ESCOLA DE DOUTORAMENTO
INTERNACIONAL DA USC

Laura
Davila Pena

Tese de doutoramento

Optimization and cooperation with logistics applications

Santiago de Compostela, 2022

Programa de Doutoramento en Estatística e Investigación Operativa

TESE DE DOUTORAMENTO

OPTIMIZATION AND COOPERATION WITH LOGISTICS APPLICATIONS

Laura Davila Pena

ESCOLA DE DOUTORAMENTO INTERNACIONAL
DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

PROGRAMA DE DOUTORAMENTO
EN ESTATÍSTICA E INVESTIGACIÓN OPERATIVA

SANTIAGO DE COMPOSTELA
2022

DECLARACIÓN DA AUTORA DA TESE

Dna. **Laura Davila Pena**
Título da tese: **Optimization and cooperation with logistics applications**

Presento a miña tese, seguindo o procedemento axeitado ao Regulamento, e declaro que:

1) A tese abarca os resultados da elaboración do meu traballo.
2) De ser o caso, na tese faise referencia ás colaboracións que tivo este traballo.
3) Confirmo que a tese non incorre en ningún tipo de plaxio doutros autores nin de traballos presentados por min para a obtención doutros títulos.
4) A tese é a versión definitiva presentada para a súa defensa e coincide coa versión enviada en formato electrónico.

E comprométome a presentar o Compromiso Documental de Supervisión no caso de que o orixinal non estea na Escola.

En **Santiago de Compostela, a 10 de novembro de 2022**

Laura Davila Pena

AUTORIZACIÓN DOS DIRECTORES/TITORA DA TESE

Optimization and cooperation with logistics applications

Dna. Balbina Casas Méndez
D. Ignacio García Jurado

INFORMAN:

Que a presente tese se corresponde co traballo realizado por Dna. Laura Davila Pena, baixo a nosa dirección/titorización, e autorizamos a súa presentación, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como directores/titora desta non incorre nas causas de abstención establecidas na Lei 40/2015.

De acordo co indicado no Regulamento de Estudos de Doutoramento, declaran tamén que a presente tese de doutoramento é idónea para ser defendida en base á modalidade de Monográfica con reproducción de publicacións, nas que a participación da doutoranda foi decisiva para a súa elaboración e as publicacións se axustan ao Plan de Investigación.

En Santiago de Compostela,
a 10 de novembro de 2022

En A Coruña,
a 10 de novembro de 2022

Balbina Casas Méndez

Ignacio García Jurado

A Linda, por acompañarme más de media vida.
A Darwin y Pipo, por su amor incondicional.
A los tres, gracias por hacerme feliz.

Agradecimientos

Me gustaría agradecer, en primer lugar, a mis directores de tesis, Ignacio García Jurado y Balbina Casas Méndez, por haberme guiado a lo largo de este camino. Ignacio, gracias por estar siempre dispuesto a ayudar. Balbina, gracias por los consejos, las conversaciones y el apoyo.

Mi más sincera gratitud a las personas que han contribuido a mejorar esta tesis. A los miembros del comité de seguimiento, Estela Sánchez, José Alonso Meijide y Julián Costa, por su compromiso a lo largo de estos años. Gracias a Arantza Estévez Fernández y José Fernando Oliveira, por haber revisado el documento y por los comentarios aportados. *Un recoñecemento especial a David Rodríguez Penas. David, moitas grazas por estar presente en toda esta etapa e pola axuda brindada.*

Quiero expresar también mi agradecimiento a todos los compañeros de la Facultad de Matemáticas, especialmente a los del Departamento de Estadística, Análisis Matemático y Optimización. A Pedro, Mercedes y Balbina, porque no podría haber empezado esta aventura con mejores compañeros de asignatura. Gracias Pedro por acompañarme en mi primera clase y por estar pendiente de mí en todo momento.

Me siento afortunada por haber compartido estos años con personas maravillosas. Gracias a mis compañeros de la sala π, me habéis acompañado con risas y necesaria distracción. Gracias a todas aquellas compañeras a las que he tenido la suerte de conocer en congresos, sois una inspiración para mí, un ejemplo a seguir y que me hace continuar luchando por la carrera que he elegido.

Gostaria também de agradecer a José Fernando Oliveira e Maria Antónia Carravilla por me acolherem na Universidade do Porto. José Fernando, muito obrigada por me fazeres sentir sempre em casa. María Antonia, muito obrigada pela tua bondade e gentileza. Aos meus colegas do INESC TEC agradeço também a vossa companhia durante o meu tempo lá.

I would also like to thank Peter Borm for hosting me at Tilburg University. Peter, I really appreciate your guidance and kindness. To Jop Schouten, for his help and enthusiasm. And I could not forget you, Wout, for being my great support during those three months. We do not know what will happen in the future, but we will certainly have each other.

Y, como no, esto no habría sido posible sin el apoyo de otras muchas personas. A mis amigas, Alicia, Carme, Lore, Vera y Xeila, por acompañarme en este camino y porque sigamos juntas toda la vida. *To Yanna, for being there all along the way.* A Sofía, porque a pesar de la distancia te siento siempre cerca. A Laura, mi valenciana favorita, gracias por aportar felicidad a mi vida. A María y Damián, por ser siempre un apoyo para mí. A Ignacio, por *entenderme* y acompañarme. A Geno, por estar en todo momento, por valorarme y por quererme. A Érika y Bea, mis cuerdas de guitarra, gracias por hacer que mi vida suene armoniosa. A Almu, por todo lo anterior y más, por hacer que siempre salga a flote.

Agradezco finalmente a mi familia, por ser mi pilar fundamental. A la Abuela Ángeles, por enseñarnos cómo ser buena persona. A la Tía y Lino, por estar presentes y por animarme en todo momento. A mi *mini yo*, Nuria, por verte crecer a mi lado, alcanzando siempre tus objetivos, sean los que sean. Una tesis, y todo lo que ello conlleva, no se puede hacer sin **amor**. Ma, Pa, no hay palabras que describan el amor que siento por vosotros. Gracias por apoyarme siempre, por respetarme y por querer lo mejor para mí. No soy nadie sin vuestro cariño. Gracias Álex, porque eres luz en un mundo oscuro. Quiero estar siempre a tu lado, acompañándote y sirviéndonos de faro. Termino agradeciendo a las personas más grandes que conozco, a quienes les debo todo. Porque si alguien sabe de amor, esos sois vosotros: Lelina y Lelino, os quiero.

Laura Davila Pena
Boiro, octubre de 2022

Funding

Research of the author has been supported by the former Spanish Ministerio de Educación, Cultura y Deporte through the contract FPU17/02126. This work has also been supported by the Spanish Ministerio de Economía, Industria y Competitividad and Ministerio de Ciencia e Innovación, through MINECO/AEI grants MTM2017-87197-C3-1-P, MTM2017-87197-C3-3-P, and MCIN/AEI grant PID2021-124030NB-C32, which include support from the European Regional Development Fund (ERDF), and by the Xunta de Galicia, through Competitive Reference Groups (ED431C 2017/38 and ED431C 2021/24). Part of the research done in Chapter 3 was carried out during three visits to the School of Mathematics at the University of Edinburgh and to the Department of Industrial Engineering and Management of the Faculty of Engineering at the Universidade do Porto, supported by grant MTM2017-87197-C3-3-P. Part of the work conducted in Chapters 5 and 6 was carried out during a visit to the Department of Econometrics and Operations Research of Tilburg School of Economics and Management at Tilburg University, supported by the mobility grant EST21/00360 from the Spanish Ministerio de Universidades. The Supercomputing Center of Galicia (CESGA) is acknowledged for providing the computational resources that allowed to run most of the simulations. Finally, Ricardo Cao Abad and the Dirección Xeral de Saúde Pública of Xunta de Galicia are thankfully acknowledged for providing the data sets employed in Chapter 4 of this dissertation.

Contents

List of Acronyms

ALNS	Adaptive Large Neighborhood Search
ALNS-TS	Adaptive Large Neighborhood Search with Tabu Search
BSR	Block Splitting Rule
CESGA	CEntro de Supercomputación de GAlicia
CVRP	Capacitated Vehicle Routing Problem
CW	Clarke–Wright
EGS	Equal Gain Splitting
GENI	GENeralized Insertion
GMS	Graph Machine Scheduling
GPS	Global Positioning System
ICU	Intensive Care Unit
ILP	Integer Linear Programming
ISA	Instance Space Analysis
ITS	Iterated Tabu Search
LNS	Large Neighborhood Search
LS	Local Search
MATILDA	Melbourne Algorithm Test Instance Library with Data Analytics
MC-TTRP	Multi-Compartment Truck and Trailer Routing Problem
MC-VRP	Multi-Compartment Vehicle Routing Problem
MC-VRP-SP	Multi-Compartment Vehicle Routing Problem with Split Pattern
MC-VRPSD	Multi-Compartment Vehicle Routing Problem with Stochastic Demands
MC-VRPTW	Multi-Compartment Vehicle Routing Problem with Time Windows

MCST Minimum Cost Spanning Tree
MCSTP Minimum Cost Spanning Tree Problem
MDVRP Multi-Depot Vehicle Routing Problem
MILP Mixed-Integer Linear Programming
MVR Mixed Vehicle Route

OR Operations Research

PACVRP Partial Accessibility Constrained Vehicle Routing Problem
PALNS-TS Penalized Adaptive Large Neighborhood Search with Tabu Search
PTR Pure Truck Route
PVR Pure Vehicle Route

SD-VRP Site-Dependent Vehicle Routing Problem
SDVRP Split-Delivery Vehicle Routing Problem
SHAP SHapley Additive exPlanations

TC Truck Customer
TS Tabu Search
TSP Traveling Salesman Problem
TTRP Truck and Trailer Routing Problem
TTRPTW Truck and Trailer Routing Problem with Time Windows

US Unstringing and Stringing

VC Vehicle Customer
VNS Variable Neighborhood Search
VRP Vehicle Routing Problem
VRPPD Vehicle Routing Problem with Pickup and Delivery
VRPSD Vehicle Routing Problem with Stochastic Demands
VRPT Vehicle Routing Problem with Trailers
VRPTW Vehicle Routing Problem with Time Windows

XSTTRP eXtended Single Truck and Trailer Routing Problem

Chapter 1

Introduction

This chapter aims to provide an introductory overview of the present thesis. First, Section 1.1 gives a brief summary of the current use of Operations Research and game theory in the field of logistics, focusing on problems similar to those addressed in this dissertation. Section 1.2 describes the objectives of each of the investigated problems, as well as the organization of the manuscript. Finally, the methodology adopted for the current thesis is presented in Section 1.3.

Contents

1.1 Operations Research, game theory, and logistics

Operations Research (OR) is a discipline in which advanced analytical methods are developed and used to solve problems and aid in decision-making. There are a wide variety of fields where OR can be applied, one of which is logistics. This area is responsible for managing the flow of goods, people, or resources between different locations, in compliance with certain requirements. It also extends to the management and scheduling of personnel or resources in several domains.

Route planning problems are one of the most well-known and studied logistics problems, with Dantzig et al. (1954) as the seminal work introducing the traveling salesman problem (TSP). The subsequent generalization to vehicle routing problems (VRPs, Dantzig and Ramser, 1959), opened up a field of research that has grown exponentially in recent decades. The diverse nature of transportation problems leads to the incorporation of more or less complex constraints, resulting in a wide variety of models and objectives to be pursued. For instance, Carpente et al. (2010) considers a decision problem that arises in an agricultural cooperative where the routes of harvesters need to be planned according to the requirements

of the partners. Amorim et al. (2014) and Neves-Moreira et al. (2020) study real-world VRPs that include novel aspects: the former focuses on a food distribution problem encountered by a Portuguese company that must take into account the transportation conditions of each product, and the latter analyzes a VRP where the price of fuel comes into play and, consequently, refueling decisions need to be made. Méndez-Fernández et al. (2020) addresses a route and schedule planning problem in a home care business. Santos et al. (2020) tackles a rich VRP in which environmental and social objectives are incorporated. More recently, Amorosi et al. (2021) discusses a drone route optimization problem, for which a mathematical formulation and a metaheuristic solution are proposed. Gayialis et al. (2022) studies a problem of goods delivery in urban areas and develops an information system that integrates the OR algorithms implemented to assist companies in logistics operations. Giménez-Palacios et al. (2022) analyzes a VRP with packing constraints under disruption in the context of first-mile logistics parcel pickup and de Morais et al. (2022) tackles a VRP variant in the domain of smart waste collection. Speranza (2018) reviews the historical contributions of OR to transportation and logistics problems.

In light of the above, it becomes evident that the research on routing problems is a topic of great interest and activity at the present time. In this thesis, we will investigate, among other subjects, a rich VRP motivated by a real situation arising in an agricultural cooperative. Following the methodology adopted in most of the aforementioned works, we will first introduce the model to be considered and then propose a mathematical formulation and solving algorithms. In Section 1.2 we will further elaborate on the objectives and contents of this research.

The previously cited papers focus mainly on the optimal design of a set of routes, but logistics problems are also often treated by the practitioners of OR from a cooperative standpoint. Once the routes or networks are created, the question of how to distribute the costs among the agents involved arises. Just to give a few instances, Algaba et al. (2019a) considers a savings sharing problem in a public transport system and Estañ et al. (2021) addresses the allocation of a tram line's fixed cost among the municipalities it operates through. The present dissertation also studies a novel OR problem in which the cost of a certain network must be divided between the customers that compose it. This problem is motivated by a logistics situation of water supply in urban areas, and it will be tackled both from the optimization and allocation point of view. Section 1.2 gives a more detailed description of this study.

