.544	424	24.404	22.121	1122.9	9.92	10.71	0.88
Avg.			**1.5**			**231.0**			**281.5**	**8.28**	**8.35**	**0.08**
1-10			1.09			97.3			210.7	10.16	9.52	**-0.74**
11-20			1.8			364.8			352.3	6.40	7.18	**0.90**

Chapter 8

Analysis and discussion

Figure 8.1 shows the results presented in Table 7.1, where we represent the gaps in terms of the total time between the three different approaches for the 20 instances created.

It is easy to see that both algorithms, BR-MS and BR-VNS, have a similar performance, with 50% of the gaps between 2-11% (1st and 3rd quartile), but slightly better in average for the former. When comparing the performance of these two approaches, it really depends on the instance whether one approach or the other will obtain better results.

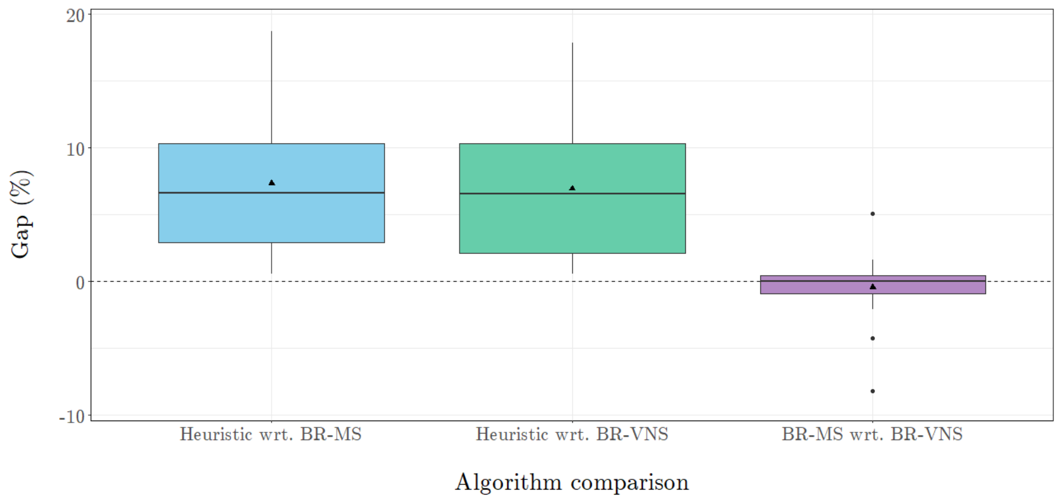


FIGURE 8.1: Performance comparison showing the gaps of TT.

However, if we take a look at Table 8.2, where the computational times are represented for each solving approach, we can grasp that the metaheuristic is substantially faster than the BR-MS (46.47%), and that the deterministic heuristic is way faster than both other approaches, having computational times of less than 6 seconds in the worst case. The difference resides in the fact that, at each iteration, the BR-VNS takes advantage of the previous one reusing its partial solution, so that only a small percentage of decisions has to be rolled back, and resolving it from an advanced system state is faster than solving it from scratch, as the BR-MS does.

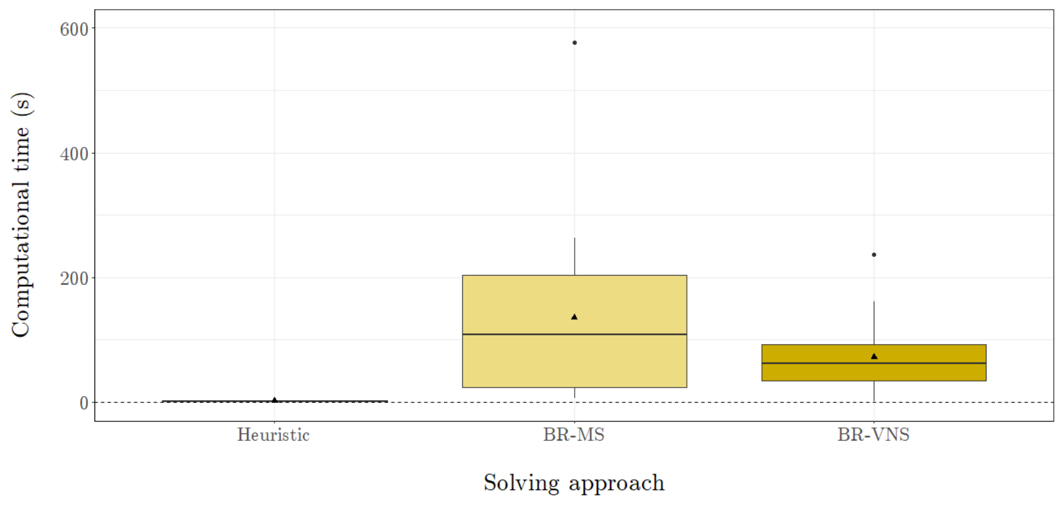


FIGURE 8.2: Performance comparison in terms of the computational time.

Figure 8.3 represents the gaps in terms of the shipping time, when the objective function wasn't the total time (Table 7.2). In this case, better results were reported by the BR-MS, even though the computational times, which are not represented, were still greater for this approach.

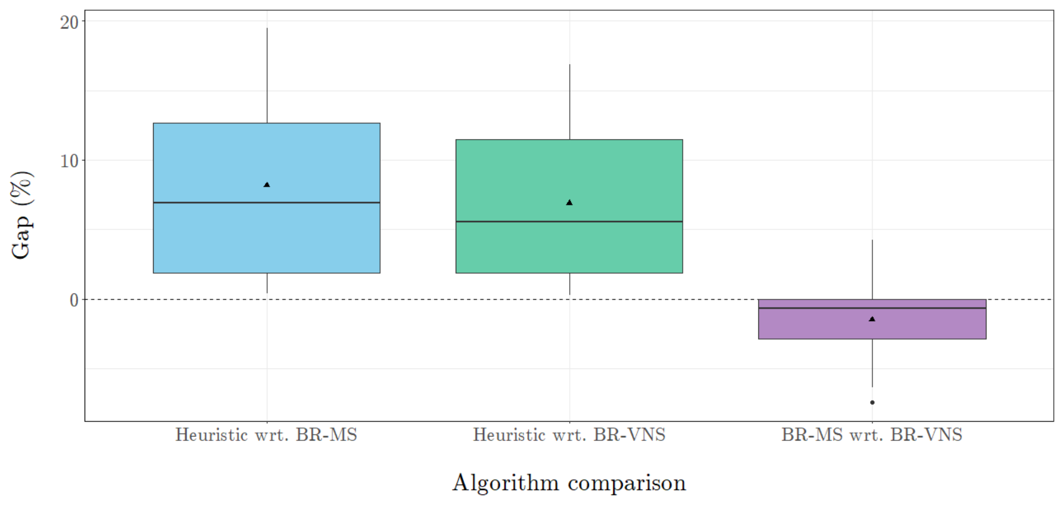


FIGURE 8.3: Performance comparison showing the gaps of ST.

Figures 8.4 and 8.5 represent the results of Table 7.3 in which the number of iterations was increased. However, in the former figure we have distinguished the small instances (25 and 50 containers), from the larger ones (100 and 200 containers). Two differences arise from this figure: *i*) greater gaps have been obtained for small instances; and *ii*) the BR-MS approach tends to outperform the BR-VNS in small instances, whereas the opposite is also true, *i.e.*, the BR-VNS obtains better results than the BR-MS in larger instances.

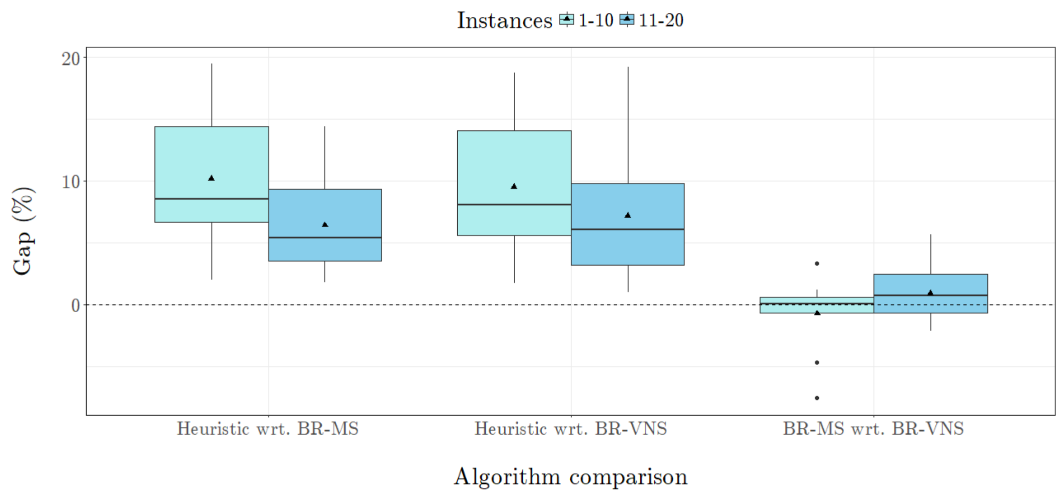


FIGURE 8.4: Performance comparison with a greater number of iterations between small and large instances.

The difference on the results when both approaches were boosted with a greater number of iterations is shown in Figure 8.5. In this case, the BR-VNS obtains lower or equal total times than in the initial results of all instances except for the *pi-14*.

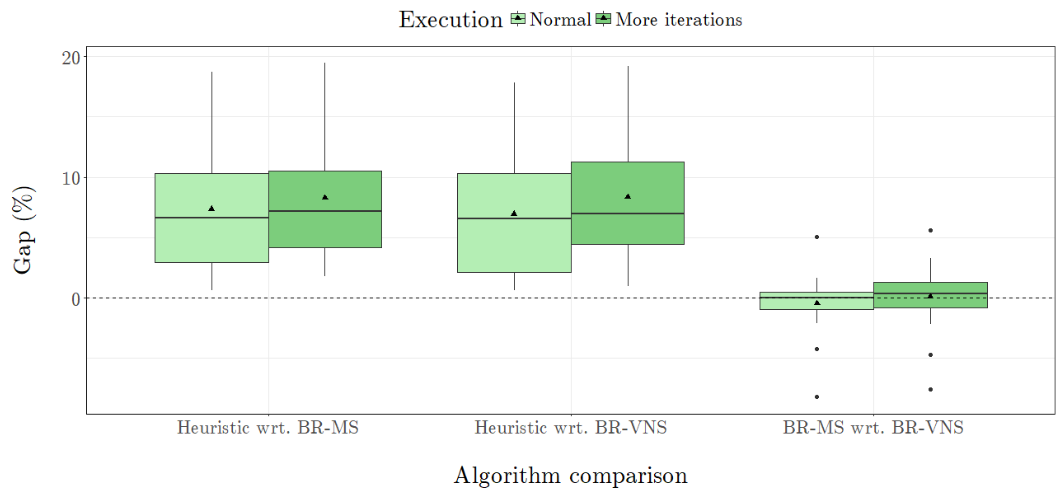


FIGURE 8.5: Performance comparison between initial results and boosted ones.

As mentioned earlier, the BR-VNS approach tends to find high-quality solutions in less computational time than BR-MS. Figure 8.6 shows the execution of both algorithms for instance *pi-02* in a 7.5 seconds time frame. The horizontal purple dashed line represents the cost (upper bound) of the solution obtained by the deterministic heuristic. That's why the first solution obtained by the BR-VNS (in blue) starts on a solution with that cost, whereas the BR-MS (in green) does not in this case. The feasible solutions obtained by the algorithms are marked with dots, while unfeasible ones are marked with crosses. The best solution of the BR-VNS is obtained at 0.9 seconds, quite earlier than the BR-MS, which is obtained around 4.8. The BR-VNS approach speeds up the convergence to a high-quality solution in this case.

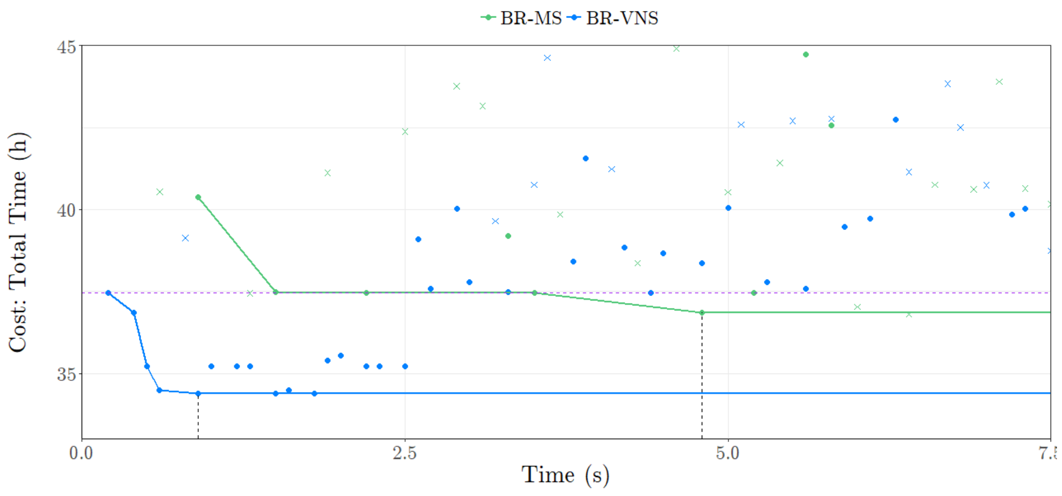


FIGURE 8.6: Convergence of BR-MS and BR-VNS approaches for instance *pi-02*.

There are many other characteristics that define a solution beyond the total time, the shipping time and the computational time. For instance, the no. of drivers deployed, the amount of hours spent by the drivers, the total time that the drivers were actually driving, among others. Seems interesting to represent solutions in a way that all these features can be represented and taken into account. Figure 8.7 represents 6 characteristics of three different solutions of instance *pi-06* obtained by a different solving approach. We have represented the total, shipping and computational time, and as mentioned before, the drivers' traveling time, *i.e.*, total amount of hours driven by drivers (without taking into account stops), drivers total/elapsed time, *i.e.*, total amount of shift working hours, and the no. of drivers moved.