The cooperative aspect of logistics problems can be approached through game theory, which is precisely the methodology taken in the two papers just mentioned. This discipline can also be applied in many other domains, with machine learning standing out in recent

years. Works such as Merrick and Taly (2020), Smith and Alvarez (2021), and Chen et al. (2022) highlight the growing trend of bringing these two subjects together, mainly using game theoretical solutions to improve the interpretability of complex machine learning models. In this way, another of the research lines conducted in this thesis combines these two fields. Driven by the pandemic situation derived from COVID-19, a method is proposed to evaluate the influence that certain characteristics of infected patients have on their evolution. Although this problem does not directly address a logistics application, it does serve as a decision support tool for medical professionals. As already stated, resource management is one of the practices encompassed by logistics, and constitutes a key task in the healthcare environment. Rais and Viana (2011) reviews numerous applications of OR in healthcare, including management and logistics. The following section provides more detail on the contents of our work.

1.2 Objectives and manuscript distribution

The main goal of this thesis is to address several OR and game theoretical problems that have or may be related to applications in the field of logistics. The objectives and contents of the manuscript are described as follows:

Part I. On rich vehicle routing problems. The vehicle routing problem (VRP) is a combinatorial optimization problem that aims to find an optimal route configuration traveled by multiple vehicles to service a set of customers. This part of the manuscript focuses on a specific variant of the VRP, the multi-compartment truck and trailer routing problem (MC-TTRP) and it is composed of two chapters that present the model and solution algorithms.

Chapter 2. Multi-compartment truck and trailer routing problem (MC-TTRP). Chao (2002) analyzes a VRP with accessibility constraints, where the fleet of vehicles consists of trucks and trailers. This kind of restriction often appears in agricultural logistics problems, which may also present other requirements such as the demand for different types of products. Throughout this chapter, we study the problem resulting from combining these two aspects: the MC-TTRP. In particular, we propose a mathematical formulation for this model and present a real application within the aforementioned field. This chapter is mainly based on Davila-Pena et al. (2023).

Chapter 3. Heuristics for the multi-compartment truck and trailer routing problem. The exact solving of the MC-TTRP was demonstrated to be difficult in Chapter 2, which makes it therefore necessary to resort to approximate solution methods. In this chapter, we

propose different two-phase heuristic algorithms to solve the MC-TTRP. A computational study is conducted to analyze the performance of our proposals. This chapter is mainly based on Davila-Pena et al. (2023) and Davila-Pena et al. (2022c).

Part II. On game theory in classification problems. A classification problem involves identifying the category to which a given individual belongs, based on their attributes. In this type of problem, in addition to classifying, it is important to know how influential each of the features was on the response. This second part of the manuscript is composed of a single chapter that presents an influence measure grounded on game theory ideas.

Chapter 4. An influence measure for classification problems. A method based on the Shapley value of cooperative games is introduced in Štrumbelj and Kononenko (2010) to study the influence that the characteristics of a particular individual have on their categorization. The objective of this chapter is to further extend this proposal by not only focusing on an individual-level, but by investigating the importance of the features on a sample of observations already classified. An axiomatic characterization of this measure is provided considering properties widely used in game theory. In addition, we validate the performance of our influence measure and present a real application of it to COVID-19 data. The given method can be useful to medical practitioners within the domain of health logistics. This chapter is mainly based on Davila-Pena et al. (2022b).

Part III. On cooperative Operations Research problems. Minimum cost spanning tree problems and sequencing problems are multi-agent OR problems that pursue two objectives. On the one hand, to minimize the costs related to the construction or design of the network, and on the other hand, to distribute these costs among the agents involved. A combination of ideas from these two problems gives rise to the graph machine scheduling problem (GMS-problem), where the goal sought is also two-fold. The third part of the manuscript focuses on this problem and it is composed of two chapters that present the model and study it from an optimization and allocation perspective.

Chapter 5. The graph machine scheduling problem (GMS-problem). This problem is motivated by a situation in the field of water supply. The solving of the general model turns out to be complex, so we will restrict ourselves to situations where the networks are trees. Throughout this chapter, we propose solution algorithms for these tree GMS-problems and theoretically prove their optimality. This chapter is mainly based on Davila-Pena et al. (2022a).

Chapter 6. An allocation rule for GMS-problems. After solving the GMS-problem, the question of how to distribute the optimal network design costs among the participating

agents arises. In this chapter we propose an allocation rule based on the algorithms and theoretical results discussed in Chapter 5. Furthermore, several examples are presented to illustrate how this rule is calculated. This chapter is mainly based on Davila-Pena et al. (2022a).

The manuscript includes some concluding remarks in Chapter 7. Finally, a summary of this dissertation in both Galician and Spanish languages is provided.

1.3 Methodology

This thesis follows the classical research methodology in the field of OR. In general, the research process begins by carrying out a comprehensive study of the topics to be addressed and by reviewing some classical and recent bibliographical references. The next stages include the proposal of new models and tools to tackle them, and their theoretical and/or computational analysis. Continuing with the structure outlined in the previous section, the specific methodology employed in each chapter is detailed hereafter.

Part I. On rich vehicle routing problems. As already mentioned, this part proposes a novel routing problem for which several algorithms are implemented. We describe below the methodology followed in each of the two chapters.

Chapter 2. Multi-compartment truck and trailer routing problem (MC-TTRP). We start by presenting a new route optimization problem that arises from a real case study. After situating this problem in the existing literature on OR logistics problems, we provide a mathematical formulation as a mixed-integer linear programming model. The `AMPL` software is used for the model implementation and the `Gurobi` solver is employed to obtain exact solutions on small instances taken from the aforementioned real-world application.

Chapter 3. Heuristics for the multi-compartment truck and trailer routing problem. Given the limitations to solve large problems, and according to the methodology adopted in previous works, four heuristic algorithms to obtain solutions for the problem are developed and thoroughly described. Computational studies are the most commonly chosen approach for assessing the performance of models and algorithms, due to the useful results they provide. The implementation of these methods is carried out in the `R` software and numerous simulations are conducted on the CESGA supercomputer. We design and solve specific test instances for the proposed problem. In addition, the algorithms are also applied to real instances of the case study mentioned above.

Part II. On game theory in classification problems. This part of the thesis combines game theory concepts with machine learning, specifically with classification problems. We detail below the methodology adopted in the chapter that composes it.

Chapter 4. An influence measure for classification problems. We begin by introducing the elements required for the subsequent proposal of the influence measure, both related to classification problems and to game theory. Afterward, desirable properties in the games domain are suggested for this measure and an axiomatic characterization is provided by means of these properties, following the classical scheme in this type of study. The influence measure is implemented in the `R` software and a number of simulations are designed and contextualized to validate its performance. Finally, the proposed method is applied to a COVID-19 database, serving also to illustrate its proper behavior.

Part III. On cooperative Operations Research problems. This part of the thesis begins by reviewing the optimization problems that have served as the basis of the new proposed problem, which will be examined from the optimization and cost allocation points of view. The methodology followed in each chapter is detailed below.

Chapter 5. The graph machine scheduling problem (GMS-problem). First, a new OR problem is motivated and formally presented. Solving algorithms for different subcases are proposed and their optimality is proved. The application of these algorithms is illustrated by examples and graphical elements are provided throughout the chapter.

Chapter 6. An allocation rule for GMS-problems. Following the methodology adopted by many OR practitioners, we present a cost allocation rule that is based on solving algorithms for the problem under consideration, in particular, those provided in the previous chapter. In addition, numerous illustrative examples of the calculation of our rule are given. All the graphs in this chapter have been produced with the `tikz` package.

In some of the works that compose this dissertation, the design of simulation studies to empirically assess the performance of new methods was necessary. For such aim, the use of `AMPL` and `R` softwares was essential, as already stated. In addition, the `yEd` diagram editor was used for the realization of most of the figures in Part I of the thesis. Furthermore, `R` software was employed to produce the graphical representations displayed in Part II, as well as to apply the implemented methods to diverse data sets throughout the manuscript.

Part I

On rich vehicle routing problems

Introduction to vehicle routing problems

Part I of this thesis is based on a rich variant of vehicle routing problems called the multi-compartment truck and trailer routing problem (MC-TTRP). It is structured in two closely related chapters that describe the model and solution methods, respectively.

The *vehicle routing problem* (VRP) attempts to determine the optimal set of routes that a fleet of vehicles should perform to serve a certain set of customers, and it is one of the most important and studied combinatorial optimization problems. It was introduced by Dantzig and Ramser (1959) and arises as a generalization of the traveling salesman problem (TSP, Dantzig et al., 1954). These authors described a real application regarding the supply of petrol from a terminal to several service stations, and provided both the first mathematical formulation of the problem and an algorithmic solution. A few years later, Clarke and Wright (1964) developed an effective voracious heuristic, which improved on the Dantzig–Ramser approach. With these seminal works as a starting point, numerous models and solving algorithms were proposed in the following years to obtain optimal or approximate solutions to routing problems.

The classic VRP concerns the delivery of goods to a number of geographically distributed customers whose demands are deterministic, known in advance, and non-divisible. There is a single depot, from where all vehicles, which are identical, must start and return to. The objective is to minimize the total distance required to serve all customers. Figure 1.1 shows a possible solution to the VRP.

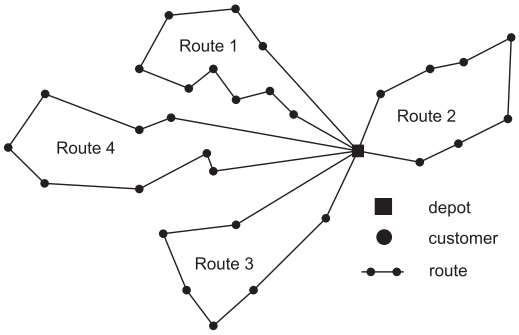


Figure 1.1: Possible solution to the VRP.

The VRP has multiple variants, which emerge in response to real situations. Some of these adaptations are:

- VRP with capacitated vehicles (CVRP), where each vehicle has a maximum capacity to carry.
- VRP with time windows (VRPTW), where each customer has an associated time interval in which it must be served.
- VRP with pickup and delivery (VRPPD), where vehicles must pick up and deliver a certain amount of goods to each of the customers.
- VRP with stochastic demands (VRPSD), where customers' demands are random variables.
- VRP with split-delivery (SDVRP), where the same customer can be served by several vehicles.
- VRP with multiple depots (MDVRP), where there are several depots from which customers can be supplied.
- VRP with multi-compartments (MC-VRP), where customers demand different types of products that cannot be mixed, so vehicles must be divided into compartments in order to transport them.

Figure 1.2 depicts a taxonomy of a representative set of VRP variants. Any combination of them is also a valid extension of the problem.

Regarding solution methods, both exact and approximate algorithms have been developed, highlighting in the latter group the heuristics, which are the ones we will focus on in this thesis. For a review of the optimal techniques implemented for the VRP, we recommend Baldacci et al. (2012) and Costa et al. (2019). As for heuristics, these are divided into classical heuristics and metaheuristics. The former allow us to obtain good quality solutions in a reasonable computation time, and the latter can be seen as a refinement of them, since a more detailed exploration of the search space is performed. Given that this problem has been extensively studied, we will not dwell on providing details of each of these methods. Instead, the reader is referred to Cordeau et al. (2005) for a further discussion of VRP heuristics. In addition, the books edited by Toth and Vigo (2002, 2014) are highly recommended for a general overview of the VRP and some of its variants, as well as solving algorithms and the applicability of this problem in the real world.

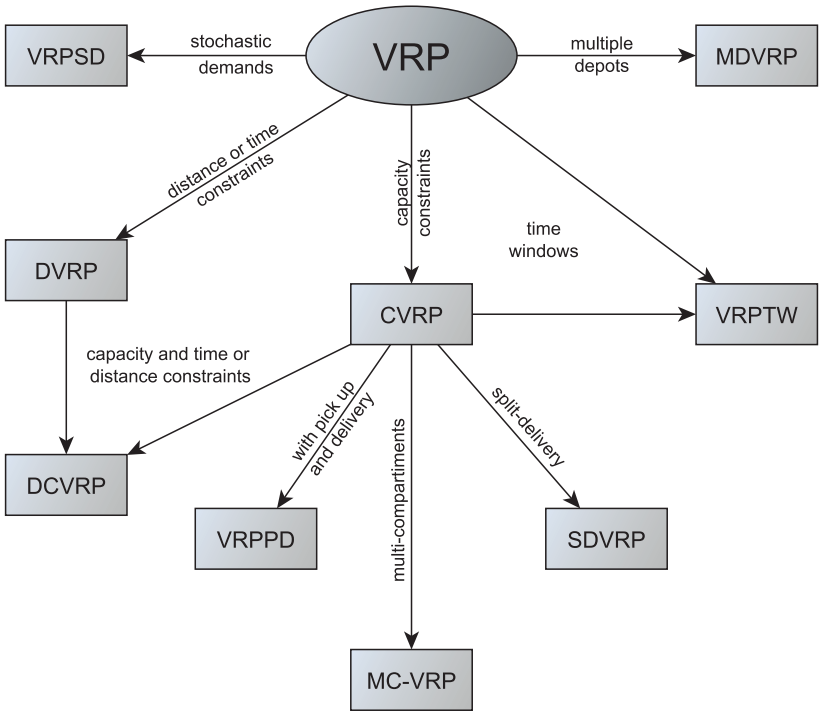


Figure 1.2: Chart with some variants of the VRP.

The problem that concerns us, the MC-TTRP, arises when combining, on the one hand, accessibility restrictions and, on the other hand, the need for compartmentalized vehicles. This variant emerges in situations where the service of large quantities of distinct goods is required, so that it is necessary to have large vehicles with compartments. However, what happens is that there are some customers who cannot be reached with vehicles of big dimensions. Thus, the fleet will consist of trucks and trailers, both compartmentalized, such that the latter can be uncoupled from the truck to serve those inaccessible customers.

Chapter 2 describes the MC-TTRP in detail and provides a mathematical formulation of it as a mixed-integer linear programming (MILP) problem. In addition, a case study from the field of logistics, which can be modeled using the MC-TTRP, is presented. An example is solved exactly, making evident the need to resort to approximate solving methods.

In Chapter 3, we introduce and implement several two-phase heuristic algorithms. A computational study is carried out to compare them and the one that provides the best results is analyzed thoroughly.

Chapter 2

Multi-compartment truck and trailer routing problem (MC-TTRP)

Vehicle routing problems admit different variants depending on the customers' needs. One of them is the truck and trailer routing problem (TTRP), where a fleet of trucks and trailers serves a set of customers such that when the trailer is not able to reach a customer, they are attended to only by the truck. The aim of this chapter is to propose a novel mixed-integer linear programming (MILP) approach to combine the TTRP with product compartmentalization, which we call the multi-compartment truck and trailer routing problem (MC-TTRP). The contents of this chapter are collected in Davila-Pena et al. (2023).

Contents

2.1 Introduction

In recent years, transport logistics has played a fundamental role in industry. Many public and private companies are interested in developing computational tools to design their routes, with objectives such as minimizing costs and/or maximizing the distribution of products. Thus, vehicle routing problems (VRPs) are a popular type of combinatorial optimization problem, through which transport routes for vehicles visiting a set of customers located at different places can be modeled. Solving these mathematical optimization problems is a challenge for operational researchers. When new features from real-world applications are considered,

such as capacitated vehicles, delivering in limited time windows, and stochastic behaviors, new variations of the original VRP arise, creating a need to develop new models and solution techniques.

One promising modification of the VRP is the truck and trailer routing problem (TTRP) proposed by Chao (2002), which incorporates accessibility restrictions. In this variant, a fleet of trucks and trailers visits a set of customers, where some customers (vehicle customers; VCs) can be served by a complete vehicle (i.e., a truck pulling a trailer), while others are only reachable by a truck alone (truck customers; TCs). Examples of TCs are customers in inner-city areas, mountainous regions, or places where maneuvering or access with a trailer is not possible. To solve this problem, we distinguish three types of routes: pure truck routes (PTRs), which can be traveled only by trucks, pure vehicle routes (PVRs), which can be traveled entirely by a complete vehicle, and mixed vehicle routes (MVRs), which consist of a main tour traveled by a complete vehicle and one or more sub-tours traveled only by the truck part of the vehicle. Figure 2.1 illustrates a possible solution to the TTRP. Although this model can be very useful in many land-based logistical applications, the presence of three different types of routes makes solving the associated optimization problem more difficult, suiting it to the application of heuristics and metaheuristics, such as in Lin et al. (2009). Real-world applications include farm milk collection (Caramia and Guerriero, 2010b), delivery by feed mills (Lin et al., 2009), and the provisioning of infrastructure services in urban areas with accessibility restrictions (Parragh and Cordeau, 2017).

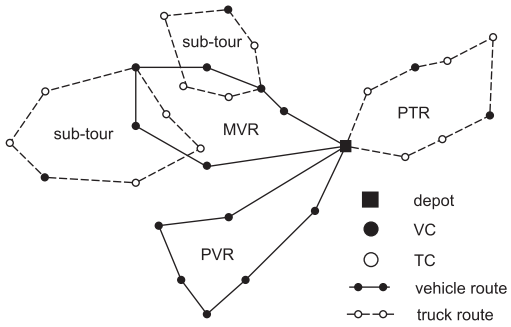


Figure 2.1: Possible solution to the TTRP.

Another interesting variation of the VRP is the multi-compartment case, where different products must be split into different storages during transport, making it challenging to maximize the use of vehicle capacity on the generated routes. Although the inclusion of compartments adds extra complexity, it can be a requirement in real logistical applications, as explained by Guitián de Frutos and Casas-Méndez (2019). Therefore, this chapter focuses

on a routing problem that combines the TTRP with product compartmentalization, which we call the multi-compartment truck and trailer routing problem (MC-TTRP). In particular, the main interest here lies in the proposal of a novel mixed-integer linear programming (MILP) approach that covers the whole problem. The combination of these two features is motivated by the needs of a Spanish agricultural cooperative that produces feed for cattle, which we used to test, illustrate, and apply the proposed formulation. A tentative MC-TTRP was originally introduced in a preliminary work by Davila-Pena (2019).

The remainder of this chapter is organized as follows. Section 2.2 reviews related work. Section 2.3 presents the current case study in detail and an in-depth description and formulation of the MC-TTRP. Finally, Section 2.4 reports a thorough study of the exact solution of the proposed MC-TTRP model using the case study previously outlined.

2.2 Related work

To solve the logistics of an agricultural cooperative that distributes feed for cattle to a large number of customers, many of whom have accessibility restrictions, the TTRP appears to be a satisfactory model. Although Chao (2002) introduced the term TTRP, previous works have incorporated trailers to solve similar case studies. The first approach could be that presented by Semet and Taillard (1993). These authors proposed a VRP that considered the use of trailers under accessibility restrictions. Semet (1995) proposed another example describing a new variant of the VRP formulated as an integer linear programming (ILP) problem called the partial accessibility constrained VRP (PACVRP). Despite being very similar to Chao's TTRP, it has specific differences, such as the utilization of all available trucks. Other studies have considered a heterogeneous fleet of vehicles composed of trucks and trailers, such as the case of Gerdessen (1996), whose model is known as the VRP with trailers (VRPT). Moreover, Chao et al. (1998) studied the site-dependent VRP (SD-VRP), where every customer has a specific type of vehicle assigned. Some seminal papers on the TTRP do not offer a mathematical formulation through an MILP model, although Scheuerer (2004) presented a formulation of the TTRP by Chao (2002). This turns out to be an adaptation of the proposal by Semet (1995) for the PACVRP and can be considered as the motivation for the current chapter.

Other researchers have built new models based on the proposal of Chao (2002) to meet various real-world requirements, such as a TTRP with time windows (TTRPTW) proposed by Lin et al. (2011). In the TTRPTW, besides its type and demand, each customer has three associated measurable times: the earliest and latest time of day at which it can be served and

the service time required. Recently, Accorsi and Vigo (2020) considered a generalization of the TTRP, the extended single TTRP (XSTTRP), which contains, all together, a variety of node types that were previously considered only separately: truck customers, vehicle customers with and without parking places, and parking-only locations. In the XSTTRP, a single vehicle, consisting of a truck and a detachable trailer, is used to serve a set of customers with known demand and accessibility constraints.

As already mentioned, another challenging variation of the VRP arises when customers demand various types of products that cannot be mixed. This is the case in the multi-compartment VRP (MC-VRP), which was initially presented in Brown and Graves (1981) and Brown et al. (1987), whose objective was the distribution of petroleum products in the United States. Derigs et al. (2011) formulated the MC-VRP as an ILP problem. As with the classic VRP, the MC-VRP can accommodate several adaptations. For instance, Mendoza et al. (2010) extended the MC-VRP to the case in which the demands are stochastic, giving rise to the MC-VRP with stochastic demands (MC-VRPSD). Asawarungsaengkul et al. (2013) proposed various mathematical formulations of the MC-VRP with split pattern (MC-VRP-SP). More recently, Hachicha et al. (2019) presented an MILP formulation for the MC-VRP with time windows (MC-VRPTW).

Let us recall that, motivated by a real example, the main objective of this chapter is to mathematically formulate a model that incorporates trucks and trailers divided into compartments, i.e., that combines the characteristics of both TTRP and MC-VRP. The work of Caramia and Guerriero (2010b) should be highlighted as the first (and to the best of our knowledge, the only) to consider the TTRP with compartments, which we refer to as the MC-TTRP hereinafter. They investigated a VRP in which at most one type of product could be assigned to each compartment. Furthermore, they established the additional constraint that some delivery locations were small and inaccessible by large vehicles. Due to the similarity between the problem addressed in that paper and this chapter, it is considered convenient to point out the differences between the two studies in terms of motivation and modeling. First, with regard to actual motivation, the problem analyzed by Caramia and Guerriero (2010b) was for milk collection on farms by an Italian company, while in our real-world case study, which will be described in more detail in the next section, the problem of the distribution of feed among members of a Spanish agricultural cooperative was analyzed. Regarding the model and methodology used, Caramia and Guerriero (2010b) proposed two mathematical programming models. One of them aimed to assign vehicles to farmers with the objective of minimizing the number of vehicles used, satisfying restrictions on capacity, demand, and types of milk. It should be noted that the group of farmers was divided into four zones, and

an initial allocation of vehicles was made to each zone. The fleet considered was heterogeneous. The second model was used to minimize the lengths of the routes. In the proposed methodology, the possibility of serving VCs on sub-tours was not permitted. Following a different approach, in our setup, there is a homogeneous truck fleet and a trailer fleet, and vehicle pre-assignments are not made to groups of customers. In this case, the sub-tours on an MVR, traveled by only a truck, can visit both TCs and VCs. In addition, what makes an important difference is that the formulation of a single model covering the whole problem is provided.

2.3 Problem description and formulation

2.3.1 Case study

The motivation for this study stemmed from the needs of a Spanish cooperative that produces and distributes feed for farm animals. The company is located in Galicia, a region in the northwestern Spain with an area of 29,565 km^2 spread over four provinces and 315 municipalities. The cooperative, which was created 16 years ago, currently has a total clientele of more than 1500 farmers distributed throughout the four provinces of Galicia (although not all of them order from the feed factory) and covering 60 municipalities across a large geographical area. The annual amount of feed produced exceeds 150,000 tons.

The agricultural company produces different types of feed, and farmers usually place one or two orders per month. The number of daily orders is approximately 40, where each order ranges from 500 to more than 14,000 kg. The average number of annual orders per feed customer is approximately 17. There are also occurrences such as the loss or incorporation of new customers. The roads leading to some of the farms or the farms themselves are inaccessible by large trailers. Moreover, customers sometimes request different types of feed because they have different species of animals. Naturally, goods that are not of the same type cannot be mixed. Thus, it is necessary to have compartmentalized vehicles. In addition to not being able to mix different kinds of feed in the same hopper, the same compartment cannot be used to supply two different customers because the cooperative does not have technology to measure out each customer's supply from their vehicles.

The purpose of this study was to provide a tool for the cooperative to automatically design routes for each vehicle such that their restrictions are met and the distance traveled is minimized. Each day, new orders may be received, trucks may experience breakdowns, and customers may change their demands at short notice. All these factors suggest that route planning is only useful within two or three days at most.

A team of agricultural engineers designed a comprehensive global positioning system (GPS) that can monitor various vehicle routes. The GPS supplies all the geographic information required to provide the data to solve the problem. We also know the capacity of each compartment, the demands of different customers, and whether a trailer can access each farm as well as its load restrictions.

2.3.2 Mixed-integer linear programming model

As stated before, this chapter proposes an MC-TTRP model–a novel MILP implementation of the TTRP with multi-compartmentalized vehicles.

The MC-TTRP can be described as follows. Let $G = (N, E)$ be an undirected graph consisting of a node set $N = \{0, 1, \ldots, n\}$, representing the depot (0) and customers $(1, \ldots, n)$, and a set $E = \{\{i, j\} \mid i, j \in N, i \neq j\}$, representing the edges that can be traveled between different nodes. N_1 and N_2 are subsets of N that contain the n_1 and $n - n_1$ VCs and TCs, respectively. A non-negative cost c_{ij}, $\{i, j\} \in E$, is assigned to each edge, which represents the distance a vehicle must travel from i to j. Each node $i \in N$ requires a service time s_i, which, in the case of the depot, refers to the time required to load the vehicles. For transportation, a set $K^T = \{1, \ldots, m_L, \ldots, m_T\}$ of trucks and set $K^L = \{1, \ldots, m_L\}$ of trailers are available. $K_1^T = \{1, \ldots, m_L\}$ and $K_2^T = \{m_L + 1, \ldots, m_T\}$ are the subsets of K^T that consist of trucks that can pull a trailer and pure trucks (without trailer attached), respectively. Note that $|K_1^T| = |K^L|$, i.e., there are m_L complete vehicles. In addition, m_L is the number of trailers, and m_T is the number of trucks ($m_L \leqslant m_T$). Complete vehicles and pure trucks are assumed to be homogeneous. Let Q_T be the capacity of each truck and Q_L be the capacity of each trailer. Hence, $Q_T + Q_L$ is the capacity of a complete vehicle.

As mentioned above and illustrated in Figure 2.1, three different types of routes can appear in this variant of the TTRP. For MVRs, the complete vehicle leaves the depot and serves some VCs; this part of the MVR is known as the main tour. The main tour is entirely covered by a complete vehicle and starts and ends at the depot. During the tour of an MVR, it is possible to uncouple the trailer from the truck and leave it parked at one of the VC locations to start a sub-tour (or even at the depot, which is always a candidate for trailer parking places). VCs and TCs can be served in a sub-tour because they are performed by a pure truck. Sub-tours begin and end at the parking place (the depot or any of the VCs of the main tour), also known as the root of the sub-tour. There are no restrictions on the number of sub-tours in an MVR or on the number of sub-tours that can start from the same VC on a given main tour, as long as the vehicle capacity restrictions are satisfied. That is, the demands transported on the MVR cannot exceed the capacity of a complete vehicle, $Q_T + Q_L$, and Q_T cannot be surpassed in a

sub-tour. Another type of route is a PVR, which is fully traveled by a complete vehicle, implying that only VCs can be delivered to and their demands cannot exceed $Q_T + Q_L$. On the contrary, PTRs serve both types of customers because trucks travel without a trailer attached. The demand transported on a PTR cannot exceed Q_T.