Notice that the data of these radar plots has been reversed, that is, we would normally want to minimize all six features, which is the same as having a solution with all features at 0%. However, in order to ease the visualization, we've done the opposite, optimizing a feature means having the greatest value, 100%, *e.g.*, the solution obtained by the deterministic heuristic (in purple) is the fastest with the minimum computational time, thus, we can see in Figure 8.8a that this feature is maximized.

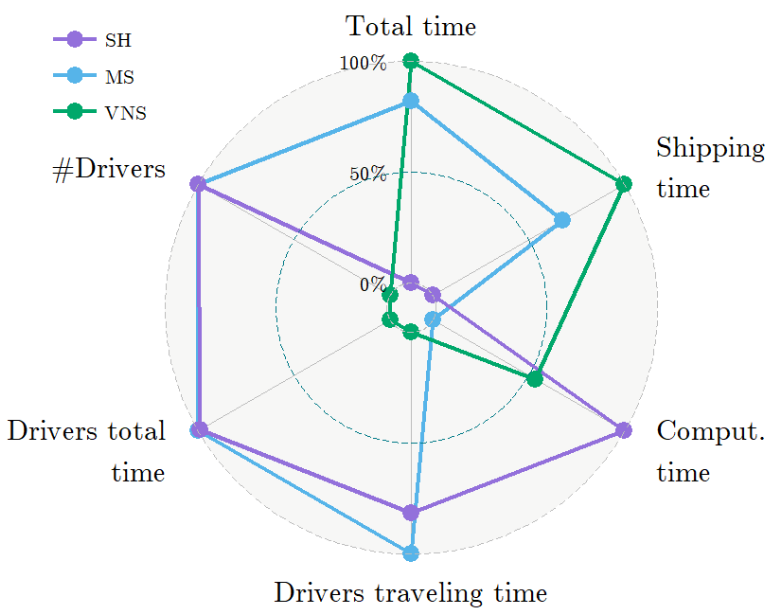


FIGURE 8.7: Comparison of six main characteristics between three solutions of instance *pi-06*.

The main features extracted from the three solutions chosen and represented in Figure 8.7 can be seen in Table 8.1, where *#D* stands for the no. of drivers moved, *DET* is the drivers elapsed time (or drivers total time) and *DTT*, the drivers traveling time. All units are in hours, except for the number of drivers.

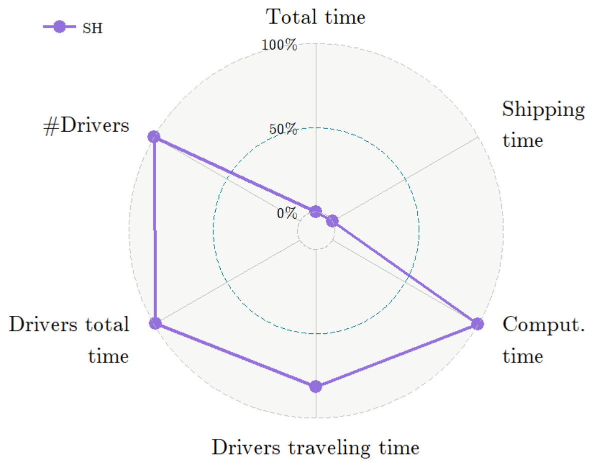
Approach	TT	ST	CT	#D	DET	DTT
Heuristic	37.963	34.191	0.4	186	1648.2	1305.3
BR-MS	32.479	29.378	10.3	186	1646	1279.2
BR-VNS	31.281	27.078	5	211	1863.9	1421.5

TABLE 8.1: Three solutions and their characteristics for instance *pi-06*.

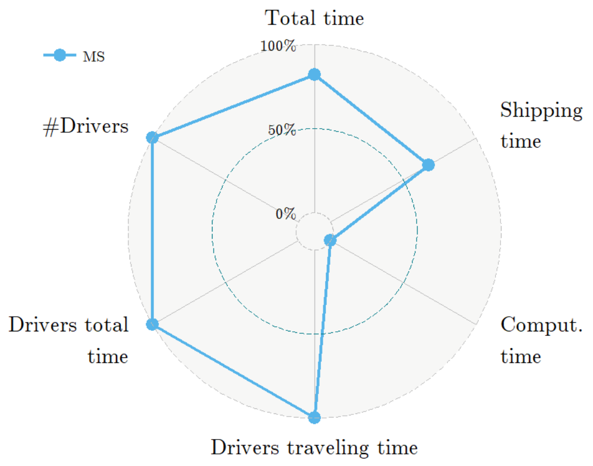
Figures 8.8a to 8.8c show each solution separately. In this case, the solution obtained by the BR-VNS approach is the one with minimum total and shipping times, but it also uses more resources than the others, that is, 25 more drivers and more traveling hours spent.

With regard to the physical internet paradigm addressed in this thesis, their advantages seem quite relevant when routing a fleet of trucks. For instance, we can do a simple comparison between the conventional transportation system, that is, pairing a driver with the whole route of a container, and the PI approach. Taking as an example the *pi-02* instance we can easily state the differences on the solutions obtained by both models just focusing on the longest shipment: container no. 3 has a traveling time of 23.241h from origin to destination using the shortest path in the network, a 46h due time and the closest driver is at 0.425h from it.

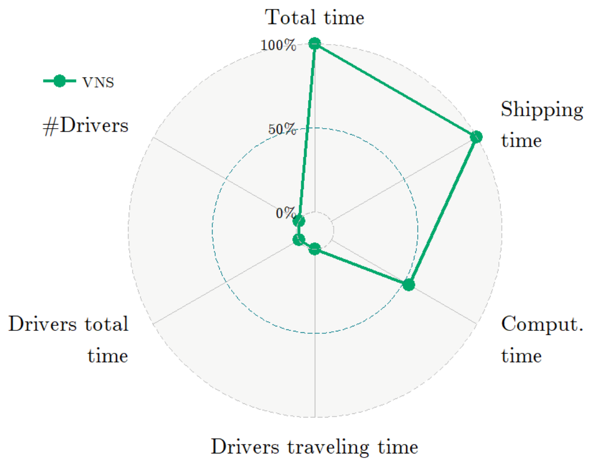
The traditional paradigm would activate a truck driver to perform the whole service, which means that the hours of service regulation (HOS) for property-carrying drivers established in the U.S. [69], or similar regulations would apply. This regulation can be summarized in 2 main rules: *i*) the 11/14h driving limit rule, *i.e.*, a worker "*may drive a maximum of 11 hours*



(A) Heuristic solution characteristics.



(B) BR-MS solution characteristics.



(C) BR-VNS solution characteristics.

FIGURE 8.8: Visualization of six characteristic from three solutions side by side.

after 10 consecutive hours off duty" and also "*may not drive beyond the 14th consecutive hour after coming on duty, following 10 consecutive hours off duty*"; and *ii*) a 30-minute rest break (at least) is required before the 8th consecutive driving hour by law. The 60/70h limit rule doesn't apply in this example, as the shipment doesn't exceed a 4-day trip and this rule tries to limit the 14-10h on and off duty cycles so that can't be abused. So if we don't consider the service times – loading, unloading, fueling, etc. – nor the eating breaks as we will already account the rest breaks, the minimum shipping time estimated for the container no. 3 would be of 44.666h, and the total time, 88.482h at best. Notice that we are not considering the 0.425h traveling time from the depot to the endpoint in the return. Table 8.2 summarizes the estimated route that the driver would do following these regulations.

Start	End	Duration	Activity	Traveling left
0.000	0.425	0.425	driving	23.241
0.425	8.425	8.000	driving	15.241
8.425	8.925	0.500	break	15.241
8.925	11.50	2.575	driving	12.666
11.50	21.50	10.00	resting	12.666
21.50	29.50	8.000	driving	4.666
29.50	30.00	0.500	break	4.666
30.00	33.00	3.000	driving	1.666
33.00	43.00	10.00	resting	1.666
43.00	44.666	1.666	driving	0.000
44.666	51.00	6.334	driving	-6.334
51.00	51.50	0.500	break	-6.334
51.50	54.50	3.000	driving	-9.334
54.50	64.50	10.00	resting	-9.334
64.50	72.50	8.000	driving	-17.334
72.50	73.00	0.500	break	-17.334
73.00	76.00	3.000	driving	-20.334
76.00	86.00	10.00	resting	-20.334
86.00	88.482	2.482	driving	-22.916

Table 8.2: Traditional route of a driver.

On the other hand, the best results obtained by our algorithms with driver's working shifts between 8 and 10 hours report a shipping time of 31.558h for the container no. 3, and a total time of 34.403h for the 25 containers. Seems clear the benefits that this model would report, although a deeper comparison on the resources used by one and the other should also be further addressed, as 8 drivers were needed in the PI model.

Chapter 9

Management

9.1 Planning

The length of this project is estimated in 15 weeks, starting from October the 2nd, 2017 until January the 14th, 2018. The oral defense is scheduled on the week of 22nd - 27th of January, which means that we aim to have our thesis dissertation written, finished and revised on the 15th of January, having a whole week to prepare the oral defense.

These deadlines as well as the elapsed time to carry on the project cannot be changed during its development. However, it is important to remark that, as for any project based on agile methodologies, new tasks might come in with new requirements as well as others might go away. So we will need to adapt and be flexible enough so that the project can be finished on the due time mentioned.

Figures 9.1, 9.2 respectively show a final and initial time planning with the use of a Gantt chart generated on an online platform (`teamgantt.com`). However, *MS Team Foundation Server* will be used to keep track of smaller tasks in 2-week duration sprints.

Tasks description

In this section we will describe the tasks involved in the development of the project. Tasks significantly related to each other have been aggregated into small groups. These groups have a workload that ranges from 1 to 4 weeks to be properly finished, but no more than that. Otherwise seems that they are too big for the appropriate control and monitoring.

First of all, it is necessary to point out that we are taking into account the time required for documentation in those tasks that are specifically related to content that will be in the thesis, *e.g.*, literature review, discussion and conclusions. This way we won't need to write the whole thesis all at once two weeks before the deadline. However, for technical tasks such as the development of an algorithm, specific documentation tasks have been defined.

For all tasks, a computer and access to Internet are needed. So from now on we won't be mentioning them as material resources. On the other hand, the only human resources needed will be our dedication as well as the one from our supervisors. We won't be mentioning them either.

In table 9.1 the groups of tasks that we aim to finish are listed with the estimated elapsed time (in weeks and hours). We have sorted them in a logical order of precedence but an explanation on their dependencies can be found later on. Each week is approximately equal to 40 hours of work at least, that is, 8 hours a day, 5 days a week. So the expected overall elapsed time of the project will be of 600 hours.

Task	*Elapsed time*
Research extension	*2 weeks ≈ 80 hours*
Basic approaches validation and testing	*3 weeks ≈ 120 hours*
Metaheuristics framework	*4 weeks ≈ 160 hours*
Create benchmark instances	*2 weeks ≈ 80 hours*
Experiments, analysis and conclusions	*3 weeks ≈ 120 hours*
Oral defense	*1 week ≈ 40 hours*
	15 weeks ≈ 600 hours

Table 9.1: Groups of tasks with their duration.

Research extension

A wide context and theoretical background for this project was provided in the first GEP deliverable, however, we consider important to extend it so that the reader can achieve a deep understanding on logistics topics and the physical internet initiative, but more importantly, on combinatorial optimization problems, and optimization methods such as heuristics and metaheuristics.

We also showed the state-of-the-art problems tackled by other researchers through a literature review on similar problems they are dealing with. But we will try to consider more authors, and try to find more scientific papers that might help us with our specific problem.

In order to do so, we have defined two tasks: *background and context extension*, *literature review extension*. We will be working on them for two weeks at the same time, because one can help the other. A printer might be needed as a material resource to ease the process of reading and summarizing dozens of papers.

Basic approaches validation and optimization

Because this project was started a bit earlier we already have started to implement two solving approaches: a simple heuristic and a biased randomized heuristic algorithms. We need to finish their development, verify and validate that they are working fine. In order to do so we will test some of our code using *JUnit* and implement another algorithm that, given a solution for the problem, validates whether is correct and consistent. We will also optimize the parameters of the code that are significantly big enough to be defined as a task: *parameter fine-tuning.* To perform these tasks we will need specific software: *Java* programming language, *IntelliJ Idea* IDE and *JUnit* testing software.

Metaheuristics framework

Another solving approach for our problem will be based on the use of a specific metaheuristic. During this part of the project we will need to decide which metaheuristic use, implement it in *Java* and document it. For this reason we have defined two tasks: *algorithm development* and *documentation.* Within the development of an algorithm we will always have two subtasks: *implementation* and *testing.* This bunch of tasks has a 4-week expected time of completion.

Create benchmark instances

A set of benchmark instances will be adapted from a dataset of 5 networks. Different scenarios will be generated with different fleet sizes and number of containers to deliver. The assignments will be randomized so that we can generate hundreds of tryouts and choose those in which the basic solving approaches are able to obtain feasible solutions. We have estimated that we will be working on this for 2 weeks.

Experiments, analysis and conclusions

Computational experiments from the generated instances will be done. These experiments will be executed using all the listed solving approaches. The analysis will be performed according to different comparison methods between solutions. Three tasks have been defined for this group. The expected elapsed time of this group of tasks is 3 weeks as it might take quite long to gather all the results of different executions when the size of the instances is big enough.

Oral defense

The preparation of our oral defense will be done during a week. We will need to decide and prepare the resources to do it, such as the type of presentation slides we will use (*MS Powerpoint* or *Beamer*). It will be really helpful to practice with our supervisors, family and friends (human resources), so we will ask for their time.

Rescue week

In addition, we have defined a whole spare week that we can use in case any task takes longer than estimated. This way the time planning isn't as tight as it would've been if we hadn't defined this week as free. With this strategy we gain flexibility on our plan.

Alternatives and action plan

One of the most common obstacles or risks is to underestimate the workload of a task, *e.g.*, some programming tasks might have to face bugs or strange behaviors that are really time consuming. If that happens we have scheduled a free week that can be used anytime for those important tasks that are needed to keep going forward on the project. The way to go will be moving the time planning a day or as many as needed. But if we really have to face it more than once we will need to shorten the scheduled time for one of the frameworks we want develop or we will even have to omit one of them.

This project is really aimed to be able to develop the metaheuristics framework, that's why we have decided to give more time to this group of tasks, as it is also more complex.

We also dispose of the help of two supervisors that will monitor the tasks as well. So it won't only be one person paying attention to possible deviations to the time planning. Some other possible causes of modifications on the plan and delays are listed below.