For the sake of simplicity, it is also assumed that travel costs are the same for all vehicles, regardless of whether a trailer is attached. Each trailer $r \in K^L$ is divided into a set of compartments, H^L, where Q_L^H is the capacity of a trailer compartment. Similarly, each truck $k \in K^T$ is split into a set of compartments, H^T, where Q_T^H is the capacity of a truck compartment. Furthermore, let the set $F = \{1, \ldots, n_F\}$ of feed types be given. Each node (except for the depot) has a non-negative demand d_{if} ($i \in N \backslash \{0\}$, $f \in F$) for every feed type. The demands must be served at customers' locations and transported from the depot without the feed types being mixed. In addition, products for different customers cannot be carried within the same compartment. The total demands of each customer must be met by the same vehicle, and it is possible to divide a customer's demand for the same feed type among several compartments. Trucks and complete vehicles have a maximum usage time allowed of D, an average speed of vm, and legal capacities, L_T and $L_T + L_L$, respectively, which may appear depending on regulations or laws in some specific areas.

Regarding the decision variables involved in the model, x_{ij}^{kr} and y_{ij}^{klv} (both binary) are related to the construction of routes. The former involves routes covered by complete vehicles (MVRs or PVRs), and it takes a value of 1 if the complete vehicle consisting of truck $k \in K_1^T$ and trailer $r \in K^L$ travels from node i to j ($i, j \in N_1 \cup \{0\}$); otherwise, it is 0. In contrast, y_{ij}^{klv} refers to routes covered only by trucks (PTRs or sub-tours of MVRs), and it takes a value of 1 if truck $k \in K^T$ traverses the edge $\{i, j\}$ on the v-th route/sub-tour ($v \in \mathcal{V} = \{1, \ldots, n\}$)[1] with root $l \in N_1 \cup \{0\}$; otherwise, it is 0. For $l = 0$, the associated tour is a PTR ($k \in K_2^T$). However, if $l \in N_1$, then such a root refers to the VC of the main tour of an MVR working as a trailer parking place to start a sub-tour (and $k \in K_1^T$). The remaining variables are related to the vehicle compartments. In particular, $ZT_{i,f,ht}^k$ takes values in $[0, 1]$ and represents the proportion of compartment $ht \in H^T$ of truck $k \in K^T$ carrying feed $f \in F$ for customer $i \in N \backslash \{0\}$, whereas $U_{i,f,ht}^k$ is a binary variable equal to 1 if $ZT_{i,f,ht}^k > 0$ and 0 otherwise. Analogously, $ZL_{i,f,hl}^r$ takes values in $[0, 1]$ representing the proportion of compartment $hl \in H^L$ of trailer $r \in K^L$ loaded with feed $f \in F$ for customer $i \in N_1$, while $V_{i,f,hl}^r$ is a binary variable equal to 1 if $ZL_{i,f,hl}^r > 0$ and 0 otherwise.

The objective of the MC-TTRP is to determine a set of vehicle tours that minimizes the total cost of all edges to be traveled, i.e., the total distance of the solution routes, such that

[1] Note that a root candidate $l \in N_1 \cup \{0\}$ can have as many sub-tours as there are customers, n.

Table 2.1: Notation of the proposed MC-TTRP.

Set or parameter	Definition	Parameter	Definition
$\{0\}$	Depot	n	Total number of customers
N_1	Set of VCs	n_1	Number of VCs
N_2	Set of TCs	m_L	Number of trailers (and of complete vehicles)
$N = \{0\} \cup N_1 \cup N_2$	Set of nodes (customers and depot)	m_T	Number of trucks
$N^* = N \backslash \{0\}$	Set of customers	c_{ij}	Distance a vehicle must travel from i to j
$N_1^0 = N_1 \cup \{0\}$	Set of VCs and depot	n_F	Number of different types of feed
K_1^T	Set of trucks that can hitch a trailer	d_{if}	Demand of customer i for feed f
K_2^T	Set of pure trucks	Q_T	Capacity of a truck
$K^T = K_1^T \cup K_2^T$	Set of trucks	Q_L	Capacity of a trailer
K^L	Set of trailers	Q_T^H	Capacity of a truck's compartment
F	Set of different types of feed	Q_L^H	Capacity of a trailer's compartment
H^T	Set of truck compartments	D	Maximum time allowed to use a truck/vehicle
H^L	Set of trailer compartments	vm	Average speed of trucks/vehicles
$\mathcal{V}$	Set of tours/sub-tours leaving a specific root	s_i	Service time required for node $i \in N$
L_T	Legal capacity of trucks	L_L	Legal capacity of trailers

Variable	Definition
x_{ij}^{kr}	Binary variable equal to 1 if truck k with trailer r passes through edge $\{i, j\}$; 0 otherwise
y_{ij}^{klv}	Binary variable equal to 1 if truck k passes through edge $\{i, j\}$ on route/sub-tour v with parking place l; 0 otherwise
$U_{i,f,ht}^{k}$	Binary variable equal to 1 if compartment ht of truck k is loaded with feed f for customer i; 0 otherwise
$V_{i,f,hl}^{r}$	Binary variable equal to 1 if compartment hl of trailer r is loaded with feed f for customer i; 0 otherwise
$ZT_{i,f,ht}^{k}$	Proportion of compartment ht of truck k loaded with feed f for customer i
$ZL_{i,f,hl}^{r}$	Proportion of compartment hl of trailer r loaded with feed f for customer i

all constraints are met, i.e., the demands are satisfied, no vehicle capacities are exceeded, and the restrictions of access to customers and constraints related to the loading of compartments are considered.

From now on, we will denote the set of VCs and the depot, $N_1 \cup \{0\}$, as N_1^0, and we will denote the set of customers, $N\backslash\{0\}$, by N^*. Table 2.1 gives a summary of the sets, parameters, and decision variables involved in the model to facilitate better understanding of our proposal. Given this terminology and notation, the objective function and constraints of the MC-TTRP can be formulated as follows:

$$\text{minimize} \sum_{i\in N_1^0}\sum_{j\in N_1^0}\sum_{k\in K_1^T}\sum_{r\in K^L} c_{ij}x_{ij}^{kr} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in K^T}\sum_{l\in N_1^0}\sum_{v\in\mathcal{V}} c_{ij}y_{ij}^{klv} \tag{2.1}$$

subject to

$$\sum_{i\in N_1^0}\sum_{k\in K_1^T}\sum_{r\in K^L} x_{ij}^{kr} + \sum_{i\in N}\sum_{k\in K^T}\sum_{\substack{l\in N_1^0\\ l\neq j}}\sum_{v\in\mathcal{V}} y_{ij}^{klv} = 1, \qquad j \in N_1; \tag{2.2}$$

$$\sum_{i\in N_1^0}\sum_{k\in K_1^T}\sum_{r\in K^L} x_{ij}^{kr} + \sum_{i\in N}\sum_{k\in K^T} y_{ij}^{kjv} \leqslant 2, \qquad j \in N_1;\ v \in \mathcal{V}; \tag{2.3}$$

$$\sum_{i\in N}\sum_{k\in K^T}\sum_{l\in N_1^0}\sum_{v\in\mathcal{V}} y_{ij}^{klv} = 1, \qquad j \in N_2; \tag{2.4}$$

$$\sum_{i\in N}\sum_{k\in K^T}\sum_{l\in N_1}\sum_{v\in\mathcal{V}} y_{i0}^{klv} = 0, \tag{2.5}$$

$$\sum_{j\in N} y_{lj}^{klv} \leqslant 1, \qquad k \in K^T;\ l \in N_1^0;\ v \in \mathcal{V}; \tag{2.6}$$

$$y_{lj}^{klv} \leqslant \sum_{i\in N_1^0} \sum_{r\in K^L} x_{il}^{kr}, \qquad j \in N^*;\ k \in K_1^T;\ l \in N_1;\ v \in \mathcal{V}; \tag{2.7}$$

$$y_{ij}^{klv} \leqslant \sum_{p\in N} y_{lp}^{klv}, \qquad i,j \in N;\ k \in K^T;\ l \in N_1^0;\ v \in \mathcal{V}; \tag{2.8}$$

$$\sum_{i\in N} \sum_{j\in N} \sum_{v\in\mathcal{V}} y_{ij}^{klv} = 0, \qquad k \in K_2^T;\ l \in N_1; \tag{2.9}$$

$$\sum_{i\in N} \sum_{j\in N} \sum_{v\in\mathcal{V}} y_{ij}^{k0v} = 0, \qquad k \in K_1^T; \tag{2.10}$$

$$\sum_{v\in\mathcal{V}} \sum_{j\in N*} y_{0j}^{k0v} \leqslant 1, \qquad k \in K_2^T; \tag{2.11}$$

$$\sum_{j\in N_1} \sum_{r\in K^L} x_{0j}^{kr} \leqslant 1, \qquad k \in K_1^T; \tag{2.12}$$

$$\sum_{j\in N_1} \sum_{k\in K_1^T} x_{0j}^{kr} \leqslant 1, \qquad r \in K^L; \tag{2.13}$$

$$\sum_{i\in N*} \sum_{\substack{j\in N* \\ j\neq l}} \sum_{f\in F} d_{jf} y_{ij}^{klv} \leqslant Q_T, \qquad k \in K_1^T;\ l \in N_1;\ v \in \mathcal{V}; \tag{2.14}$$

$$\sum_{i\in N} \sum_{j\in N*} \sum_{f\in F} d_{jf} y_{ij}^{k0v} \leqslant Q_T, \qquad k \in K_2^T;\ v \in \mathcal{V}; \tag{2.15}$$

$$\sum_{i\in N_1^0} \sum_{j\in N_1} \sum_{f\in F} d_{jf} x_{ij}^{kr} + \sum_{i\in N*} \sum_{\substack{j\in N* \\ j\neq l}} \sum_{l\in N_1} \sum_{v\in\mathcal{V}} \sum_{f\in F} d_{jf} y_{ij}^{klv} \leqslant Q_T + Q_L, \quad k \in K_1^T;\ r \in K^L; \tag{2.16}$$

$$\begin{aligned} &\sum_{i\in N_1} \sum_{j\in N_1^0} s_j x_{ij}^{kr} + \sum_{l\in N_1^0} \sum_{v\in\mathcal{V}} \sum_{\substack{i\in N* \\ i\neq l}} \sum_{j\in N} s_j y_{ij}^{klv} + \sum_{i\in N_1^0} \sum_{j\in N_1^0} (c_{ij}/vm) x_{ij}^{kr} \\ &\quad + \sum_{l\in N_1^0} \sum_{v\in\mathcal{V}} \sum_{i\in N} \sum_{j\in N} (c_{ij}/vm) y_{ij}^{klv} \leqslant D - s_0, \qquad k \in K_1^T;\ r \in K^L; \end{aligned} \tag{2.17}$$

$$\sum_{v\in\mathcal{V}} \sum_{i\in N} \sum_{j\in N} (s_j + c_{ij}/vm) y_{ij}^{k0v} \leqslant D, \qquad k \in K_2^T; \tag{2.18}$$

$$\sum_{i\in N} y_{ij}^{klv} = \sum_{p\in N} y_{jp}^{klv}, \qquad j \in N;\ k \in K^T;\ l \in N_1^0;\ v \in \mathcal{V}; \tag{2.19}$$

$$\sum_{i\in N_1^0} x_{ij}^{kr} = \sum_{p\in N_1^0} x_{jp}^{kr}, \qquad j \in N_1^0;\ k \in K_1^T;\ r \in K^L; \tag{2.20}$$

$$\sum_{i\in B} \sum_{j\in B} x_{ij}^{kr} \leqslant |B| - 1, \qquad k \in K_1^T;\ r \in K^L;\ \forall B \subseteq N_1 \,:\, |B| \geqslant 2; \tag{2.21}$$

$$\sum_{i\in B} \sum_{j\in B} y_{ij}^{klv} - \sum_{i\in B\cap N_1} \sum_{j\in N_1\setminus B} \sum_{r\in K^L} x_{ij}^{kr} \leqslant |B| - 1, \qquad \begin{aligned} &k \in K_1^T;\ l \in N_1; \\ &v \in \mathcal{V};\ \forall B \subseteq N \,:\, |B| \geqslant 2; \end{aligned} \tag{2.22}$$

$$\sum_{i\in B}\sum_{j\in B} y_{ij}^{k0v} \leqslant |B|-1, \qquad k\in K_2^T;\ v\in\mathcal{V};\ \forall B\subseteq N\ :\ |B|\geqslant 2; \quad (2.23)$$

$$\frac{1}{|H^T|}\sum_{f\in F}\sum_{ht\in H^T} ZT_{j,f,ht}^{k} \leqslant \sum_{i\in N_1^0}\sum_{r\in K^L} x_{ij}^{kr} + \sum_{i\in N}\sum_{l\in N_1}\sum_{v\in\mathcal{V}} y_{ij}^{klv}, \qquad k\in K_1^T;\ j\in N_1; \quad (2.24)$$

$$\frac{1}{|H^T|}\sum_{f\in F}\sum_{ht\in H^T} ZT_{j,f,ht}^{k} \leqslant \sum_{i\in N}\sum_{v\in\mathcal{V}} y_{ij}^{k0v}, \qquad k\in K_2^T;\ j\in N_1; \quad (2.25)$$

$$\frac{1}{|H^L|}\sum_{f\in F}\sum_{hl\in H^L} ZL_{j,f,hl}^{r} \leqslant \sum_{i\in N_1^0}\sum_{k\in K_1^T} x_{ij}^{kr}, \qquad r\in K^L;\ j\in N_1; \quad (2.26)$$