Access to scientific journals

As a student our access to some scientific papers is restricted to the access granted to the UPC, or to the open public. In this case if some fundamental papers were needed and we don't have access to them the only thing we can do is to ask our supervisors or to ask directly to the authors, as it is done sometimes in the *Research Gate* platform.

Dependencies between tasks

We've tried to minimize the dependencies between tasks so that we don't end up stuck in a bottle neck working on a single big task that enables the rest. However, in the group of tasks *Basic solving approaches validation* the *validation and verification* task is required for the metaheuristics framework, as it will be implemented on top of the biased-randomized heuristic. It is obvious though, that the *experiments, analysis and conclusions* can only be done once all the algorithms have been developed. Also notice that the experiments can be started only after adapting and creating our instances. The rest of tasks are independent and they wouldn't need to be ordered this way, but we've decided to sort them by increasing potential performance gain.

Resources

This project doesn't require many material resources apart from a computer and an Internet connection, but in this subsection the resources needed for the development of the project are listed distinguishing hardware from software resources. We have also identified four different roles needed as human resources.

Hardware

- Acer Aspire E 15
- HP Elite Desktop

Human

- Project manager
- Software developer
- Data scientist
- Researcher

Software

- Microsoft Windows 10/Linux Ubuntu 16.04
- Microsoft Office 365
- Microsoft Visio Pro 2016
- Microsoft Team Foundation Server
- IntelliJ IDEA
- Oracle JDK 8
- Sublime Text 3
- Git/GitHub
- Overleaf

Final planning

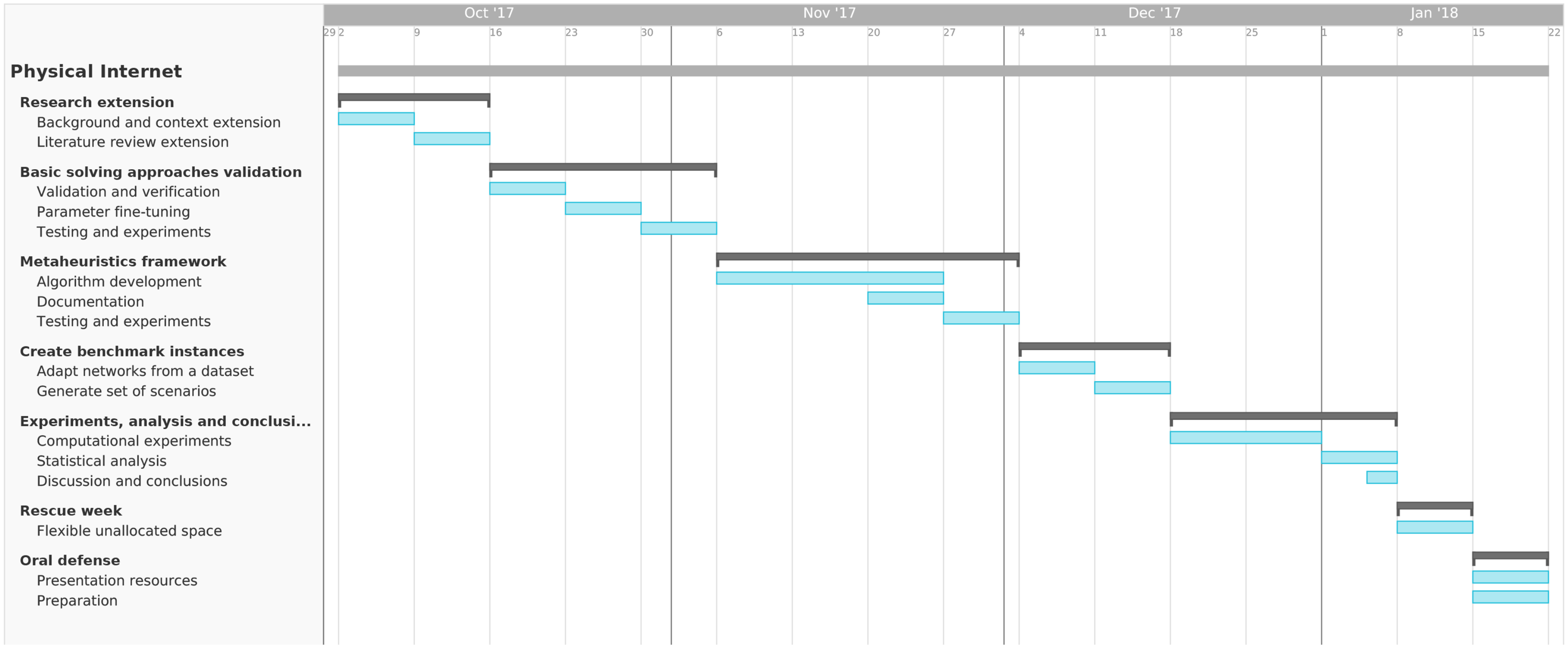


FIGURE 9.1: Gantt chart of the final planning.

Initial planning

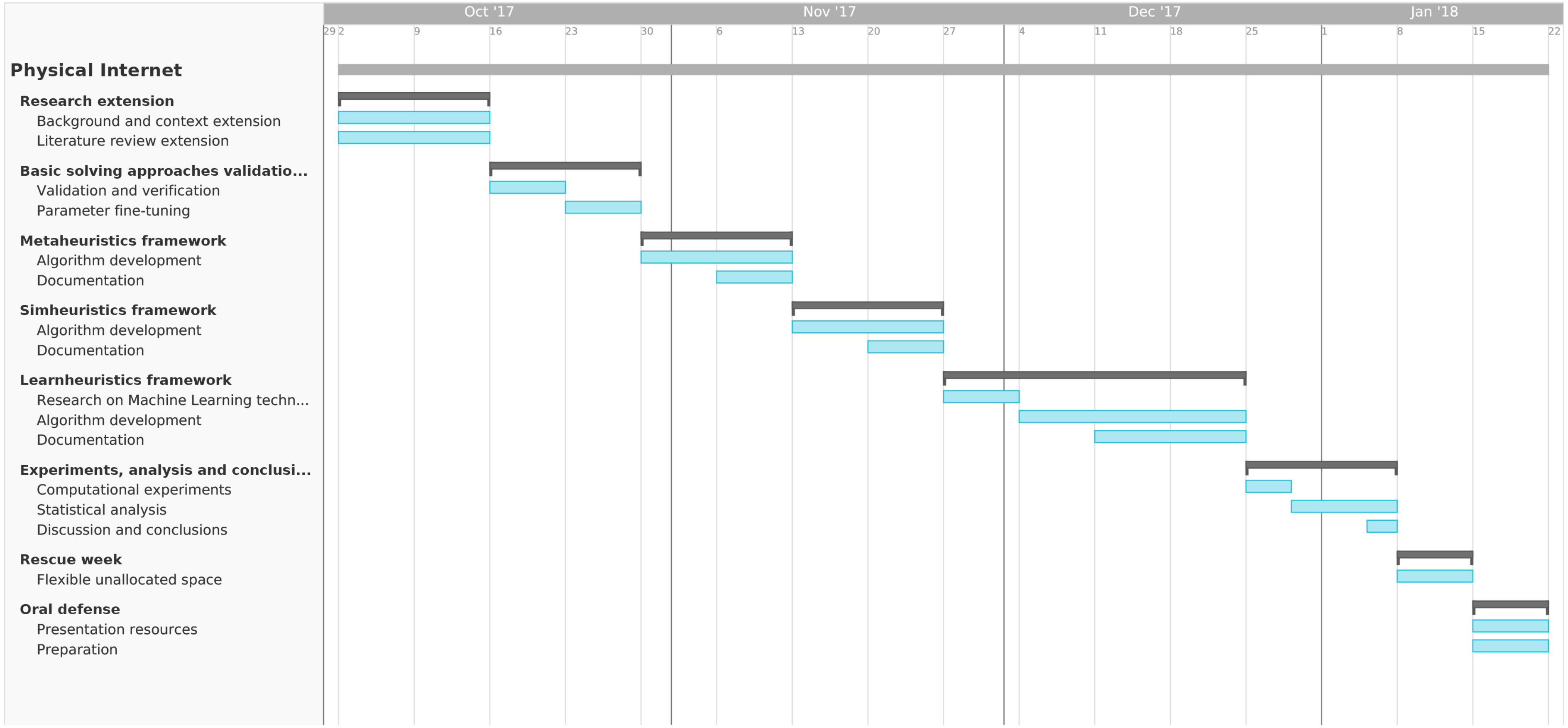


FIGURE 9.2: Gantt chart of the initial planning.

9.2 Budget estimation

In this section a detailed estimation of the budget for this project is done taking into account direct, indirect and incidental costs. Within the direct costs, we emphasize the differences between the costs associated to hardware, software and human resources.

Direct costs

Direct costs are strictly related to the costs of the execution of the tasks and activities explained in our planning, that is, the ones arranged in the Gantt chart. For that reason we need to list the needed resources, material and human, mentioned in the previous section to consider their costs or prices and their amortizations.

The amortization A_p for a given product p has been calculated as follows:

$$A_p = PP \cdot \frac{TU}{50\,UL} \qquad \text{where,}$$

PP is the Product Price (in euros)
TU stands for the Time in Use of the product (in weeks)
UL represents its Useful Life (in years)

We are assuming that a year has an average of 52 weeks, 50 of them are workable, the rest are accounted as holidays. We also assume that the product isn't used off work and that generally it could be used 8h a day, as in a full time job, also the premise for the planning of this project. This way seems easier to do the math in weeks, instead of hours, because we account the same number of hours per week in TU as in UL.

We'd like to mention that useful life periods for the computers have been extracted from the current tax depreciation in the U.S., that is, *MACRS GDS*. Other useful life periods such as the license of different software are just the development elapsed time of the the project, that is, 4 months.

Finally, it is also necessary to mention that we only need one unit of each product and for this reason the number of units isn't placed in every table.

Hardware resources

In this subsection we have estimated a budget for the hardware resources needed. In our case we only need the use of two computers: a laptop and a desktop computer. The useful life of a computer is estimated in 5 years and even though its use is one of the factors of its depreciation other factors such as the efflux of time, obsolescence or accidents are also considered in this estimation. table 9.2 shows the hardware resources needed in detail.

Product	Price (€)	Useful life (years)	Amortization (€)
Acer Aspire E 15	390	5	23.4
HP Elite Desktop	400	5	24.0
Total	**790**		**47.4**

Table 9.2: Hardware resources budget.

Software resources

In this subsection we have estimated a budget for the software resources needed. We've tried to choose open source software as well as student free packs for specific programs such as GitHub or IntelliJ IDEA. However, Office pack and Visio Pro seem really adequate for this project. Table 9.3 shows a detailed list of the software resources needed.

Product	Price	Useful life	Amortization
	€/month	months	€
Microsoft Windows 10	Included	-	0
Microsoft Office 365	7	4	28
Microsoft Visio Pro 2016	11	4	44
IntelliJ IDEA	0	-	0
Oracle JDK 8	0	-	0
Team Foundation Server	0	-	0
Sublime Text 3	0	-	0
GitHub	0	-	0
Overleaf	0	-	0
Total	**18**		**72**

Table 9.3: Software resources budget.

Human resources

We have identified four essential roles that will take part in the project:

- **Project manager**: responsible for planning, executing, monitoring and closing the life cycles within the project, that is, all its tasks and subtasks. The average gross salary of an IT Project manager is about €42,000[1].
- **Software engineer**: responsible for the design, development, testing and evaluation of a software system, as well of its documentation. The average salary of a Software engineer is about €31,000.

[1]All wages mentioned are gross and averaged in Spain. They have been extracted from glassdoor.com.

- **Data scientist**: responsible for data extraction, collection, cleaning, processing in order to analyze, interpret and obtain knowledge. The average salary of a Data scientist is about €45,000.

- **Computer scientist researcher**: it would be the role of a PhD student or a post-doc in the strict responsibilities of research, review, investigation and documentation. The average salary of a Research assistant is €21,000.

Tables 9.4 and 9.5 show the dedication of these four specialists in the project and the salaries and costs associated to their involvement, respectively.

		Dedication (hours)			
Task	**Elapsed (hours)**	**Project Manager**	**Software Engineer**	**Data Scientist**	**CS Researcher**
Research extension	80	0	0	0	80
Basic approaches validation	120	10	80	0	30
Metaheuristics framework	160	20	100	0	40
Create benchmark instances	80	10	30	40	0
Experiments and analysis	120	10	50	40	20
Oral defense	40	10	0	0	30
Total	**600**	**60**	**260**	**80**	**200**

Table 9.4: Specialist dedication by task.

Role	**Elapsed**	**Salary (€/year)**	**Salary (€/h)**	**Cost (€)**
Project manager	60	42,000	21.88	1,312.8
Software engineer	260	31,000	16.15	4,199
Data scientist	80	45,000	23.44	1,875.2
CS Researcher	200	21,000	10.94	2,188
Total	**600**			**9,575**

Table 9.5: Human resources budget.

Indirect costs

These costs are not directly related to an activity or a specific task. For instance, electricity, printed paper, office lease, amortizations, etc. Table 9.6 shows the detailed budget for this kind of costs, including the accumulated amortizations from the hardware budget, not the software, though, because it'll be considered as a direct cost.

Product	**Price**	**Units**	**Cost (€)**
Electricity	€0.129/kWh	1,000kWh	129.0
Internet connection	€25/month	4 months	100.0
Printed paper	€0.05/unit	500 units	25.0
Office lease	€250/month	4 months	1,000.0
Amortizations	-	-	47.4
Total			**1,373.4**

Table 9.6: Indirect costs.

Incidental costs

The main risk in this project is to underestimate its time of development. For this reason it was already considered an extra free week to compensate delays, but it still could happen that we need more time to finish the project. Table 9.7 shows the extra time needed of each role. We have studied to add an extra 10% of the overall time for each specialist, paid with the same salary, not as extra hours.

Role	**Extra (h)**	**Salary (€/h)**	**Cost (€)**
Project manager	6	21.88	131.28
Software engineer	26	16.15	419.9
Data scientist	8	23.44	187.52
CS Researcher	20	10.94	218.8
Total	**60**		**957.5**

Table 9.7: Incidental costs.

Nonetheless, it is important to take in mind that these added costs need to be considered only in the worst case, thus we estimate the probability of the worst case scenario in 15%. So the final cost of the incidentals is €143.6.

Budget by task

Table 9.8 shows the estimated cost for each task taking into account the direct costs only. This way we can distinguish the cost linked to each task.

Task	**Hardware (€)**	**Software (€)**	**Human (€)**	**Total (€)**
Research extension	105.3	9.6	875.2	**990.1**
Basic approaches validation	158.7	14.4	1,839	**2,012.1**
Metaheuristics framework	210.6	19.2	2,490.2	**2,720.0**
Create benchmark instances	105.3	9.6	1,640.9	**1,755.8**
Experiments and analysis	158.7	14.4	2,182.7	**2,355.8**
Oral defense	52.9	4.8	547	**604.7**
Total	**790**	**72**	**9,575**	**10,438.5**

TABLE 9.8: Cost by task.

Total budget

And finally the total budget including all costs and applying a level of contingency of 10% is shown in table 9.9.

Concept		**Cost (€)**
Direct costs		**10,438.5**
	Hardware	**790.0**
	Software	**72.0**
	Human	**9,575**
Indirect costs		**1,373.4**
Subtotal		**11,811.9**
Contingency (10%)		**1,181.2**
Incidentals		**143.6**
Total		**13,136.7**

TABLE 9.9: Total budget.

Budget monitoring

The budget isn't fixed in the sense that needs to be adjusted according to justified deviations of the planning of this project. For instance, needs of different specialists might arise during its development. However this would only imply a simple adjustment in the direct costs of the project (specifically in the human resources costs). Furthermore, if that free week scheduled in the end of the planning was necessary we would only need to add direct and indirect costs to this budget. And additionally, we have already established a contingency cushion of 10% of the overall budget as well as the cost of incidentals, such as extra hours needed beyond the free week.

It is obvious that if the budget estimation exceeds the actual cost of the project the difference has to be discounted and that surplus could be used to maintain the project, however, hardly ever this is the case.

The monitoring of the budget might be easier to be done using a management application, like Team Foundation Server, where we can keep updated the time spent on each task. This way we can control the direct costs (human related) just by adjusting the hours worked by each specialist.

9.3 Modifications and deviations

The simheuristics and learnheuristics frameworks were removed during the late development of this project. It seemed convenient at that time to focus on the other three solving approaches and also on performing a rigorous design of experiments with realistic instances. The main problem we had to face was the lack of standardized benchmark instances in the literature to test our algorithms, as the exact same problem hasn't yet been addressed to the best of our knowledge. So the planning and the budget had to be modified accordingly to these changes. A part from that the rest of this chapter stays the same.

It is also important to state that, both methodologies would have solved a different more complex problem than the one we are dealing with now in container transportation. In both cases we would be considering stochastic traveling times and in the second case, we would also be implementing techniques that were defined this very last year for a really too complex problem. Maybe we were a bit naive to think that we would be able to do so much in the limited time that we dispose, as it is the usual case in this kind of projects.

In addition, we have to remark that we have been writing in parallel a scientific paper to submit in a journal. So most of that work has been adapted to this format, which made it incredibly harder, as many aspects are assumed when referenced in a journal article. However, wider explanations and an explicit chapter dedicated to contextualize and give background on the physical internet as well as optimization methods was written in this thesis.

9.4 Legal aspects and regulations

While writing this chapter, regulations related to the price of the kWh, average office lease prices and average wage in Spain of the four professionals mentioned earlier, had to be consulted. But no more legal aspects were relevant in this chapter.

Nevertheless, during the development of this thesis a specific regulation concerning the hours of service for property-carrying drivers established in the U.S. [69], was taken into consideration. As explained in Chapter 2, the physical internet aims to improve the working conditions of the employees in the logistics sector, particularly for truck drivers, who have incredibly long working shifts. For that reason, we had to understand the main rules under this usual regulation that not only prevails in the U.S.

Chapter 10

Sustainability and social commitment

In this chapter the impact of our project is discussed through the three sustainability dimensions and its development taken into consideration from these three perspectives.

Environmental dimension

From the environmental perspective, the resources needed to develop this project are really scarce, and they are limited to the resources mentioned above, thus seems to be in line with a so called sustainable project. The only environmental impact that these activities wouldn't had if this TFG hadn't been done would be the saving of resources such as paper, electricity and the use of computers and an Internet connection. On the other hand, four people wouldn't had a job during its duration. Because this project is not building any specific product, but instead is doing a study in logistics and trying to propose novel approaches for tackling a problem related to freight transportation, we will focus on the impact that these solutions might have instead of focusing on the tiny impact that its development has to the environment.

The Physical Internet has in one of its fundamental goals to reduce by an order of magnitude the direct and indirect logistics-induced global pollution. Hence, if this project tries to implement algorithms and ways of doing of this paradigm so that they can be adopted by companies, its impact is direct to the society, environmentally speaking, and beneficial for everyone as it aims to reduce the environmental footprint. This study will also start a line of research that can be further studied by the academia, so it can be reused and transformed somehow.

Economic dimension

As it has been stated in section 9.2, the total budget for this project has been estimated around €$13,000$ and its cost adjustments already assessed and taken into account in the budget monitoring, as well as in the budget itself with incidentals and the contingency cushion. So we can ensure with high probability the viability of the project from the economic perspective. However, because this project is done in the framework of the Physical Internet, a bigger contribution in these three dimensions could be achieved with the organization and coordination of different projects on the same topic. It is also relevant to mention that, as it happens with other projects, once it is finished it is easier to plan it with fewer resources, and more is the case of a final degree project done by a student whose main goal is to learn. Thus, the budget by task and the temporal planning reflect the difficulty and importance of certain tasks, for instance, the metaheuristics framework.

One of the goals of the PI is also to reduce the global economic burden in global logistics, while unlocking highly significant gains in production, transportation, and business productivity. So the development of this project has a big impact on companies that might implement a more sustainable and efficient logistics system, from an economic point of view.

Social dimension

Right now the political and social situation of the country isn't at its best due to the conflict between Spain and Catalonia. However this shouldn't affect its proper development. Despite that fact, there's a clear need and a big margin of improvement with regard to the logistics sector, not only in Spain but worldwide. And the clearest scenario from a social perspective is the poor working conditions in this sector. The PI aims to improve their working conditions, as well as increase the quality of life of the overall population by making much more accessible the physical objects across the world. If companies start to gradually adopt this paradigm, and the routings presented in this project are taken into consideration, truck drivers, for instance, won't have to face such long periods away from their families.

Sustainability matrix

Table 10.1 shows the sustainability matrix of this project with an overall punctuation of 70 in the range of -60 to 90. We would only like to remark that the environmental dimension has high marks due to the reasons mentioned before with respect to the limited number of resources needed for its development. The rest of marks are justified above on the respective discussion of each dimension.

	PPP	**Useful life**	**Risks**
Environmental	Design consumption	Ecological footprint	Environmental risks
	9 in [0:10]	17 in [0:20]	-2 in [-20:0]
Economic	Bill	Viability plan	Economic risks
	8 in [0:10]	16 in [0:20]	-4 in [-20:0]
Social	Personal impact	Social impact	Social risks
	9 in [0:10]	18 in [0:20]	-1 in [-20:0]
Sustainability range	26 in [0:30]	51 in [0:60]	-7 in [-60:0]
	70 in [-60:90]		

TABLE 10.1: Sustainability matrix.

Chapter 11

Conclusions

Real life transportation problems can drastically scale up as needs in logistics arise by increasing customer demands. In these situations a human expert is only able to route a fleet of trucks with reasonable local choices, that is, taking care of the most urgent shipments, activating the closest drivers and using the shortest paths. However, these solutions are most likely far from being near-optimal if they are feasible at all, and they are inadequate for the huge volumes of shipments that multinational companies are taking care in modal transportation networks throughout the entire world.

In this thesis, a greedy deterministic heuristic algorithm based on discrete-event simulation tries to imitate a human expert planner to solve this novel problem. This algorithm is able to find feasible solutions for all instances minimizing a time-related single-objective function, which is the total time (TT) to deliver all containers and return all drivers. We have also considered the shipping time (ST), an alternative objective function which only aims to minimize the elapsed time to ship all containers. Both functions are separately used in different computational experiments. Although the performance of this heuristic is envious, as it is really fast, 5 seconds at most for large instances (800 nodes, 200 containers), the quality of the solutions can be easily improved.

As an extension to this approach, we randomize the heuristic by introducing well-known biased-randomization techniques and embed it into a multi-start framework (BR-MS). We build numerous solutions until a stopping criterion based on the number of iterations is met and return the best one. The randomization of the heuristic is done in three main points: when selecting (*i*) the next container to be dispatched; (*ii*) the next driver to be activated; and (*iii*) the next node that the container will traverse. With this algorithm, we are able to find better solutions in reasonable times, gaps of approximately 8% in average taking around 2 minutes of computational time, 30 seconds or less for small instances and up to 4 or 5 minutes for larger ones.

In the end, a variable neighborhood search metaheuristic is built (BR-VNS) to address the search in an informed way. This approach is usually faster than the BR-MS, as it partially destroys and resolves solutions without starting from the beginning, but backtracking a certain amount of decisions. Its main strength resides in the fact that these partial routings from high-quality solutions can be further exploited, taking advantage of its underlying base routes. The performance of this algorithm is slightly better than the BR-MS approach for large instances and also faster, but for some other instances it is outperformed by the BR-MS.

11.1 Future work

Increasing reality of the problem

Several lines of research stem from this work. For instance, stochasticity in traveling times could be considered, which would require integrating additional simulation techniques into the metaheuristic-based framework [70], *e.g.*, Monte-Carlo simulation. However, as this problem is really time-dependent, the usual simheuristic techniques would not be applied the same way. If a single traveling time changes in the network, the whole solution might radically change. So innovative approaches would be needed to address stochastic systems as such. We could consider the policies of common sense deviation, that is, the policies for the beta parameters' selection that would minimize the expected cost of a set of different scenarios derived from the same instance, *i.e.*, with alterations on the traveling times only.

It would also be interesting to introduce further realistic constraints such as service times for loading and unloading containers in hubs or considering a limited capacity for hubs. We could also allow delays on the driver's returns, that is, violating in some cases their constraints but adding a guarantee on the minimum no. of drivers that would arrive on time. We could even extend this problem in a wider direction if we consider a continuous system where new container shipment orders arrive at different times with the same goal: dispatch all orders as fast as possible so that we are able to deliver all containers as soon as we can. Then it would make sense to have more than one working shift for drivers, although regulations and limitations on the number of consecutive driving hours would be needed.

In the context of PI, the vision is to integrate Wireless Sensors and RFID technologies, thus creating a network of responsive containers, where optimization systems could re-assign drivers and re-calculate the routes according to real-time data, thus taking efficient decisions as the containers move within the network.

Improvement in our solving approaches

With regard to the actual way of solving this problem, that is, the underlying base heuristic in all approaches, different strategies on the first assignments of drivers could be discussed. As the first decisions are radically decisive to the configuration of the solution and the search space explored, it seems quite important to diversify this search by using different methods on the initial assignment, *e.g.*, minimize the total sum of traveling times from driver's depots to containers besides the urgency of the containers.

Right now, when scheduling new events we only take into account the available drivers, *i.e.*, the ones that are idle in the current moment. However, we could also consider all drivers looking at their estimated times of arrival (ETA). For instance, if a driver currently transporting a container has an ETA of a couple of minutes to arrive to a hub which is the closest to another node where a more urgent container resides, we could already schedule an event so that no other further driver is activated, but this one which is the nearest.

Analysis of the results and experiments

Taking a closer look to the parameter fine tuning, there's room for improvement. First of all, the beta parameters are chosen in the interval (0.95, 1.00), which works more than fine. Nonetheless, this interval could be separately optimized for each one of the betas, as it is not the same to deviate from the most urgent container than choosing a slower path. We can further diversify the search in one direction than in another.

Results from Table 7.3 suggested that the variable neighborhood search is not well parameterized, or at least, is not performing at its best. We are using a more complex solving approach (BR-VNS) but not obtaining better results in average for small instances, compared to the ones obtained by the BR-MS approach. The simplest explanation points to the tuning of the VNS parameters, that is, the minimum and maximum percentages of destruction, and in a less relevant manner, the step parameter. Different trial-and-error tests ended up setting them to: 20-100% with an increment of 1%. However, seems reasonable to diversify the search in the beginning and intensify it as the time passes by, *i.e.*, start with three high values of pmin, pmax and step and let them settle slowly to lower values, just as the temperature parameter does in the simulated annealing metaheuristic (SA).

A more complex study on the interrelation between the beta parameters could be performed with a proper design of experiments that would need to go beyond trial-and-error tests. Although, it was not the aim of this thesis to spend too much time in these experiments. That's why we consider it important to be addressed as future work.

For all algorithms, we have used the no. of iterations as the stopping criteria when performing our experiments. This choice was a meditated one, because we wanted to see the differences in gaps between all instances, knowing that the difficulty to solve them varies, as we have created scenarios with a no. of containers ranging from 25 to 200. But to compare the performance of both algorithms (BR-MS and BR-VNS), even though seemed a fair comparison to experiment with 200 and 300 iterations respectively, a criteria based on the computational time would be more convenient. Hence, seems necessary to do more experiments changing the stopping criteria. In addition, a proper comparison between both approaches with statistical hypothesis testing could be done. An analysis of variance (ANOVA) would be a reasonable testing to perform on the mean gaps, so that we could statistically prove that one of them performs better in average. However, in this problem we are more interested in the best results rather than the average performance, as we are seeking near-optimal solutions. We prefer an algorithm that gives one really good solution and four other solutions quite horrible, than another one that gives five average good solutions. This tests could also be used to compare mean computational times, which then would statistically prove that an algorithm is faster than the other.

Even though we've been able to assess the complexity of single parts of our algorithms, *e.g.*, the complexity of the shortest paths computation, a complexity analysis could be further addressed in the future. It was recently studied a methodology to give algorithm's complexity bounds for a given family of graphs, as if we were stating that this algorithm would perform in this range of complexities for these set of structures or scenarios.

So far we've only considered single-objective functions, *i.e.*, the total time and the shipping time, separately. But multi-objective functions could also be built taking into account different characteristics of the solution (drawn in Figure 8.7), *e.g.*, the total traveling time, the total elapsed time of drivers or the number of drivers moved. This way more balanced solutions could be found out and compared to others. Typical multi-objective functions include the weighted sums or the *pareto* optimization.