$$\frac{1}{|H^T|}\sum_{f\in F}\sum_{ht\in H^T} ZT_{j,f,ht}^{k} \leqslant \sum_{i\in N}\sum_{l\in N_1^0}\sum_{v\in\mathcal{V}} y_{ij}^{klv}, \qquad k\in K^T;\ j\in N_2; \quad (2.27)$$

$$\sum_{i\in N_1^0} x_{ij}^{kr} \leqslant \sum_{f\in F}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k} + \sum_{f\in F}\sum_{hl\in H^L} Q_L^H ZL_{j,f,hl}^{r}, \qquad \begin{array}{r} k\in K_1^T;\ r\in K^L; \\ j\in N_1; \end{array} \quad (2.28)$$

$$\sum_{i\in N}\sum_{v\in\mathcal{V}} y_{ij}^{k0v} \leqslant \sum_{f\in F}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k}, \qquad k\in K_2^T;\ j\in N_1; \quad (2.29)$$

$$\sum_{i\in N}\sum_{l\in N_1}\sum_{v\in\mathcal{V}} y_{ij}^{klv} \leqslant \sum_{f\in F}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k}, \qquad k\in K_1^T;\ j\in N_1; \quad (2.30)$$

$$\sum_{i\in N}\sum_{l\in N_1^0}\sum_{v\in\mathcal{V}} y_{ij}^{klv} \leqslant \sum_{f\in F}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k}, \qquad k\in K^T;\ j\in N_2; \quad (2.31)$$

$$\sum_{r\in K^L}\sum_{hl\in H^L} Q_L^H ZL_{j,f,hl}^{r} + \sum_{k\in K^T}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k} = d_{jf}, \qquad j\in N_1;\ f\in F; \quad (2.32)$$

$$\sum_{k\in K^T}\sum_{ht\in H^T} Q_T^H ZT_{j,f,ht}^{k} = d_{jf}, \qquad j\in N_2;\ f\in F; \quad (2.33)$$

$$\sum_{i\in N^*}\sum_{f\in F}\sum_{ht\in H^T} Q_T^H ZT_{i,f,ht}^{k} \leqslant L_T, \qquad k\in K^T; \quad (2.34)$$

$$\sum_{i\in N_1}\sum_{f\in F}\sum_{hl\in H^L} Q_L^H ZL_{i,f,hl}^{r} \leqslant L_L, \qquad r\in K^L; \quad (2.35)$$

$$\sum_{i\in N^*}\sum_{f\in F} ZT_{i,f,ht}^{k} \leqslant 1, \qquad k\in K^T;\ ht\in H^T; \quad (2.36)$$

$$\sum_{i\in N_1}\sum_{f\in F} ZL_{i,f,hl}^{r} \leqslant 1, \qquad r\in K^L;\ hl\in H^L; \quad (2.37)$$

$$ZT_{i,f,ht}^{k} - U_{i,f,ht}^{k} \leqslant 0, \qquad i\in N^*;\ f\in F;\ k\in K^T;\ ht\in H^T; \quad (2.38)$$

$$ZL_{i,f,hl}^{r} - V_{i,f,hl}^{r} \leqslant 0, \qquad i\in N_1;\ f\in F;\ r\in K^L;\ hl\in H^L; \quad (2.39)$$

$$U_{i,f_1,ht}^{k} + U_{j,f_2,ht}^{k} \leqslant 1, \qquad i,j\in N^*;\ f_1,f_2\in F;\ f_1\neq f_2;\ k\in K^T;\ ht\in H^T; \quad (2.40)$$

$$V_{i,f_1,hl}^{r} + V_{j,f_2,hl}^{r} \leqslant 1, \qquad i,j\in N_1;\ f_1,f_2\in F;\ f_1\neq f_2;\ r\in K^L;\ hl\in H^L; \quad (2.41)$$

$$U_{i,f,ht}^{k} + U_{j,f,ht}^{k} \leqslant 1, \qquad i,j\in N^*;\ i\neq j;\ f\in F;\ k\in K^T;\ ht\in H^T; \quad (2.42)$$

$$V_{i,f,hl}^{r} + V_{j,f,hl}^{r} \leqslant 1, \qquad i,j\in N_1;\ i\neq j;\ f\in F;\ r\in K^L;\ hl\in H^L; \quad (2.43)$$

$$x_{ij}^{kr} \in \{0,1\}, \qquad i,j \in N_1^0;\ k \in K_1^T;\ r \in K^L; \quad (2.44)$$

$$y_{ij}^{klv} \in \{0,1\}, \qquad i,j \in N;\ k \in K^T;\ l \in N_1^0;\ v \in \mathcal{V}; \quad (2.45)$$

$$ZT_{i,f,ht}^{k} \in [0,1], \qquad i \in N;\ f \in F;\ k \in K^T;\ ht \in H^T; \quad (2.46)$$

$$ZL_{i,f,hl}^{r} \in [0,1], \qquad i \in N_1^0;\ f \in F;\ r \in K^L;\ hl \in H^L; \quad (2.47)$$

$$U_{i,f,ht}^{k} \in \{0,1\}, \qquad i \in N;\ f \in F;\ k \in K^T;\ ht \in H^T; \quad (2.48)$$

$$V_{i,f,hl}^{r} \in \{0,1\}, \qquad i \in N_1^0;\ f \in F;\ r \in K^L;\ hl \in H^L. \quad (2.49)$$

The objective function and above restrictions are explained in the following, separating them into thematic blocks to fully understand the model:

- **Objective function** (2.1): This minimizes the total cost of all tours. The first term refers to the routes traveled by a complete vehicle (PVRs and main tours of MVRs), while the second term includes the PTRs and MVR sub-tours.

- **Customer-specific restrictions** (2.2)–(2.8): (2.2) establishes that each VC must be present exactly once either in the main route of a complete route or in a sub-tour of which it is not the parking place. From (2.3), a VC can be present twice if it is the root of a sub-tour. (2.4) indicates that TCs must be visited only once, either on a sub-tour or on a PTR (when $l = 0$). From (2.5), the depot cannot be present in any sub-tour that begins at a VC. Constraint (2.6) shows that for each sub-tour or PTR, the corresponding truck goes from the parking place or depot, as appropriate, to a customer no more than once. (2.7) implies that if a VC is not served by a complete vehicle, then that customer cannot be a parking place candidate to start a sub-tour. (2.8) describes that other customers can only belong to a sub-tour starting from a candidate parking place if that candidate is actually selected as a parking place.

- **Vehicle-specific restrictions** (2.9)–(2.18): From (2.9), trucks that do not have an associated trailer do not have a main route. (2.10) indicates that trucks with a trailer attached cannot perform PTRs. From (2.11), for trucks without trailers, there can be at most one route, i.e., a PTR (there is no multiple use of vehicles). From constraints (2.12)–(2.13), for complete vehicles, the number of sub-tours is not limited, but the number of main routes cannot exceed one. (2.14)–(2.15) describe the customer demands, which cannot exceed the capacity of a truck, Q_T, on a route/sub-tour without a trailer. (2.16) explains the demand limits on a route performed by a truck pulling a trailer, which cannot exceed $Q_T + Q_L$. From (2.17), complete vehicles cannot exceed their maximum usage time, including the loading or unloading time of a vehicle in the

depot, s_0. (2.18) specifies the maximum usage time in the case of trucks, which also cannot be surpassed.

- **Formulation-specific restrictions** (2.19)–(2.23): (2.19) covers flow conservation in sub-tours and PTRs. (2.20) represents the flow conservation on PVRs and main tours. (2.21) provides disconnected cycle elimination constraints on the main tours of MVRs and PVRs. (2.22) outlines the suppression of disconnected cycles in sub-tours. (2.23) models the removal of disconnected cycles on PTRs.

- **Relations between routes and load distribution in different compartments** (2.24)–(2.31): (2.24) specifies that if a truck in an MVR or PVR does not visit a VC, then that truck does not load goods for that customer in any of its compartments. From (2.25), if a truck in a PTR does not visit a VC, then that truck does not load products for that customer in any of its compartments. From (2.26), if a trailer does not visit a VC then that trailer does not load goods for that customer in any of its compartments. From (2.27), if a truck does not visit a TC, then it does not load goods for that customer. (2.28) states that if a complete vehicle visits a VC, then either the truck or the trailer is loaded with goods for that customer. From (2.29), if a VC is visited by a truck (in a PTR), then the truck distributes goods for that customer. Constraint (2.30) states that if a truck in a sub-tour serves a VC, then that truck is loaded with goods for that customer. From (2.31), if a truck visits a TC, then it transports goods to that customer in some of its compartments.

- **Demand delivery restrictions** (2.32)–(2.33): From (2.32), every VC receives all of its demand. From constraint (2.33), every TC is delivered all of its demand.

- **Volume of goods that can be transported** (2.34)–(2.37): From (2.34), no truck loads more than what is legally allowed. Constraint (2.35) states that no trailer loads more than what is legally allowed. (2.36) states that no truck compartment loads above its capacity. From (2.37), no trailer compartment loads above its capacity.

- **Technical restrictions in the loading procedure** (2.38)–(2.43): (2.38) defines the logical relationship between U and ZT. (2.39) defines the logical relationship between V and ZL. From (2.40), it is not possible to mix products of different types (neither from the same customer nor from different customers) in the same compartment of a truck. Constraint (2.41) states that products of different types cannot be mixed (neither from the same customer nor from different customers) in the same compartment of a trailer.

(2.42) prohibits mixing products from different customers in the same truck compartment. (2.43) disallows the mixing of products from different customers in the same compartment of a trailer.

- **Nature of the variables involved in the model** (2.44)–(2.49): From (2.44) and (2.45), x and y variables are binary, respectively. (2.46) and (2.47) specify that ZT and ZL variables take values in $[0, 1]$, respectively. From (2.48) and (2.49), U and V variables are binary, respectively.

Our proposed MC-TTRP offers a comprehensive way to model compartments using the TTRP approach. It was formulated as an MILP problem, considering the complexity associated with this type of problem. The following section analyzes how to solve it exactly, alongside the advantages and disadvantages of this method.

2.4 Exact solving of a real example

As we have real data for this case study, an optimization scenario was built to assess our MC-TTRP model. In particular, we know the distances between the different nodes (customers and central depot) as well as customer demands and vehicle capacities. Moreover, the drivers work 8 h per day, and we estimated the average vehicle speed to be 60 km/h. Furthermore, because we do not have information about the service time of each customer or the time employees need to load the trucks, we assumed it to be negligible. In addition, in our real instance of the model, 10 customers were selected, five of each type, provided by the company's vehicles: three trucks and two trailers.

The trucks have 13 hoppers each, which can carry up to 1.5 tons of loads, while the trailers have 15 hoppers each with a maximum capacity of 2 tons. The demand d of the customers (in kilograms), according to the four types of feed distributed by the cooperative, and the matrix of distances, C, (in kilometers) between the nodes are as follows:

$$d = \begin{pmatrix} 1000 & 0 & 0 & 2300 \\ 4000 & 0 & 0 & 2041 \\ 1959 & 0 & 4000 & 0 \\ 0 & 951 & 2000 & 0 \\ 0 & 3500 & 1385 & 0 \\ 0 & 3003 & 0 & 0 \\ 516 & 0 & 0 & 2500 \\ 978 & 0 & 3500 & 0 \\ 2000 & 0 & 2513 & 900 \\ 0 & 3490 & 0 & 0 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 0 & 21 & 20 & 17 & 65 & 63 & 60 & 19 & 22 & 24 & 60 \\ 21 & 0 & 4 & 6 & 60 & 58 & 55 & 15 & 18 & 20 & 55 \\ 20 & 4 & 0 & 4 & 59 & 56 & 53 & 13 & 8 & 12 & 53 \\ 17 & 6 & 4 & 0 & 57 & 54 & 52 & 11 & 13 & 16 & 52 \\ 65 & 60 & 59 & 57 & 0 & 3 & 7 & 66 & 69 & 71 & 6 \\ 63 & 58 & 56 & 54 & 3 & 0 & 4 & 64 & 66 & 69 & 3 \\ 60 & 55 & 53 & 52 & 7 & 4 & 0 & 61 & 64 & 66 & 2 \\ 19 & 15 & 13 & 11 & 66 & 64 & 61 & 0 & 3 & 5 & 61 \\ 22 & 18 & 8 & 13 & 69 & 66 & 64 & 3 & 0 & 7 & 64 \\ 24 & 20 & 12 & 16 & 71 & 69 & 66 & 5 & 7 & 0 & 66 \\ 60 & 55 & 53 & 52 & 6 & 3 & 2 & 61 & 64 & 66 & 0 \end{pmatrix}.$$

To solve the associated mathematical problem, `Gurobi` (Gurobi Optimization, LLC, 2022) was used as an exact MILP solver. The optimization process required 31.2 h to obtain the optimal solution. The value of the objective function was 207 km, containing an MVR of 74 km, whose main tour, 0–3–2–1–0, is traveled by truck 1 with trailer 1 attached. Customer 2 serves as a trailer parking place for the sub-tour 2–8–7–9–2 with the truck. The remaining customers are served by truck 3 on a PTR, 0–6–5–4–10–0.

It can be seen that only a trailer is required to supply these 10 customers. We studied the effect of trailers in the solution and compared the results with those obtained if only trucks were used. In such a case, and after a runtime of 6.23 h, we obtained the following results: the value of the objective function was 232 km; truck 1 distributes feed to customers 1, 2, and 3, traveling 46 km; truck 2 travels 53 km and serves customers 7, 8, and 9; and truck 3 performs the route 0–4–5–10–6–0, whose length is of 133 km.

As can be seen, when the company introduces trailers, a reduction is achieved not only in the total length traveled (which decreases by 25 km) but also in the number of drivers required (which decreases from 3 to 2).