Alternative approaches

Learnheuristics was one of the main techniques that this thesis was pursuing in the beginning: the hybridization of metaheuristics and machine learning (ML). Excuses notwithstanding, the complexity of this problem didn't leave time to develop this framework, and it wasn't until the last month that we understood that this methodology should be first applied to a simpler problem such as the plain VRP. And we are actually doing so in the ICSO research group. However, once more research is developed in this direction and more case examples are published it could be considered to generate models for dynamic traveling times that change according to intrinsic and extrinsic variables, such as traffic, meteorology or driver fatigue. Online approaches where models were refined during the construction of the solution could also be considered.

Exact methods were discarded in the first place, but the need to obtain optimal solutions for small instances only seems possible using algorithms such as Simplex. For this reason a mathematical model would have to be defined and it hasn't been an option from the beginning. Still, tackling this complex task could be addressed in the future. Otherwise we are not able to compare our solutions with the optimal and all we can do is compare solutions between them, which in the end we don't know how far are from the optimal.

Other metaheuristics could be developed for this problem, even though the VNS choice seems quite the right fit. But of course, having this vast range of great metaheuristics one is tempted to try another one, if there's a reasonable explanation behind that choice.

11.2 Competences justification

CCO1.1: *To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements.* [*In depth*]

The problem at issue has been contextualized in the combinatorial optimization field, similar routing problems have been reviewed and three different algorithmic strategies were discussed, analyzed and developed to solve it. Their performances have been compared through a set of experiments carried out on our own standardized realistic instances, adapted from a different problem.

CCO2.1: *To demonstrate knowledge about the fundamentals, paradigms and the own techniques of intelligent systems, and analyse, design and build computer systems, services and applications which use these techniques in any applicable field.* [*In depth*]

A wide background on heuristics and metaheuristics has been given to support the further development of these techniques. The field of metaheuristics is closely related to artificial intelligence, as the former can be seen as an individual agent. In [71] is reported that *multi-agent systems with metaheuristic agents provide effective strategies for solving difficult optimization problems.* Related literature proposed an organizational multi-agent framework to hybridize metaheuristics algorithms.

CCO2.2: *Capacity to acquire, obtain, formalize and represent human knowledge in a computable way to solve problems through a computer system in any applicable field, in particular in the fields related to computation, perception and operation in intelligent environments.* [*Quite developed*]

The greedy deterministic heuristic developed to solve the problem faced in this thesis tries to imitate how a human would behave and solve it taking the most reasonable choices at each moment, without foreseeing the subsequent consequences of his choices. [72] states that early developed metaheuristics "*can be characterized under the umbrella term of artificial intelligence because they involve mimicking human problem-solving behavior and learning lessons from this behavior on a more abstract level*".

CCO2.4: *To demonstrate knowledge and develop techniques about computational learning; to design and implement applications and system that use them, including these ones dedicated to the automatic extraction of information and knowledge from large data volumes.* [*In depth*]

The variable neighborhood search metaheuristic takes advantage of underlying good base structures of solutions to exploit them and learn a way to improve them. However, computational learning is directly related to machine learning and, in the end, we haven't added a learnheuristics framework in this thesis, even though we are now working on that for a different problem. So this competence should have been changed to *quite developed*, as we decided not to include any background and results related to it.

Appendix A

Network visualizations

A.1 Small test instances

In this section, the small test instances used for testing and debugging purposes will be presented with a visualization of the network (including the traveling times/weights), specifying the no. of drivers, no. of containers, the shipments (origin, destination, due time) and the working shift limits. The format used for all them is explained below:

- For each driver, we detail his characteristics as $di : n\ (t)$, where i stands for the driver id, n is his depot number and t is his working shift.
- For each container, we specify its shipment information as $ci : o \rightarrow d\ (t)$, where i stands for the container id, o is the origin, d is the destination and t, its due time.

If all drivers working shift or containers due time are the same, they will be separately specified once. Otherwise the above format will be followed.

Instance 1

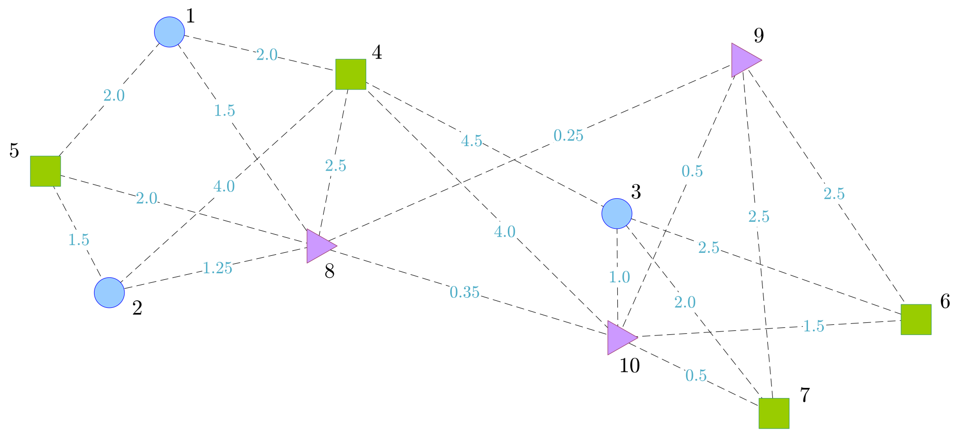
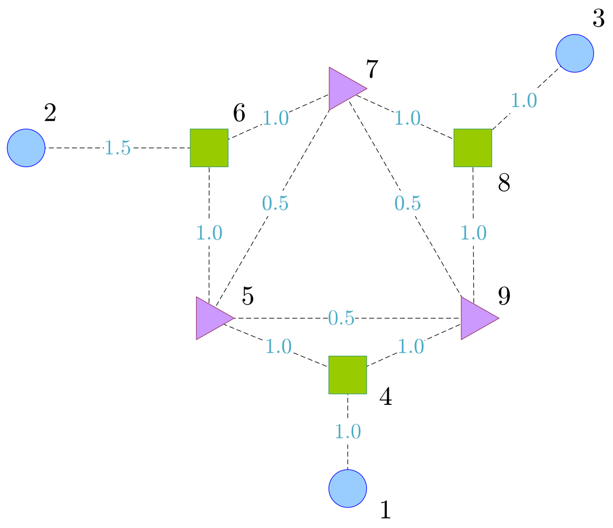


Figure A.1: Network representation of small test instance-01.

No. of drivers: 3	Drivers working shift: $8h$	*Drivers*	$d1 : 1, \quad d2 : 3, \quad d3 : 2$
No. of containers: 2	Containers due time: $8h$	*Containers*	$c1 : 4 \rightarrow 6, \quad c2 : 5 \rightarrow 7$

Instance 2



No. of drivers: 3

No. of containers: 3

Drivers

$d1 : 1\ (6h)$

$d2 : 2\ (12h)$

$d3 : 3\ (4h)$

Containers

$c1 : 4 \rightarrow 6\ (6h)$

$c2 : 6 \rightarrow 8\ (8h)$

$c3 : 8 \rightarrow 4\ (6h)$

FIGURE A.2: Network representation of small test instance-02.

Instance 3

No. of drivers: 6

No. of containers: 6

Drivers working shift: $9h$

Containers due time: $7.5h$

Drivers

$d1 : 1$

$d2 : 1$

$d3 : 7$

$d4 : 7$

$d5 : 10$

$d6 : 10$

Containers

$c1 : 2 \rightarrow 8$

$c2 : 3 \rightarrow 8$

$c3 : 6 \rightarrow 9$

$c4 : 8 \rightarrow 2$

$c5 : 9 \rightarrow 6$

$c6 : 9 \rightarrow 3$

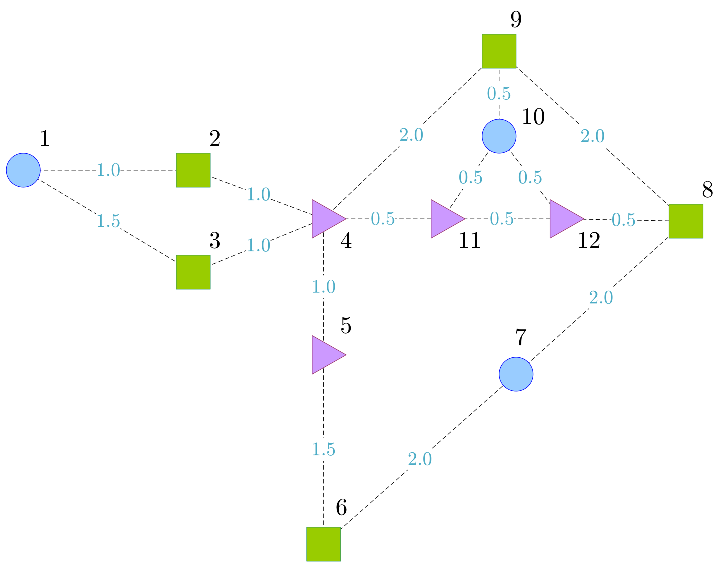


FIGURE A.3: Network representation of small test instance-03.

Instance 4

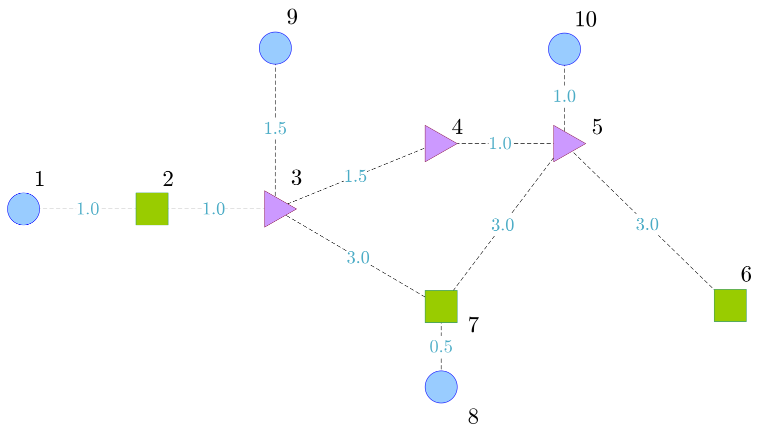


FIGURE A.4: Network representation of small test instance-04.

No. of drivers: 5
No. of containers: 2
Drivers working shift: $8h$

Drivers
$d1 : 1$
$d2 : 9$
$d3 : 8$
$d4 : 10$
$d5 : 10$

Containers
$c1 : 2 \rightarrow 6\ (16h)$
$c2 : 7 \rightarrow 6\ (8h)$

Instance 5

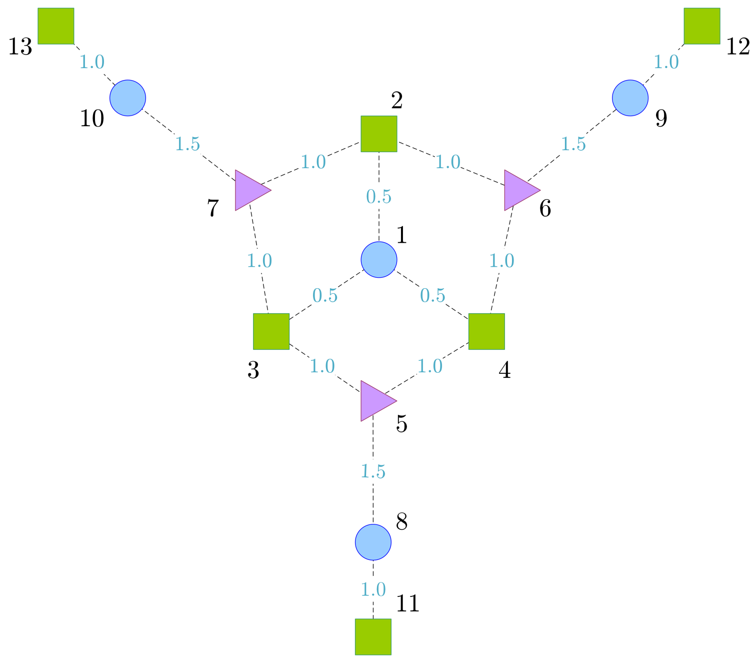


FIGURE A.5: Network representation of small test instance-05.

No. of drivers: 12
No. of containers: 9
Drivers working shift: $8h$
Containers due time: $8h$

Drivers
$d1 - d6 : 1$
$d7 - d8 : 10$
$d9 - d10 : 8$
$d11 - d12 : 9$

Containers
$c1 : 2 \rightarrow 11$
$c2 : 3 \rightarrow 12$
$c3 : 4 \rightarrow 13$
$c4 : 13 \rightarrow 11$
$c5 : 11 \rightarrow 12$
$c6 : 12 \rightarrow 13$
$c7 : 2 \rightarrow 12$
$c8 : 3 \rightarrow 13$
$c9 : 4 \rightarrow 11$

Instance 6

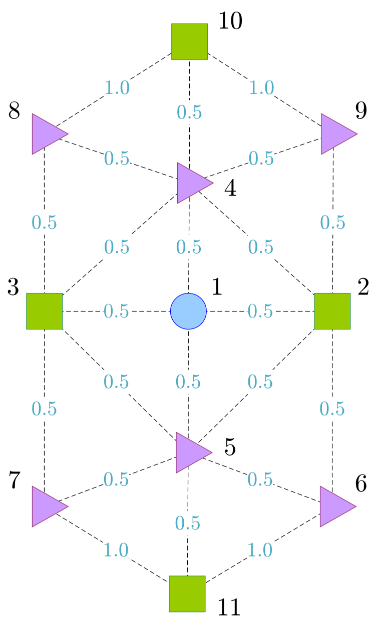


FIGURE A.6: Network representation of small test instance-06.