Moreover, we can deduce that when using only trucks, we are faced with an MC-VRP. However, because of the large amount of feed that the cooperative must distribute daily, we suggest the incorporation of trailers to accommodate the use of the MC-TTRP model. Furthermore, given the existence of access restrictions to some farms, this model seems appropriate. The purchase or rental of these additional vehicles can indeed be a significant initial investment, but the benefits usually compensate in the long term because, among other things, it is not necessary to hire more drivers.

To study the increase in the computation time when the instance is slightly modified, we tried to solve this problem when adding a new VC. By having one more node, the number of variables involved increases considerably, and this causes the execution time to exceed two weeks.

Agricultural cooperatives often must supply a large number of members, and it is not feasible to take more than 15 days to solve a problem with 11 customers. As stated in Subsection 2.3.1, the cooperative that motivated our study receives orders from approximately 40 customers per day. Furthermore, certain occurrences could require a sudden reorganization of the routes. This issue encouraged us to consider designing other solving approaches for this problem. In particular, as we will see in the next chapter, we have developed different heuristics that quickly and efficiently solve the problem of multi-compartmental trucks and trailers.

Chapter 3

Heuristics for the multi-compartment truck and trailer routing problem

Given that MC-TTRPs have an NP-hard complexity, optimal solving via exact methods for large-size instances is computationally expensive. Thus, the use of approximated techniques, such as heuristics, becomes necessary in order to obtain quality solutions in a reasonable time. Therefore, we introduce and implement different two-phase heuristic algorithms. The first phase coincides for all the approaches: an initial solution is generated through a constructive heuristic algorithm based on concepts from the classic Clarke–Wright algorithm. In the second phase, the initial solution is improved by different metaheuristics that will be detailed in the current chapter.

Initially, our algorithms were tested on 21 instances that were converted from the classic TTRP. It is observed that one of the methodologies achieves better solutions than the others. Thus, a more exhaustive analysis of this method is conducted. The results of our computational study prove the effectiveness of our proposal; the algorithm always finds a feasible solution, which in small-sized problems it is proven to be of good quality. In addition, the algorithm outperforms previous approaches for certain TTRP instances. Furthermore, an application of the model proposed in Chapter 2 and heuristics is demonstrated in the field of agricultural logistics by comparing the obtained results.

The contents of this chapter are collected in Davila-Pena et al. (2023) and Davila-Pena et al. (2022c).

Contents

3.1 Introduction

The effectiveness of the Clarke–Wright algorithm (Clarke and Wright, 1964) in building a solution for different VRPs and the requirement to solve MC-TTRPs that involve a relatively high number of customers served as motivation to modify this heuristic algorithm for the case of the MC-TTRP. To the best of our knowledge, only Derigs et al. (2013) reported adaptation of this algorithm to build an initial TTRP solution, although no details about such adaptation were provided. In addition to this constructive method, we propose several metaheuristic approaches to improve the initial solution obtained. These are based on tabu search and large neighborhood search methodologies. Specifically, we provide four algorithms: two of them are completely independent, the iterated tabu search (ITS) and the adaptive large neighborhood search (ALNS); while the other two combine ideas from both, the hybrid ALNS with tabu search (ALNS-TS) and the penalized ALNS-TS (PALNS-TS).

Both the constructive and improvement phases are integrated into novel two-stage algorithms to solve the MC-TTRP. A corresponding computational study was conducted through a series of instances created from other existing ones in the literature, showing that ITS achieved better results than the ALNS-based algorithms. For this reason, a more detailed study of the ITS is carried out, adapting the implementation of this algorithm to the standard TTRP and comparing the results obtained using our method with previous works. A deeper analysis of the new 21 instances is also performed, providing additional aspects to those shown in the comparison with the other algorithms. However, these problems could not be benchmarked with the exact model due to the computational time required. In contrast, a series of small-sized real-world problems were solved, achieving solutions that are competitive and close to those provided by the exact method.

The remainder of this chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 presents the two-stage heuristics to solve the MC-TTRP. Section 3.4 reports the

computational results for the designed heuristics on the MC-TTRP, using a set of instances adapted from those in literature and data of a real application. Finally, Section 3.5 summarizes the main conclusions of our study.

3.2 Related work

As explained in Chapter 2, the MC-TTRP arises as a combination of the TTRP and the MC-VRP. In order to gain insight into which solving methods may be suitable for this problem, we have investigated which techniques are most commonly used when tackling the two underlying routing problems.

Regarding TTRP solution methods, heuristics are popular approximation-based strategies for solving medium- to large-scale instances. In fact, heuristics have been used in the solutions of several VRP variants, with Lespay and Suchan (2021) being one of the most recent references. That study considered the problem of a food company's distribution center. This was solved by constructing an initial solution, which was subsequently improved using a guided local search. Gerdessen (1996) proposed constructive and improvement heuristics for solving the VRPT. Semet (1995) described a two-stage heuristic method for obtaining PACVRP solutions: the first phase of the algorithm involves assigning trailers to trucks and determining the optimal allocation between customers and trucks/vehicles, and then, the second phase builds the routes. Other works, such as those by Chao (2002) and Scheuerer (2006), also proposed two-phase methods, where they first defined an initial solution by applying constructive procedures and then used improvement metaheuristics based on techniques such as tabu search (Glover and Laguna, 1998). Later, Caramia and Guerriero (2010a) combined a mathematical programming and local search approach to solve the TTRP, and they compared their results with those of Chao (2002) using a set of benchmarks. Furthermore, the TTRP can be addressed using a metaheuristic approach, as in the work by Lin et al. (2011), where a simulated annealing algorithm was designed to find approximate TTRP solutions according to given time windows, achieving improved results in 11 of the 21 instances of Chao (2002). In addition, in an original research, Derigs et al. (2013) analyzed different variants of the TTRP and proposed two-stage heuristics for solving these problems, starting by building an initial solution and then moving to an improvement phase combining techniques such as local search (LS) and large neighborhood search (LNS). The behavior of the heuristics created for the TTRPTW was compared with that of the heuristic proposed by Lin et al. (2011). Depending on the TTRP variant considered, the authors applied a specific construction heuristic and, among them, an adaptation of the Clarke–Wright savings algorithm stood out. Subse-

quently, Parragh and Cordeau (2017) tailored an ALNS algorithm for the TTRPTW. This methodology has also been adapted to other routing problems with trailers, as can be seen in Drexl (2021). In terms of exact solution methods, recent references include the paper by Parragh and Cordeau (2017), which also proposes a branching and pricing algorithm for the TTRPTW. It provides an ALNS algorithm, as mentioned, to obtain good initial columns. Compared with existing metaheuristic algorithms, such as those designed by Lin et al. (2011) and Derigs et al. (2013), they obtained highly competitive results. Some instances with up to 100 customers were optimally solved. Rothenbächer et al. (2018) also solved the TTRPTW exactly using a branching, bounding, and cutting algorithm. Their computational studies showed that their algorithm outperforms existing approaches on the TTRP and TTRPTW benchmark instances used in the literature. To solve the XSTTRP, Accorsi and Vigo (2020) developed a fast and efficient hybrid metaheuristic based on a four-phase solution approach, in which the main improvement phase consists of an iterated local search.

Concerning the solving of multi-compartment problems, different approaches have been followed in recent years based on heuristics and metaheuristics. Simple constructive algorithms, such as the Clarke–Wright algorithm, have also been successfully adapted in this context, as can be seen in the literature. El Fallahi et al. (2008) compared a constructive algorithm, memetic algorithm, and tabu search, concluding that the results provided by the tabu search were slightly better, although it required more computation time. Muyldermans and Pang (2010) used the Clarke–Wright savings algorithm to obtain a feasible initial solution. Subsequently, they performed a local search with movements taken from the literature and improved the quality of the solution previously obtained through a metaheuristic based on a guided local search. They performed a sensitivity analysis on certain parameters (number of customers and their demands, depot location, vehicle capacity, or number of products). Their computational study included a comparison with the work of El Fallahi et al. (2008). Derigs et al. (2011) considered a model with a homogeneous fleet, i.e., all vehicles have the same number of compartments, all with equal capacities. This problem is a particular case of that addressed in the current thesis. They implemented their own benchmarks and a collection of optimization methods capable of obtaining high-quality solutions, which covered a wide range of alternative approaches to construction, such as LS, LNS, and metaheuristics. Coelho and Laporte (2015) defined and compared four categories of multi-compartment problems. They proposed two formulations for each case and presented a branching and cutting algorithm to solve single- and multi-period cases containing up to 50 and 20 customers, respectively. Mendoza et al. (2011) proposed a set of constructive heuristics to solve the MC-VRPSD, which included stochastic versions of the nearest neighbor, nearest insertion, and

savings-based approaches, adapted to the multi-compartment scenario.

Among the most recent investigations of solution methods for the MC-VRP, we highlight those by Henke et al. (2015) and Henke et al. (2019). Starting from a real problem of collecting glass containers, a model formulation and branch-and-cut algorithm for solving the problem to optimality were presented. The performance of the proposed algorithm was evaluated through extensive numerical experiments. Furthermore, the economic benefits of introducing compartments to vehicles were investigated. Silvestrin and Ritt (2017) proposed a tabu search heuristic algorithm and integrated it with an iterated local search to solve the MC-VRP. In several experiments, they analyzed the performance of the algorithm and compared it with results in the literature, finding that it produces better solutions than those provided by other existing heuristic algorithms. They considered an initial solution obtained by the Clarke–Wright savings algorithm extended to handle multiple compartments. Metaheuristics based on iterated local searches have shown very good behavior in various VRP variants (cf. Alvarez et al., 2018, who proposed efficient metaheuristics based on iterated local search and simulated annealing). Alinaghian and Shokouhi (2018) presented a new mathematical model for the multi-depot MC-VRP. They designed a hybrid algorithm composed of ALNS and variable neighborhood search (VNS). The results were compared to the exact solutions of small instances and compared with each other in large instances. Ostermeier and Hübner (2018) proposed an MC-VRP with a fleet of vehicles with flexible compartments. The aim of their work was to demonstrate the benefits of considering a mixed fleet consisting of both single-compartment and compartmentalized vehicles. The problem was solved using LNS. Ostermeier et al. (2018) included loading constraints in the previous MC-VRP variant, and developed a branch-and-cut and an LNS algorithm to solve the resulting problem. Ostermeier et al. (2020) introduced a typology for MC-VRPs and extensively reviewed the existing literature. They also made suggestions for future research.

3.3 Two-phase heuristic approaches

As already stated, we have proposed different algorithms for solving the novel MC-TTRP model. All of them are two-phase methods that consist of (1) a constructive heuristic that creates a feasible solution and (2) a metaheuristic approach that iteratively improves the solution provided in the previous stage. Moreover, our algorithms can consider additional features, such as fleet limitations, and solve the classic TTRP. The following subsections describe the proposed methods in detail, starting with the construction phase, common to all algorithms.

3.3.1 Construction phase

For the first phase, we designed an ad hoc Clarke–Wright algorithm (CW) capable of handling TTRPs with compartments. The CW is a popular constructive heuristic for the basic VRP. Its strategy is to build a feasible solution based on the notion of savings in the routing cost.

Typically, the CW begins by creating a starting solution for which all routes start at the depot, visit one customer, and return to the depot. It continues by computing the savings of joining each pair of these routes. Throughout each iteration, the CW considers the savings in descending order to choose which routes to merge, given that such merging is feasible. It stops when all customers have been served. Figure 3.1 shows a schematic diagram of its typical implementation for solving VRP instances.

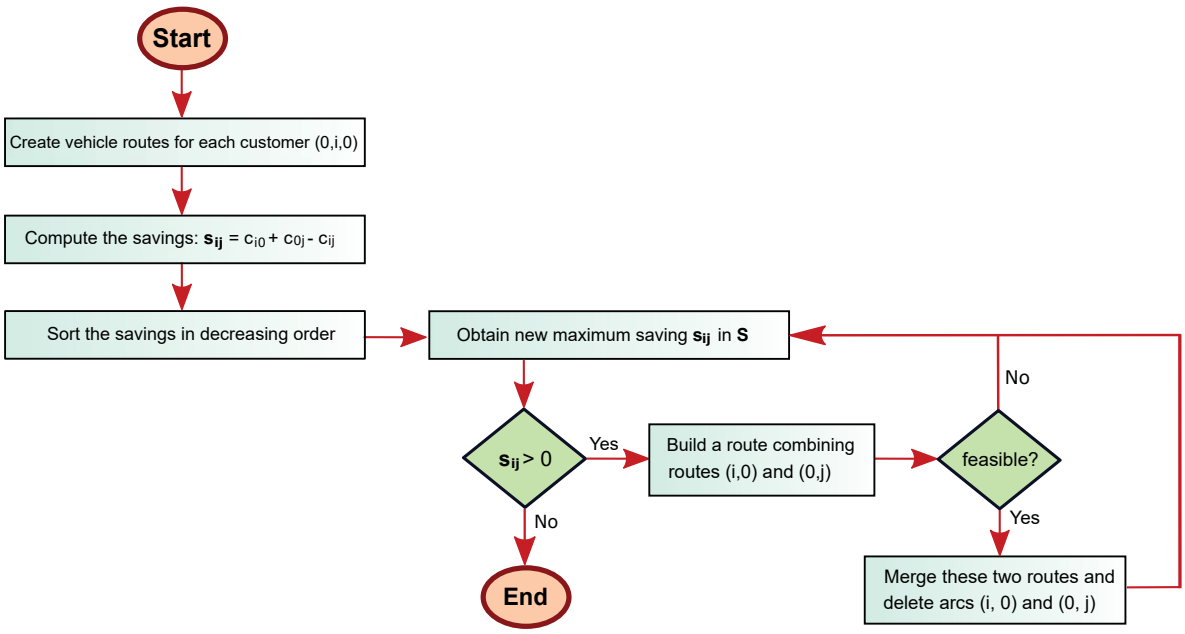


Figure 3.1: Clarke–Wright savings algorithm for the VRP.

The pseudocode for our proposed CW is shown in Algorithm 3.1. First, the initial parameters, such as the total number of customers and number of VCs (n and n_1, respectively), matrices of demands and distances (d and C), vehicle capacities (Q_T and Q_L), vehicle compartment configuration (H^T and H^L), and number of different types of products (f), are defined. Moreover, unlike classic CW, our heuristic considers two matrices of routes: R and $\hat{R}$. The former is updated considering those tours directly connected to the depot, while $\hat{R}$ considers the sub-tours whose roots are VCs. Consequently, as indicated in line 3 of Algo-

Algorithm 3.1 Savings-based construction heuristic for the MC-TTRP

procedure CW_MC-TTRP(n, n_1, d, C, Q_T, Q_L, H^T, H^L, f)
  Create $N_1 = \{1, \ldots, n_1\}$ and $N_2 = \{n_1 + 1, \ldots, n\}$ ▷ Initialization.
  Create R and $\hat{R}$
  Create S and $\hat{S}$
  Create matrices H^T_{rem}, H^L_{rem}, and $\mathcal{T}$
  $S_m = 1$
  while $S_m > 0$ **do** ▷ Constructive heuristics.
    S_m = Maximum saving in $\{S, \hat{S}\}$
    if $S_m \in S$ **then**
      Extract coordinates $[i, j]$ of $S_m \in S$
      Check type of routes r_i and r_j using $\mathcal{T}$
      if r_i and r_j can be merged **then**
        Apply fusion ▷ Different fusion cases can be seen in the description.
        Update $\{R, \hat{R}, S, \hat{S}, H^L_{\text{rem}}, H^T_{\text{rem}}, \mathcal{T}\}$
      else
        Update $\{S, \hat{S}\}$
      end if
    else
      Extract coordinates $[i, j]$ of $S_m \in \hat{S}$
      l = Check last customer using r_i or r_j and R
      if r_i or r_j can be converted into a sub-tour **then**
        Create MVR
        Update $\{R, \hat{R}, S, \hat{S}, H^L_{\text{rem}}, H^T_{\text{rem}}, \mathcal{T}\}$
      else
        Update $\{\hat{S}\}$
      end if
    end if
  end while
  `sol_route` = Create route using R, $\hat{R}$, and $\mathcal{T}$
  Add disconnected customers in `sol_route`, and update $\{\mathcal{T}, H^T_{\text{rem}}, H^L_{\text{rem}}\}$ ▷ Refinement.
  Adjust vehicle fleet creating or completing MVRs in `sol_route`, and update $\{R, \hat{R}, \mathcal{T}, H^L_{\text{rem}}, H^T_{\text{rem}}\}$
  Adjust vehicle fleet splitting routes in `sol_route`, and update $\{R, \hat{R}, \mathcal{T}, H^L_{\text{rem}}, H^T_{\text{rem}}\}$
  Adjust vehicle fleet switching residual routes in `sol_route`, and update $\{R, \hat{R}, \mathcal{T}, H^L_{\text{rem}}, H^T_{\text{rem}}\}$
  Apply tour improvement in `sol_route`, and update $\{R, \hat{R}, \mathcal{T}\}$
  Apply local search movements in `sol_route`, and update $\{R, \hat{R}, \mathcal{T}, H^L_{\text{rem}}, H^T_{\text{rem}}\}$
  `cw_sol` = Create final route using R, $\hat{R}$, and `sol_route`
  return(list(`cw_sol`, $\mathcal{T}$))
end procedure

rithm 3.1, we initialize both matrices as follows:

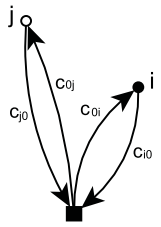
$$R = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ \vdots & \vdots & \vdots \\ 0 & n & 0 \end{pmatrix} = \hat{R},$$

assuming that routes $(0, i, 0)$ for $i \in N_1$ are traveled by complete vehicles and routes $(0, j, 0)$ for $j \in N_2$ are covered by trucks alone. Each customer's corresponding row represents the

previous and next customer in the transport network, as appropriate.

Furthermore, at the beginning of the proposed algorithm, two saving matrices are created: S and $\hat{S}$. The entries of S are computed as $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ (for $i, j \in N, i \neq j$)[1], which represents the standard savings matrix. This scenario is illustrated in Figure 3.2. However, because we have two different classes of customers and given that VCs can serve as trailer parking places, we also consider savings $\hat{s}_{ij}$. These values are calculated as follows:

- $\hat{s}_{ij} = s_{ij}$ for $i, j \in N_1$ or $i, j \in N_2$.
- $\hat{s}_{ij} = c_{j0} + c_{0j} - c_{ij} - c_{ji}$ for $i \in N_1$ and $j \in N_2$. These savings are the result of removing route $(0, j, 0)$ and converting it into a sub-tour whose root is i. That is, we will have the route $(0, \ldots, i, j, i, \ldots, 0)$ instead of $(0, j, 0)$ and $(0, \ldots, i, \ldots, 0)$. Figure 3.3 illustrates this situation.
- $\hat{s}_{ij} = c_{i0} + c_{0i} - c_{ij} - c_{ji}$ for $j \in N_1$ and $i \in N_2$. These savings are analogous to the previous ones, swapping i and j in their type of customer.



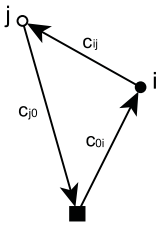
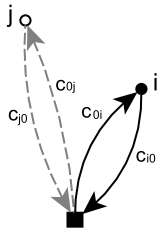


Figure 3.2: Calculation of s_{ij} for $i, j \in N$.



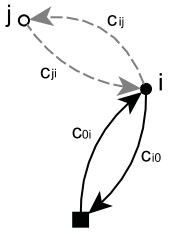


Figure 3.3: Calculation of $\hat{s}_{ij}$ for $i \in N_1$ and $j \in N_2$.

Hence, matrix $\hat{S}$ will have the following structure:

$$\hat{S} = \left(\begin{array}{ccc|ccc} s_{1,1} & \cdots & s_{1,n_1} & \hat{s}_{1,n_1+1} & \cdots & \hat{s}_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ s_{n_1,1} & \cdots & s_{n_1,n_1} & \hat{s}_{n_1,n_1+1} & \cdots & \hat{s}_{n_1,n} \\ \hline \hat{s}_{n_1+1,1} & \cdots & \hat{s}_{n_1+1,n_1} & s_{n_1+1,n_1+1} & \cdots & s_{n_1+1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hat{s}_{n,1} & \cdots & \hat{s}_{n,n_1} & s_{n,n_1+1} & \cdots & s_{n,n} \end{array}\right).$$

We must also consider the features related to compartments. As shown in line 5 of Algorithm 3.1, we create matrices H^T_{rem}, H^L_{rem}, and $\mathcal{T}$. The former two are initialized as H^T and H^L, respectively, and are updated by setting the compartments that have already been

[1]Moreover, $s_{ii} = 0$ for all $i \in N$.

used to -1. Matrix $\mathcal{T}$ plays a fundamental role in this implementation because it contains all information about the vehicles' loading procedures; it indicates the allocation details of each customer's demand for its corresponding vehicle.

Once all initial parameters have been created, our CW implementation follows an iterative process, where different routes are merged based on the maximum saving, S_m (lines 7–28 of Algorithm 3.1). Thereafter, r_i and r_j are considered to be routes containing customers i and j, respectively. By checking matrix $\mathcal{T}$ (line 11), the algorithm knows which vehicles cover each of these routes; thus, we can determine the PTRs, PVRs, and MVRs. Therefore, our proposed method repeats a set of steps, while S_m, the maximum saving of S and $\hat{S}$, is positive (stop condition). Thereafter, each iteration of the algorithm has two discernible cases: $S_m \in S$ or its opposite, $S_m \in \hat{S}$.

The first case occurs when S_m belongs to S (lines 9–17 of the pseudocode). Our heuristic applies the classic routing merge of the CW, which considers the type of customers (VCs or TCs) involved in such saving, which is a decisive factor in the type of route generated. Thus, four different unions between routes (by connecting customer i to customer j) can be conducted, as long as i is the first customer of route r_i and j the last one of route r_j:

1. **Merging two PVRs:** If routes r_i and r_j are PVRs covered by different complete vehicles, where the trucks are not yet loaded, the algorithm must count the number of unavailable compartments between both trailers. Two situations arise from this: (i) If the number does not exceed the number of trailer compartments, our method can move all goods to one trailer, emptying the other. (ii) If the number exceeds the number of trailer compartments, our method starts to load one of the trucks as long as the total number of available compartments is sufficient to meet the demands of both routes. Thus, it moves the goods from the other trailer to the chosen complete vehicle, giving preference to the filling of the trailer.

2. **Merging two PTRs:** When both routes r_i and r_j are PTRs, our heuristic can merge them as long as the number of unavailable compartments between both trucks does not surpass the number of truck compartments. In this case, the goods are moved to one of the trucks, leaving the other free.

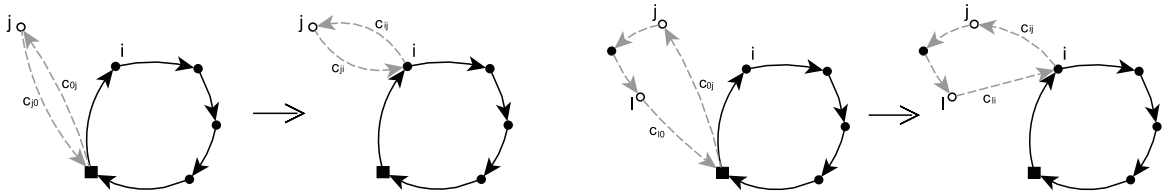
3. **Combining an MVR with a PVR:** If one of the routes is an MVR and the other is a PVR with an empty truck, say r_i and r_j, respectively, then their union is possible when the demands already met by route r_j fit in the trailer of route r_i. That is, when the trailer of route r_i has a sufficient number of available compartments to move goods from the trailer of route r_j.

4. **Inserting route r_i in a PTR:** If route $r_i = (0, i, 0)$, where i is a VC, and route r_j is a PTR, then merging both routes is possible when the number of available compartments in the truck is sufficient to hold i's demands.

For all the above considered fusions, updating matrices $\{R, \hat{R}, S, \hat{S}, H^T_{\text{rem}}, H^L_{\text{rem}}, \mathcal{T}\}$ is necessary to contemplate the changes conducted, as indicated in line 14. If merging cannot be completed, we must simply update matrices S and $\hat{S}$ by setting their corresponding entries to zero.

The second case occurs when $\hat{S}$ contains S_m, which means that the current saving arises from merging a PTR and PVR (obtaining a sub-tour as a result). Lines 18–27 of Algorithm 3.1 correspond to this situation. In such a case, it is clear that customers involved in S_m have different types. Let us see how to proceed when $i \in N_1$ and $j \in N_2$, because the other case is analogous. Our method can only execute merges when r_i is a PVR with an empty truck and r_j is a PTR. With this in mind, two possibilities arise:

- If $r_j = (0, j, 0)$, the algorithm uncouples the truck of route r_i and hitches the truck of route r_j, loaded with j's demands, to the trailer of route r_i. This proceeds as illustrated in Figure 3.4a.
- Suppose j is the first customer of the route but not the only one. In this case, we must also consider the last customer of route r_j, say, l. To join routes r_i and r_j, it is necessary that $c_{0j} + c_{l0} - c_{ij} - c_{li}$ is positive. We illustrate this merging in Figure 3.4b.



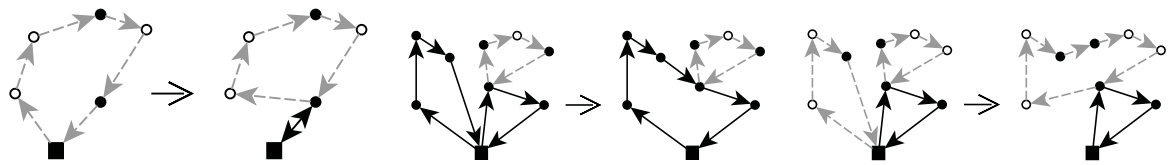
(a) Before (left) and after (right) merging when r_j has a single customer.

(b) Before (left) and after (right) merging when r_j has multiple customers.

Figure 3.4: Merges performed when $S_m \in \hat{S}$.

Refining the construction phase

Once we have exited the main loop and no saving is positive, the algorithm must check whether the resulting route configuration is feasible. The following post-processing (lines 30–35 of the pseudocode) functions are applied in the order in which they are presented to verify



(a) Convert a PTR into an MVR. *(b) Merge a PVR into the main tour of an MVR.* *(c) Merge a PTR into the sub-tour of an MVR.*

Figure 3.5: Adjusting vehicle fleet by creating or completing MVRs.

such feasibility, correct the tours if needed, improve the quality of the results, and limit the vehicle fleet:

- **Add disconnected customers**: This checks if there exists a route $r_i = (0, i, 0)$, which means that customer i has not yet been served. During the initialization of R, we mentioned that the starting routes should be covered by complete vehicles or trucks alone, depending on whether i is a VC or TC, respectively. Considering this, we supply customer i with its corresponding type of vehicle, contemplating the compartment load in matrix $\mathcal{T}$. After that, we are able to add the route $(0, i, 0)$ to the solution.