No. of drivers: 6
No. of containers: 8
Drivers working shift: $8h$
Containers due time: $7h$

Drivers
$d1 - d6 : 1$

Containers
$c1 - c3 : 2 \to 10$
$c4 - c6 : 3 \to 11$
$c7 : 10 \to 2$
$c8 : 11 \to 3$

Instance 7

No. of drivers: 40
No. of containers: 24
Drivers working shift: $12h$
Containers due time: $10h$

Drivers
$d1 - d10 : 9$
$d11 - d20 : 10$
$d21 - d30 : 11$
$d31 - d40 : 12$

Containers
$c1 - c2 : 1 \to 2$
$c3 - c4 : 1 \to 3$
$c5 - c6 : 1 \to 4$
$c7 - c8 : 2 \to 1$
$c9 - c10 : 2 \to 3$
$c11 - c12 : 2 \to 4$
$c13 - c14 : 3 \to 1$
$c15 - c16 : 3 \to 2$
$c17 - c18 : 3 \to 4$
$c19 - c20 : 4 \to 1$
$c21 - c22 : 4 \to 2$
$c23 - c24 : 4 \to 3$

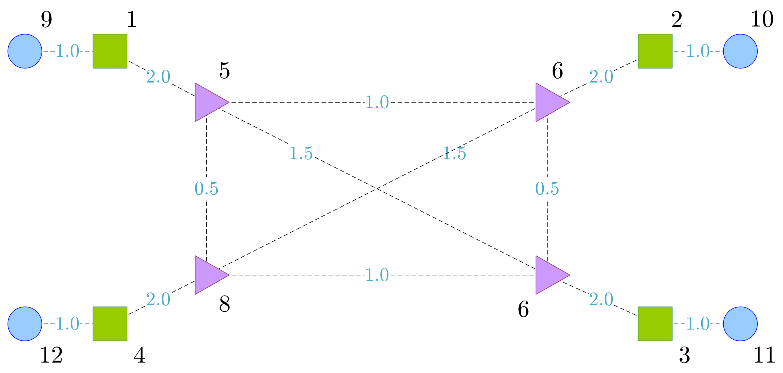


FIGURE A.7: Network representation of small test instance-07.

Instance 8

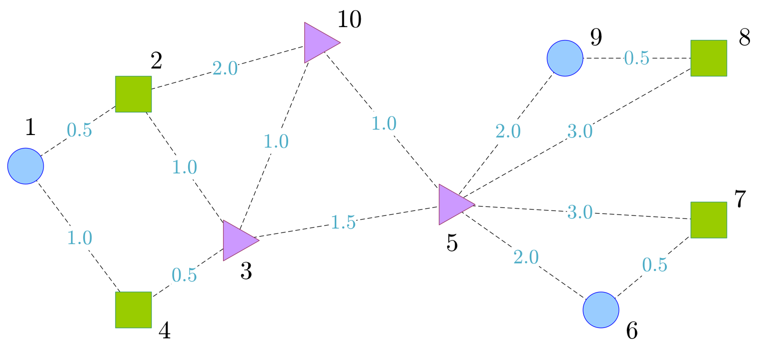


FIGURE A.8: Network representation of small test instance-08.

No. of drivers: 3
No. of containers: 2
Drivers working shift: $8h$

Drivers

$d1 : 9$

$d2 : 1$

$d3 : 6$

Containers

$c1 : 8 \to 4\ (6h)$

$c2 : 2 \to 7\ (9h)$

Instance 9

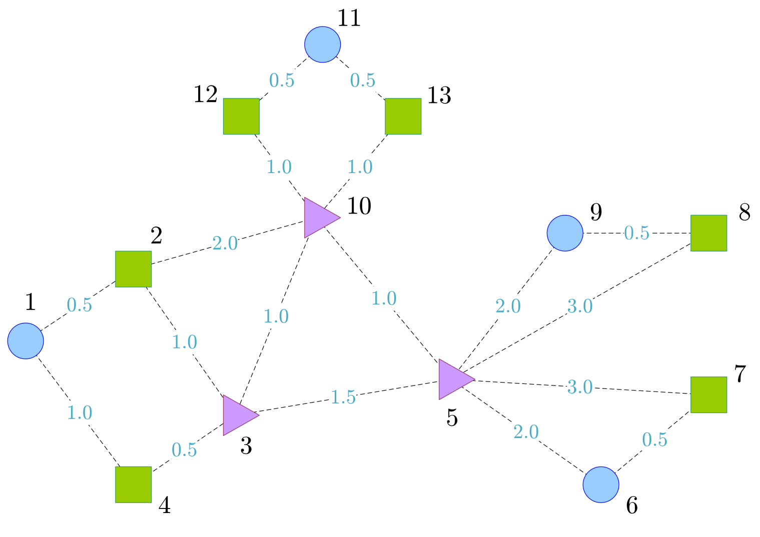


FIGURE A.9: Network representation of small test instance-09.

No. of drivers: 8
No. of containers: 4
Drivers working shift: $8h$

Drivers

$d1 : 1$

$d2 : 1$

$d3 : 6$

$d4 : 6$

$d5 : 9$

$d6 : 9$

$d7 : 11$

$d8 : 11$

Containers

$c1 : 2 \to 7\ (6h)$

$c2 : 8 \to 4\ (9h)$

$c3 : 12 \to 4\ (7h)$

$c4 : 13 \to 7\ (5.5h)$

Instance 10

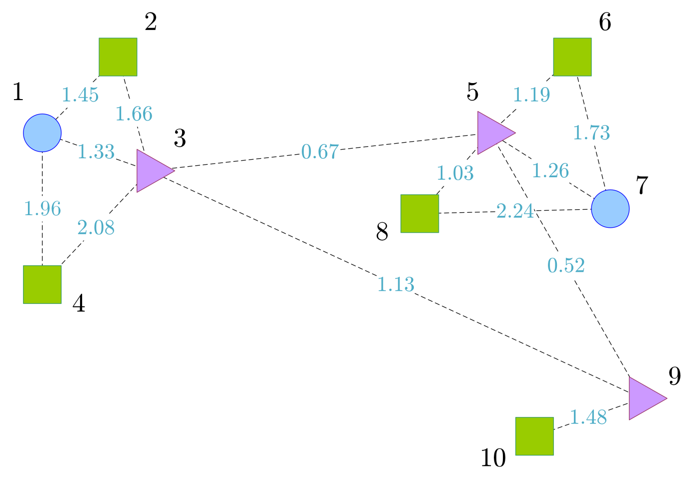


FIGURE A.10: Network representation of small test instance-10.

No. of drivers: 16

No. of containers: 14

Drivers working shift: $8h$

Containers due time: $6h$

Drivers

$d1 - 8 : 1$

$d9 - 16 : 7$

Containers

$c1 - c2 : 2 \rightarrow 10$

$c3 : 2 \rightarrow 4$

$c4 - c5 : 4 \rightarrow 6$

$c6 : 4 \rightarrow 8$

$c7 : 6 \rightarrow 4$

$c8 - c9 : 6 \rightarrow 2$

$c10 - c11 : 8 \rightarrow 4$

$c12 : 8 \rightarrow 6$

$c13 - c14 : 10 \rightarrow 8$

A.2 Benchmark standardized instances

Visual representations of the networks from instances *pi-01* to *pi-05*, respectively adapted from instances TCARP-01 to TCARP-05, are shown below in Figures A.11 to A.15. As instances *pi-06* to *pi-20* have the same structure (underlying graph) than these first five, we are not showing them. The only thing that would visually change would be the different disposition of the three types of nodes in the network, as the assignment across instances is different.

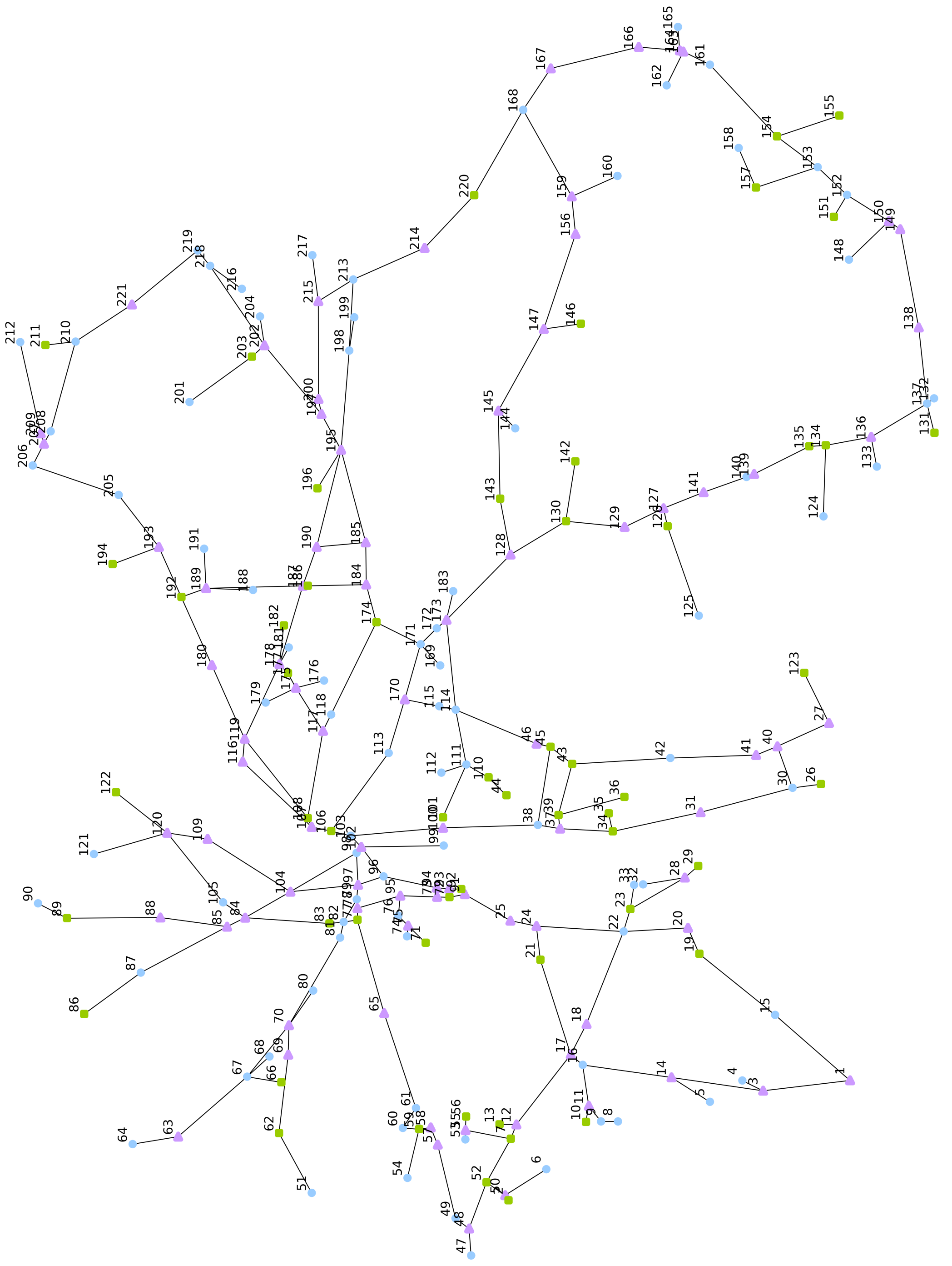


Figure A.11: Network representation of instance pi-01 adapted from TCARP-R1.

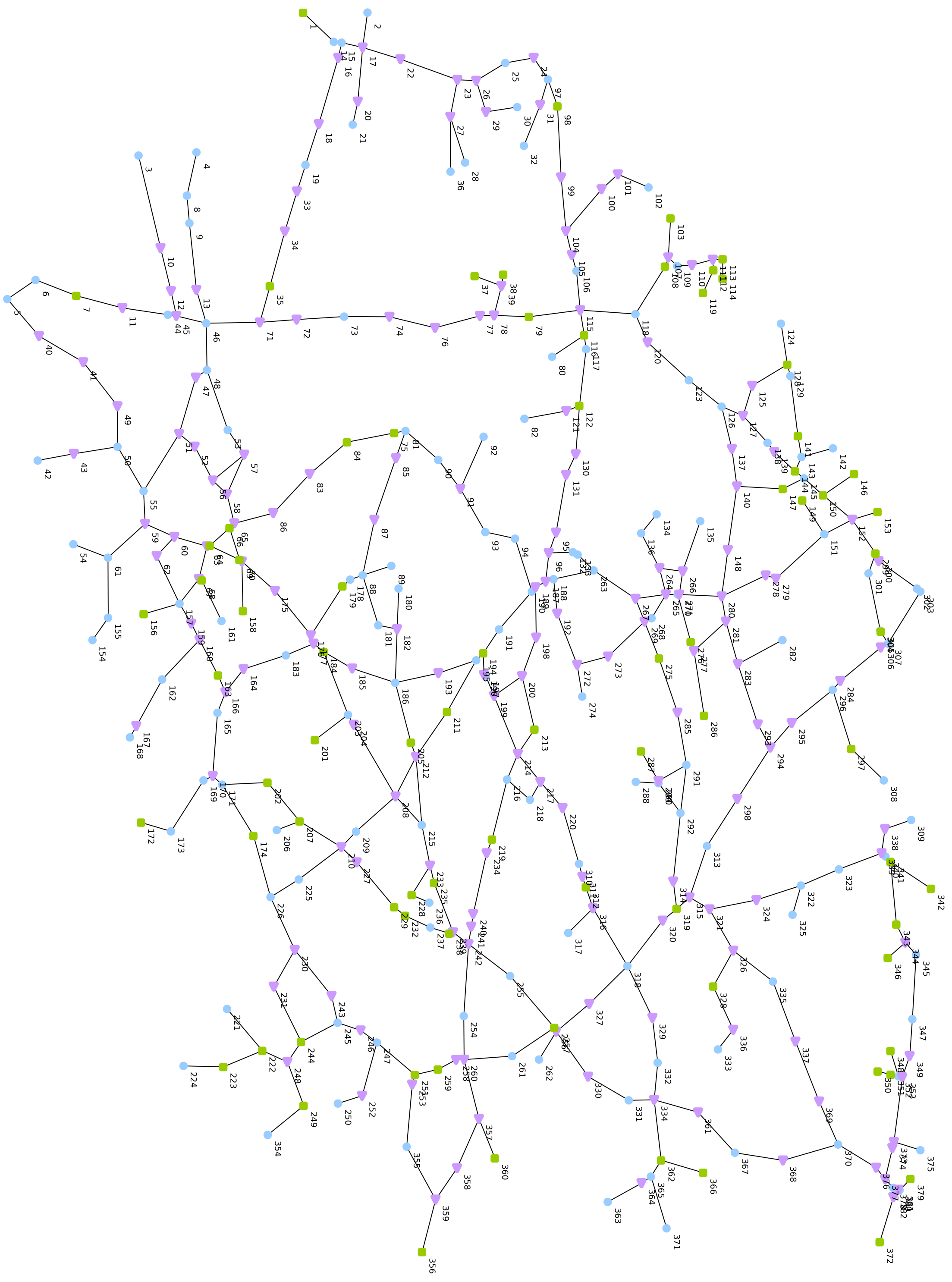


FIGURE A.12: Network representation of instance pi-02 adapted from TCARP-R2.

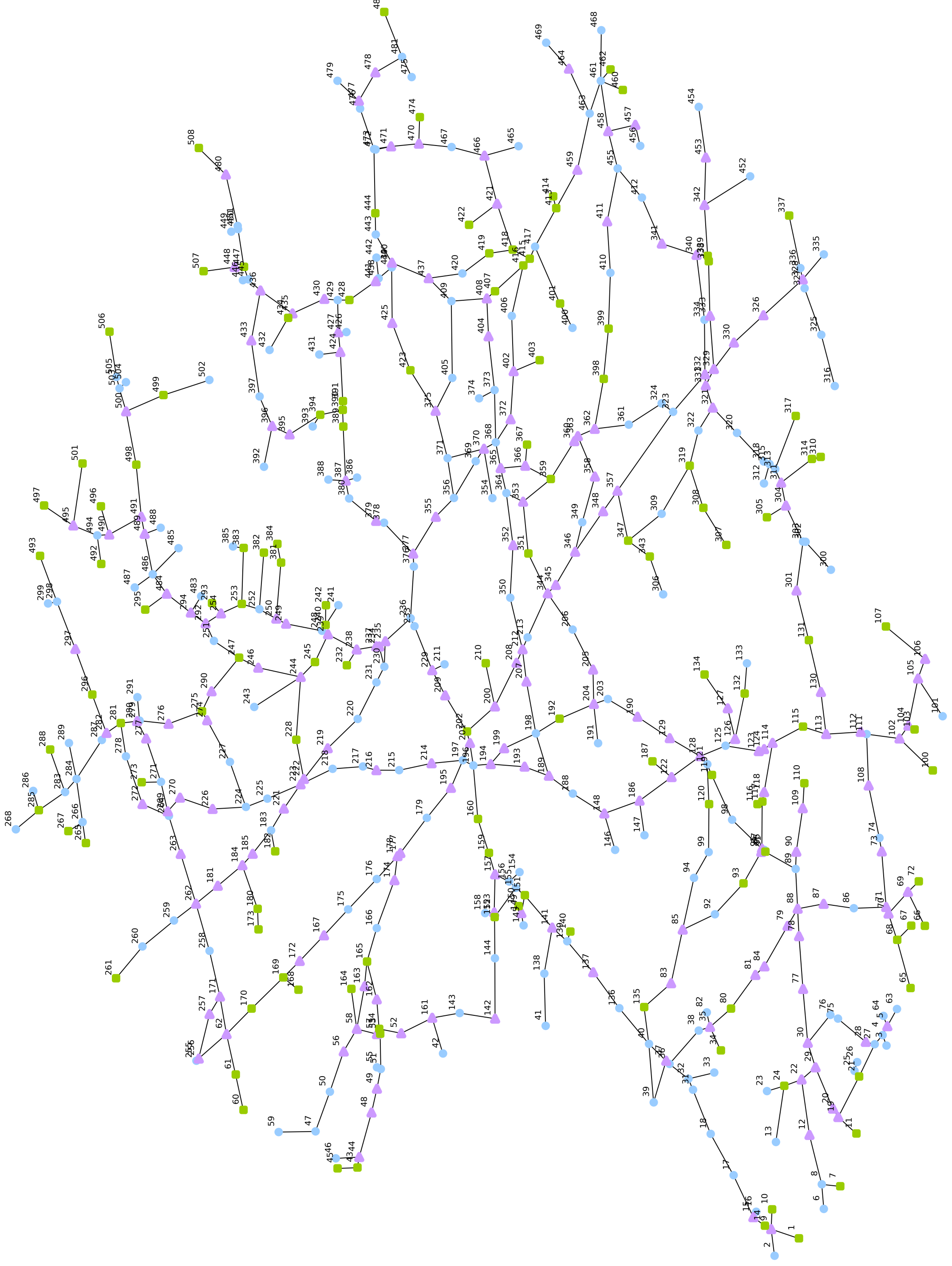


FIGURE A.13: Network representation of instance pi-03 adapted from TCARP-R3.

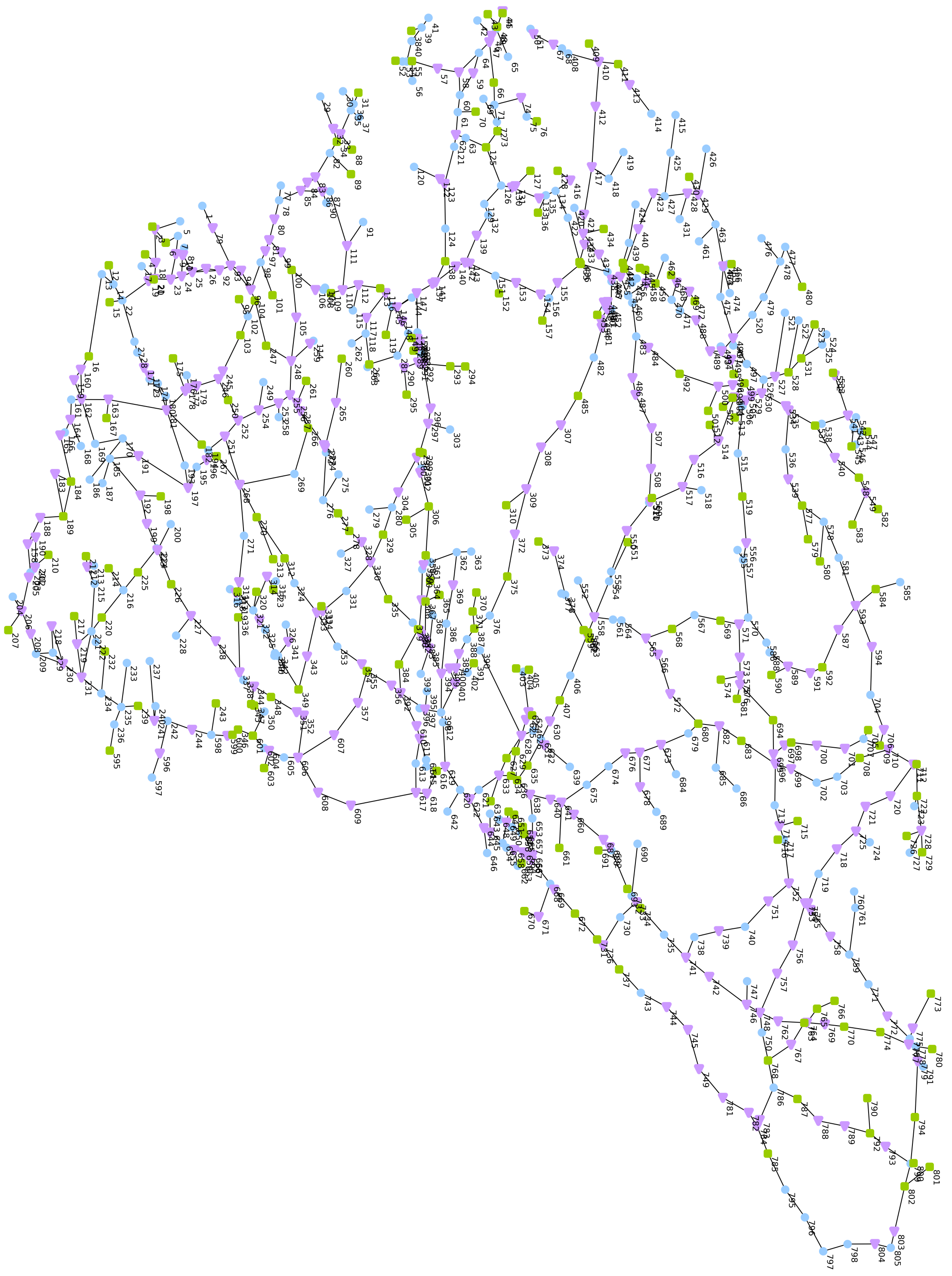


FIGURE A.14: Network representation of instance pi-04 adapted from TCARP-R4.

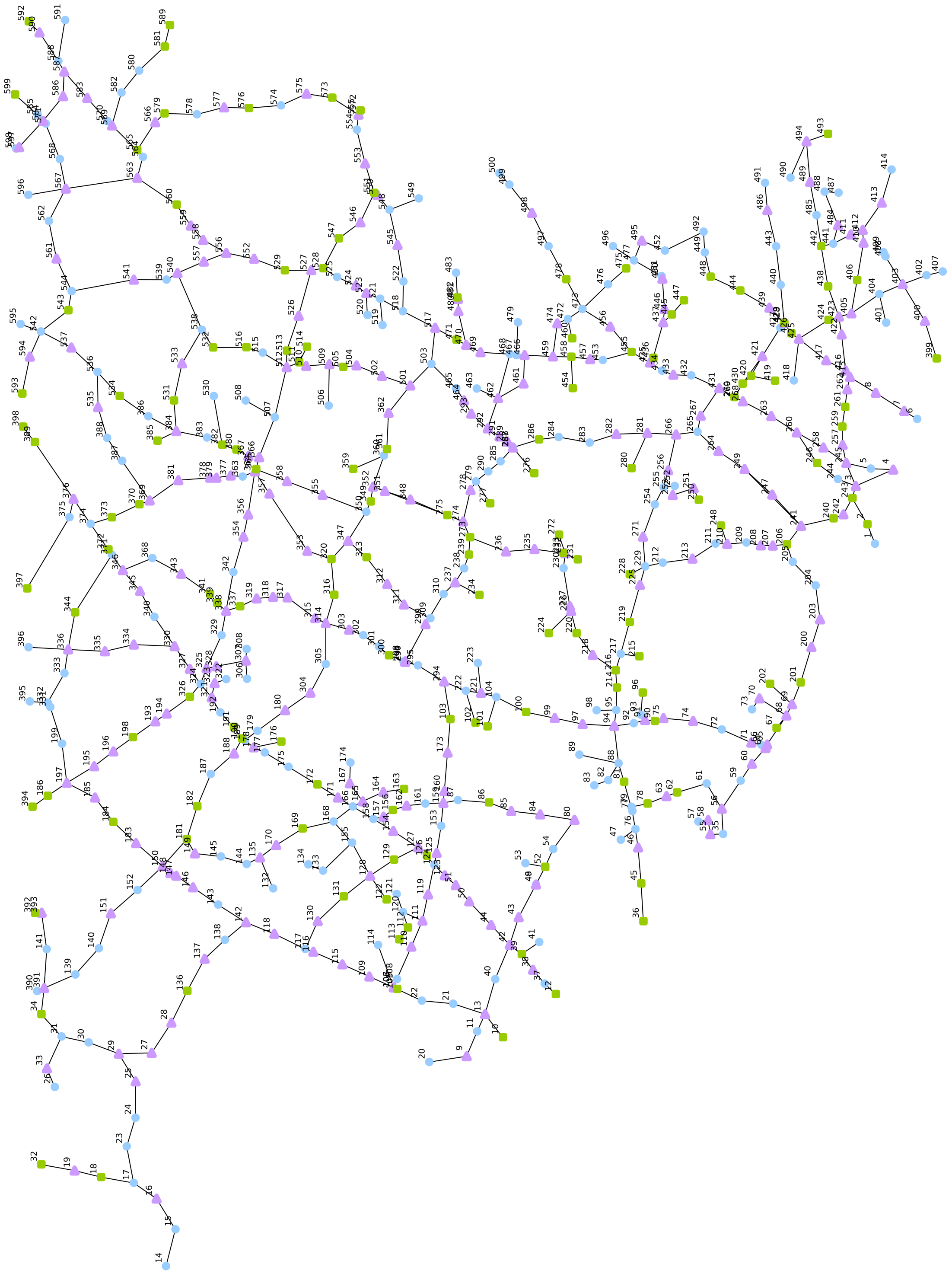


FIGURE A.15: Network representation of instance pi-05 adapted from TCARP-R5.

References

[1] Benoit Montreuil. “The Physical Internet Manifesto”. In: Atlanta, 2012.

[2] Bureau of Transportation Statistics: US Department of Transportation. *Transportation Statistics Annual Report*. Tech. rep. 1 - 222. 2015, p. 226. DOI: 10.2172/1060772. URL: https://books.google.com/books?id=Xohu_3ooPhEC&pgis=1.

[3] Eurostat. *Freight transport statistics - Statistics Explained*. URL: http://ec.europa.eu/eurostat/statistics-explained/index.php/Freight_transport_statistics.

[4] Benoit Montreuil, Russ D Meller, and Eric Ballot. “Towards a Physical Internet: the impact on logistics facilities and material handling systems design and innovation”. In: *Progress in material handling research* (2010), pp. 305–327.

[5] UNFCCC. Conference of the Parties (COP). *Paris Climate Change Conference-November 2015, COP 21*. Tech. rep. December. 2015, p. 32. DOI: FCCC/CP/2015/L.9/Rev.1. URL: http://unfccc.int/resource/docs/2015/cop21/eng/l09r01.pdf.

[6] David Jaffee and David Bensman. “Draying and Picking : Precarious Work and Labor Action in the Logistics Sector”. In: *The Journal of Labor & Society* 19.March (2016), pp. 57–79. ISSN: 10897011. DOI: 10.1111/wusa.12227.

[7] Benoit Montreuil. “Toward a Physical Internet: Meeting the global logistics sustainability grand challenge”. In: *Logistics Research* 3.2-3 (2011), pp. 71–87.

[8] Bureau of Transportation Statistics: US Department of Transportation. “Freight Facts and Figures”. In: *U.S. Department of Transportation/Bureau of Transportation Statistics* (2015), p. 110.

[9] Eric Ballot and Frédéric Fontane. “Reducing greenhouse gas emissions through the collaboration of supply chains: lessons from French retail chains”. In: *Production, plannig & control* 21 (2010), pp. 640–650.

[10] Huib Van Essen. “The Environmental Impacts of Increased International Road and Rail Freight Transport”. In: November (2008).

[11] Helia Sohrabi and Benoit Montreuil. “From private supply networks and shared supply webs to physical internet enabled open supply webs”. In: *IFIP Advances in Information and Communication Technology* 362 AICT (2011), pp. 235–244. ISSN: 18684238. DOI: 10.1007/978-3-642-23330-2{_}26.

[12] Mohammad Javad Zomorodian and Saeed Jamali. *LaTeX: A Document Preparation System*. 2013. URL: https://www.latex-project.org/.

[13] Paul Markillie. *The physical internet*. 2006.

[14] Benoit Montreuil and Russell D Meller. "Physical Internet Foundations". In: (2012). DOI: 10.1007/978-3-642-35852-4{_}10.

[15] B. Montreuil, E. Ballot, and F. Fontane. "An open logistics interconnection model for the physical internet". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)*. Vol. 14. PART 1. IFAC, 2012, pp. 327–332. ISBN: 9783902661982. DOI: 10.3182/20120523-3-RO-2023.00385. URL: http://dx.doi.org/10.3182/20120523-3-RO-2023.00385.

[16] Bureau of Transportation Statistics: US Department of Transportation. *Freight Facts and Figures*. Tech. rep. Washington: Federal Highway Administration, 2009. URL: https://ops.fhwa.dot.gov/freight/freight_analysis/nat_freight_stats/docs/09factsfigures/index.htm.

[17] Bureau of Transportation Statistics: US Department of Transportation. *FAF2 freight traffic analysis*. Tech. rep. Washington: Federal Highway Administration.

[18] Eric Ballot and Frédéric Fontane. "Rendement et efficience du transport: un nouvel indicateur de performance". In: *Revue française de gestion industrielle* 27.2 (2008), pp. 41–55. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-00878313.

[19] C D J Waters et al. "Global Logistics: New Directions in Supply Chain Management". In: Kogan Page Series. Kogan Page, 2010. Chap. 17. ISBN: 9780749457037. URL: https://books.google.es/books?id=Vxx-tgAACAAJ.

[20] Smith L. Johnston. "Consequences of insomnia, sleepiness, and fatigue: health and social consequences of shift work". In: *Medscape CME* (2005). URL: https://www.medscape.org/viewarticle/513572_2.

[21] Merrill M. Mitler et al. "The Sleep of Long-Haul Truck Drivers". In: *New England Journal of Medicine* 337.11 (1997), pp. 755–762. ISSN: 0028-4793. DOI: 10.1056/NEJM199709113371106. URL: http://www.nejm.org/doi/abs/10.1056/NEJM199709113371106.

[22] G. Don Taylor, ed. *Economic impact of logistics*. CRC Press, 2007, p. 640. ISBN: 9781420004588.

[23] T Kale C; Socolofsky. *RFC 1180: A TCP/IP Tutorial*. 1991.

[24] Benoit Montreuil, Eric Ballot, and William Tremblay. "Modular Design of Physical Internet Transport, Handling and Packaging Containers". In: *International Material Handling Research Colloquium* 13 (2015), pp. 978–1. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-01487239.

[25] Yves Sallez, Benoit Montreuil, and Eric Ballot. "On the activeness of intelligent physical internet containers". In: *Studies in Computational Intelligence* 594 (2015), pp. 259–269. ISSN: 1860949X. DOI: 10.1007/978-3-319-15159-5{_}24. URL: http://dx.doi.org/10.1016/j.compind.2015.12.006.

[26] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[27] Mehran Fazili et al. “Physical Internet, conventional and hybrid logistic systems: a routing optimisation-based comparison using the Eastern Canada road network case study”. In: *International Journal of Production Research* 55.9 (2017), pp. 2703–2730.

[28] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN: 0-13-152462-3.

[29] Ivan V Sergienko. *Topical Directions of Informatics*. ISBN: 9781493904754.

[30] I. V. Sergienko, L. F. Hulianytskyi, and S. I. Sirenko. “Classification of applied methods of combinatorial optimization”. In: *Cybernetics and Systems Analysis* 45.5 (2009), pp. 732–741. ISSN: 10600396. DOI: 10.1007/s10559-009-9134-0.

[31] Angel A Juan et al. “A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times”. In: *Simulation Modelling Practice and Theory* 46 (2014), pp. 101–117.

[32] M T Goodrich and Tamassia R. *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0321295358.

[33] T Cormen, C Leiserson, and R Rivest. *Introduction to Algorithms*. 2nd. McGraw-Hill Higher Education, 2000. ISBN: 9780262033848.

[34] Christos H Papadimitriou. “Computational Complexity”. In: *Encyclopedia of Computer Science*. Chichester, UK: John Wiley and Sons Ltd., pp. 260–265. ISBN: 0-470-86412-5. URL: http://dl.acm.org/citation.cfm?id=1074100.1074233.

[35] Herbert S. Wilf. “Algorithms and Complexity Internet Edition , Summer , 1994”. In: *Network* (2003), p. 139. ISSN: 1687-4145. DOI: 10.1155/2008/521407.

[36] Jose Caceres-Cruz et al. “Rich Vehicle Routing Problem: Survey”. In: *ACM Computing Surveys* 47.2 (2014), pp. 1–28. ISSN: 03600300. DOI: 10.1145/2666003. URL: http://dl.acm.org/citation.cfm?doid=2658850.2666003.

[37] G. B. Dantzig and J. H. Ramser. “The Truck Dispatching Problem”. In: *Management Science* 6.1 (1959), pp. 80–91. ISSN: 0025-1909. DOI: 10.1287/mnsc.6.1.80. URL: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.80.

[38] Thomas Weise. *Global Optimization Algorithms. Theory and Application*. Vol. 1. 2009, p. 820. DOI: doi=10.1.1.64.8184. URL: http://www.it-weise.de/projects/book.pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.8184&rep=rep1&type=pdf.

[39] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009. ISBN: 0470278587, 9780470278581.

[40] George Pólya. *How to Solve It*. Princeton University Press, 1945. ISBN: 0-691-08097-6. URL: https://books.google.es/books?id=X3xsgXjTGgoC.

[41] Laura Calvet et al. “Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs”. In: *Open Mathematics* 15.1 (2017), pp. 261–280. ISSN: 23915455. DOI: 10.1515/math-2017-0029.

[42] N Mladenović and P Hansen. “Variable Neighborhood Search”. In: *Comput. Oper. Res.* 24.11 (1997), pp. 1097–1100. ISSN: 0305-0548. DOI: 10.1016/S0305-0548(97)00031-2. URL: http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

[43] Alex Grasas et al. “Biased randomization of heuristics using skewed probability distributions: A survey and some applications”. In: *Computers and Industrial Engineering* 110 (2017), pp. 216–228. ISSN: 03608352. DOI: 10.1016/j.cie.2017.06.019. URL: http://dx.doi.org/10.1016/j.cie.2017.06.019.

[44] Ruta Nadeznikova. *Internet of Things and Transport Systems: a case study of the container networks optimization.* Barcelona, 2017.

[45] Rafael Martí, Mauricio G.C. Resende, and Celso C. Ribeiro. “Multi-start methods for combinatorial optimization”. In: *European Journal of Operational Research* 226.1 (2013), pp. 1–8. ISSN: 03772217. DOI: 10.1016/j.ejor.2012.10.012.

[46] Bob Costello and Rod Suarez. “Truck driver shortage analysis 2015”. In: *American Trucking Associations. Retrieved from http://www. trucking. org/ATA% 20Docs/News% 20and% 20Information/Reports% 20Trends% 20and% 20Statistics/10* 206 (2015), p. 2015.

[47] Shenle Pan et al. *Physical Internet and interconnected logistics services: research and applications.* 2017.

[48] Manuel Díaz, Cristian Martín, and Bartolomé Rubio. “State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing”. In: *Journal of Network and Computer Applications* 67 (2016), pp. 99–117. ISSN: 10958592. DOI: 10.1016/j.jnca.2016.01.010. URL: http://www.sciencedirect.com/science/article/pii/S108480451600028X%5Cnhttp://dx.doi.org/10.1016/j.jnca.2016.01.010.

[49] Julia Funke and Herbert Kopfer. “A model for a multi-size inland container transportation problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 89 (2016), pp. 70–85.

[50] Junliang He, Youfang Huang, and Daofang Chang. “Simulation-based heuristic method for container supply chain network optimization”. In: *Advanced Engineering Informatics* 29.3 (2015), pp. 339–354.

[51] Congli Hao and Yixiang Yue. “Optimization on Combination of Transport Routes and Modes on Dynamic Programming for a Container Multimodal Transport System”. In: *Procedia Engineering* 137 (2016), pp. 382–390.

[52] Ji Ming-Jun and Chu Yan-Ling. “Optimization for Hub-and-Spoke Port Logistics Network of Dynamic Hinterland”. In: *Physics Procedia* 33 (2012), pp. 827–832.

[53] Mingjun Ji et al. “Routing optimization for multi-type containerships in a hub-and-spoke network”. In: *Journal of Traffic and Transportation Engineering (English Edition)* 2.5 (2015), pp. 362–372.

[54] Chuan-xu Wang. “Optimization of hub-and-spoke two-stage logistics network in regional port cluster”. In: *Systems Engineering-Theory & Practice* 28.9 (2008), pp. 152–158.

[55] Günther Zäpfel and Michael Wasner. “Planning and optimization of hub-and-spoke transportation networks of cooperative third-party logistics providers”. In: *International journal of production economics* 78.2 (2002), pp. 207–220.

[56] Chaug-Ing Hsu and Yu-Ping Hsieh. “Routing, ship size, and sailing frequency decision-making for a maritime hub-and-spoke container network”. In: *Mathematical and Computer Modelling* 45.7 (2007), pp. 899–916.

[57] Thomas L. Magnanti and R. T. Wong. “Network Design and Transportation Planning: Models and Algorithms”. In: *Trans. Sci.* 18.January 2016 (1984), pp. 1–55.

[58] Bernard Gendron, Teodor Gabriel Crainic, and Antonio Frangioni. “Multicommodity Capacitated Network Design”. In: *Telecommunications Network Planning.* Ed. by Brunilde Sansò and Patrick Soriano. Boston, MA: Springer US, 1999, pp. 1–19. ISBN: 978-1-4615-5087-7. DOI: 10.1007/978-1-4615-5087-7{_}1. URL: https://doi.org/10.1007/978-1-4615-5087-7_1.

[59] Rochdi Sarraj et al. “Interconnected logistic networks and protocols: simulation-based efficiency assessment”. In: *International Journal of Production Research* 52.11 (2014), pp. 3185–3208.

[60] Robert W Floyd. “Algorithm 97: shortest path”. In: *Communications of the ACM* 5.6 (1962), p. 345.

[61] Pierre Hansen et al. “Variable Neighborhood Search”. In: *Handbook of Metaheuristics* 191.3 (2010), pp. 61–86. ISSN: 978-1-4419-1665-5. DOI: 10.1007/978-1-14419-1665-5. arXiv: 0102188v1 [math].

[62] Michael Creutz. “Microcanonical Monte Carlo Simulation”. In: *Phys. Rev. Lett.* 50.19 (1983), pp. 1411–1414. DOI: 10.1103/PhysRevLett.50.1411. URL: https://link.aps.org/doi/10.1103/PhysRevLett.50.1411.

[63] Peter Keenan. *TCARP large rural datasets.* 2017. DOI: 10.13140/RG.2.2.23723.75043. URL: https://www.researchgate.net/publication/320386608_TCARP_large_rural_datasets.

[64] Dale K Pace. “Verification, Validation, and Accreditation of Simulation Models”. In: *Applied System Simulation: Methodologies and Applications.* Ed. by Mohammad S Obaidat and Georgios I Papadimitriou. Boston, MA: Springer US, 2003, pp. 487–506. ISBN: 978-1-4419-9218-5. DOI: 10.1007/978-1-4419-9218-5{_}21. URL: https://doi.org/10.1007/978-1-4419-9218-5_21.

[65] Omar Baqueiro and Yj Wang. “Integrating data mining and agent based modeling and simulation”. In: *Advances in Data Mining. . . .* (2009), pp. 220–231. ISSN: 03029743. DOI: 10.1007/978-3-642-03067-3{_}18. URL: http://link.springer.com/chapter/10.1007/978-3-642-03067-3_18.

[66] Bernhard Beckert, Ferruccio Damiani, and Dilian Gurov, eds. *Formal Verification of Object-Oriented Software.* Vol. 7421. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-31761-3. DOI: 10.1007/978-3-642-31762-0. URL: http://link.springer.com/10.1007/978-3-642-31762-0.

[67] Donald B. Johnson. “Efficient Algorithms for Shortest Paths in Sparse Networks”. In: *Journal of the ACM* 24.1 (Jan. 1977), pp. 1–13. ISSN: 00045411. DOI: 10.1145/321992.321993. URL: http://portal.acm.org/citation.cfm?doid=321992.321993.

[68] E. W. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. ISSN: 0029-599X. DOI: 10.1007/BF01386390. URL: http://link.springer.com/10.1007/BF01386390.

[69] Federal Motor Carrier Safety Administration. *Hours of Service of Drivers; Final Rule.* 2011.

[70] Angel A. Juan et al. “A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems”. In: *Operations Research Perspectives* 2 (2015), pp. 62–72. ISSN: 22147160. DOI: 10.1016/j.orp.2015.03.001. URL: http://www.sciencedirect.com/science/article/pii/S221471601500007X#!.

[71] Nasser Lotfi and Adnan Acan. “Learning-Based Multi-agent System for Solving Combinatorial Optimization Problems: A New Architecture”. In: *Hybrid Artificial Intelligent Systems: 10th International Conference, HAIS 2015, Bilbao, Spain, June 22-24, 2015, Proceedings.* Ed. by Enrique Onieva et al. Cham: Springer International Publishing, 2015, pp. 319–332. ISBN: 978-3-319-19644-2. DOI: 10.1007/978-3-319-19644-2{_}27. URL: https://doi.org/10.1007/978-3-319-19644-2_27.

[72] Kenneth Sorensen, Marc Sevaux, and Fred Glover. “A History of Metaheuristics”. In: (Apr. 2017), pp. 1–27. ISSN: 14602156. DOI: 10.1093/brain/122.11.2197-a. URL: http://arxiv.org/abs/1704.00853.