- **Adjust vehicle fleet by creating or completing MVRs**: The aim of this function is to use all the available trailers in the solution obtained by eliminating excess PTRs and PVRs, either by changing or merging them into MVRs, respectively. First, because all PTRs with at least a VC can be transformed into MVRs, we convert as many PTRs into MVRs as necessary to reach the required number of trailers. Subsequently, the lowest quality PVRs and PTRs are selected to be joined either to the main tour or to one of the sub-tours of an existing MVR, provided that the result is a feasible solution. This procedure is illustrated in Figure 3.5.

- **Adjust vehicle fleet by destroying routes:** It is assumed that the routes to be deleted or preserved have already been selected to adjust the fleet. Iteratively, an attempt is made to insert the first group into the second group. The method of merging these routes depends on their nature. For instance, a PTR could only be part of a sub-tour of an MVR or join another PTR. If there are no feasible insertions of a specific route to be deleted, it is split in half. The algorithm will try to add it again in the next iteration, repeating this process until there are no more residual routes or until they cannot be divided anymore.

- **Adjust vehicle fleet by switching residual routes:** After applying the previous function, if there are still routes to be eliminated, they will consist of a single customer.

This last procedure to adjust the fleet exchanges these residual customers for others that have less load on the routes to be preserved while aiming to keep the cost function from increasing as much as possible. Once the exchange is conducted, the customer to be inserted requires less space. In this manner, the algorithm aims to relocate it to one of the existing routes, repeating the process until there are no residual routes left.

- **Apply tour improvement:** This function performs 2-opt, 3-opt, and 4-opt* moves on the resulting routes[2], including the sub-tours, individually. For each route, the move that leads to the greatest cost reduction, if any, is applied.

- **Apply local search moves:** Finally, a local search is applied to the solution obtained from Algorithm 3.1. This iteratively applies a set of small modifications to intensify the search on routes close to the current solution. To implement these moves, we were inspired by the work of Derigs et al. (2013), where a hybrid approach for solving the TTRP, combining local search and large neighborhood search, was presented. We adapted some of the moves implemented in their local search to the MC-TTRP. Therefore, this function contains both specific TTRP moves as well as the standard 2-opt, 2-opt*, exchange, and relocate operators commonly used in many VRP implementations. Figure 3.6 shows some of the moves performed in this step.

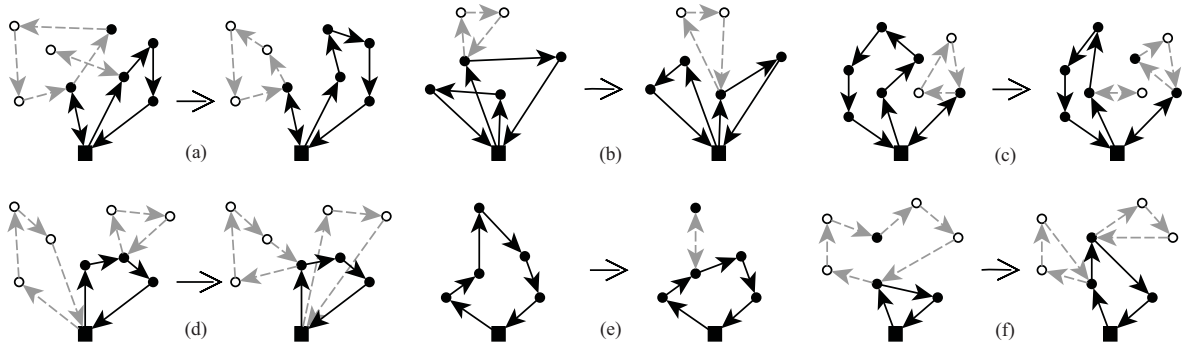


Figure 3.6: Local search moves for the MC-TTRP. (a) Replacing the single customer of a sub-tour with a VC, (b) replacing a sub-tour root with another VC, (c) replacing a main tour customer with a TC, (d) exchanging a PTR and a sub-tour and relocating the parking place, (e) moving a main tour customer to a new sub-tour, (f) moving a sub-tour customer to the main tour and splitting the sub-tour.

Finally, as shown in Algorithm 3.1, our constructive heuristic for the MC-TTRP completes the work, returning a feasible route and the loading configuration of all the vehicles involved in the transport network. Once this feasible initial solution has been obtained, the

[2]The 4-opt* procedure uses a subset of potential 4-opt moves, cf. Renaud et al. (1996).

second phase of our proposal will attempt to improve such solution by means of different metaheuristics. The following subsections detail each of these approaches.

3.3.2 Iterated tabu search (ITS)

As stated before, one of the algorithms implemented for the second phase of our proposal is an iterated tabu search (ITS). We developed this metaheuristic based on previous related works, such as that by Cordeau and Maischberger (2012) for some VRP variants and the approach implemented by Silvestrin and Ritt (2017) for the MC-VRP.

The ITS starts from a feasible solution, which, in our case, is the solution provided by the constructive heuristic described in Subsection 3.3.1. This procedure is usually based on two main actions: a tabu search and a perturbation. The former involves a set of local moves applied sequentially to improve the solution, with the ability to accept slightly worse solutions or to move through the infeasible region when the search is stuck. Moreover, to avoid returning to already-visited solutions, they use the short-term memory strategy known as the tabu list. In the case of the perturbation process, the objective is to escape from local optima by exploring the vicinity of the current solution through small changes in the routes. As a general overview, the ITS aims to improve the best-known solution by combining the diversification provided by the perturbation with the intensification and diversification produced by the tabu search.

Algorithm 3.2 describes the scheme of our proposed ITS, of which we will now present a brief outline. The input parameters are the solution returned by the CW, which is the starting point, and a maximum number of iterations used as a stopping criterion. In the main loop (lines 5–14), the current solution is perturbed at each iteration after having applied the tour improvement function (presented in Subsection 3.3.1). This modified solution will be used as an initial guess for the tabu search. In turn, as can be seen in line 8, this latter method returns its current solution, which will be the one to be perturbed in the next iteration, and its best solution, which can update the best overall solution (lines 9–11). In this way, throughout the iterations, the algorithm perturbs and searches the current solution, which is initially the best-known solution. However, as the algorithm progresses, there is a probability of working with another solution to avoid stagnation (line 12). Finally, the algorithm terminates when the stopping conditions are fulfilled, and it outputs the best solution found during the search.

Regarding the perturbation procedure, this is a key point in our proposal because it guarantees diversity in the method. First, a random number, ϕ, is chosen between 1% and 10% of the total number of customers. Then, ϕ customers are randomly selected and removed from the solution, to be subsequently reinserted. For the deletion and insertion operations, we

Algorithm 3.2 Iterated tabu search for the MC-TTRP

```
1: procedure ITS_MC-TTRP(cw_sol, max_iterITS)
2:     best_sol ← cw_sol
3:     current_sol ← cw_sol
4:     iterITS = 0
5:     while iterITS < max_iterITS do
6:         current_sol_it ← Improve tours in current_sol
7:         current_sol_it, φ ← Perturbs the current_sol_it
8:         current_sol_it, best_TS_sol ← TabuSearch(current_sol_it, φ, iterITS, max_iterITS)
9:         if cost(best_TS_sol) < cost(best_sol) then
10:            best_sol ← best_TS_sol
11:        end if
12:        with probability (iterITS/max_iterITS)^2, current_sol ← best_sol
13:        iterITS = iterITS + 1
14:    end while
15:    return(best_sol)
16: end procedure
```

use the generalized insertion (GENI) procedure and unstringing and stringing (US) algorithm proposed by Gendreau et al. (1992). This is followed by 3-opt and 4-opt* local search algorithms to improve the routes. Depending on the type of customer to be deleted or inserted, as well as its position in the solution, different possibilities of moves can arise. Some examples are shown in Figure 3.7. Each customer is inserted into the route and position which minimizes the increase in the solution cost. If the perturbation failed to introduce the nodes that have been removed, we allow a threshold of infeasibility in the perturbed solution.

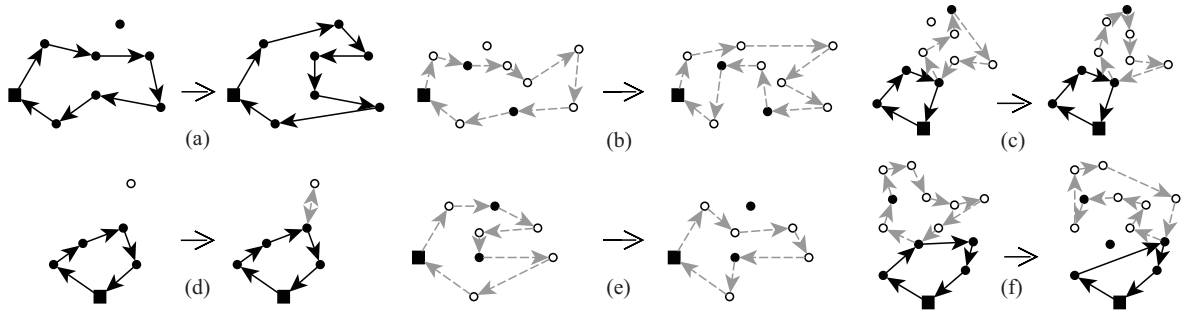


Figure 3.7: Removal and insertion operations in the perturbation for the MC-TTRP. (a) Inserting a VC into a PVR, (b) inserting a TC into a PTR, (c) inserting a TC into a sub-tour, (d) converting a PVR into an MVR by creating a sub-tour with a single TC, (e) removing a VC from a PTR, (f) removing a VC that works as parking place from an MVR.

After the solution is perturbed, the algorithm performs the tabu search, whose implementation requires an additional explanation. This is an iterative procedure, where the strategy is essentially to apply the best possible MC-TTRP local search moves (some of which are shown in Figure 3.6) in the neighborhood of the current solution, as long as these moves are

not stored in the tabu list. The approach adopted for the tabu list is as follows. As it is known, a move involves a set of routes and the customers to be deleted or inserted into them. At each iteration, the tabu list stores for each move applied to the current solution, a set of pairs $\{customi, rj\}$, with $customi$ being the customer removed from route rj. In this way, the tabu search only accepts those moves where all the customers introduced in the routes are different from all the pairs $\{customi, rj\}$ stored in the tabu list. The exception to this rule occurs when the result improves the best solution found so far, in which case an aspiration criterion is applied. Consequently, the tabu list prevents the insertion of customers into the same routes for a number η of iterations. Each pair $\{customi, rj\}$ has an associated survival counter that controls the number of iterations that will be in the tabu list, and that requires to be updated at each iteration of the tabu search. When one of these counters reaches η, its corresponding pair is removed from the tabu list. The parameter η is initialized at the beginning of the tabu search and is chosen randomly from the uniform distribution on the interval $[1, \sqrt{n \cdot nr}]$, where n is the number of total customers, and nr corresponds to the number of routes in the current solution.

As the algorithm applies the best possible move, it may not improve the cost of the current solution or may even be infeasible. However, this is desirable because exploring new regions using worse or/and infeasible solutions prevents us from getting stuck in a local optimum. Therefore, to measure the total cost of each route, r, we consider the following objective function:

$$OF(r) = \text{dist}(r) + \alpha \cdot \text{Ex}(r), \tag{3.1}$$

where $\text{dist}(r)$ is the total distance traveled, α a parameter of penalty, and $\text{Ex}(r)$ denotes the excess load on both vehicles and compartments. To calibrate the α penalty, we follow the approximation proposed by Cordeau and Maischberger (2012), which we recommend referring to for further details about this mechanism. Briefly, the α penalty is initially set to 1 and then updated throughout the tabu search, depending on the excess load in the current solution. In the case of feasibility, i.e., if constraints are not violated, the penalty is decreased by a factor $1 + \gamma$. Otherwise, if the current solution has an excess load on its routes, α is increased by $1 + \gamma$. The parameter γ is randomly selected at the beginning of the tabu search from the uniform distribution on $[0, 1]$. Consequently, this update strategy produces an oscillatory effect between feasible and infeasible solutions.

In addition, as in other tabu search methods such as Cordeau and Maischberger (2012) or Silvestrin and Ritt (2017), a table with the most frequent moves is also used to penalize the cost function when the search is stuck in a local minimum. Thus, when the best possible move decreases the current solution cost, we reevaluate all possible moves according to a

Algorithm 3.3 Tabu search for the MC-TTRP

```
1: procedure TABUSEARCH(current_sol_it, φ, iterITS, max_iterITS)
2:     Choose η randomly from U[1, √(n · nr)] and create a new tabuList = {}
3:     Create a new freqPenList = {} and choose ζ randomly from U[0, 1]
4:     Initialize α = 1 and choose γ randomly from U[0, 1]
5:     best_TS_sol ← current_sol_it
6:     counter_iters_without_improvement = 0
7:     max_iters_without_improvement = √((max_iterITS − iterITS) · φ)
8:     while counter_iters_without_improvement < max_iters_without_improvement do
9:         Check all local search moves in the neighborhood of the current_sol_it
10:        Evaluate the cost of the moves, penalizing infeasible solutions using α (see (3.1))
11:        If the moves do not produce improvement, they are penalized using ζ and freqPenList (see (3.2))
12:        Apply the best possible move in current_sol_it according to tabuList
13:        if cost(current_sol_it) < cost(best_TS_sol) then
14:            best_TS_sol ← current_sol_it
15:            counter_iters_without_improvement = 0
16:        else
17:            counter_iters_without_improvement = counter_iters_without_improvement + 1
18:        end if
19:        if current_sol_it is infeasible then
20:            α = α · (1 + γ)
21:        else
22:            α = α/(1 + γ)
23:        end if
24:        Save the selected move in freqPenList
25:        Update tabuList
26:    end while
27:    return(current_sol_it, best_TS_sol)
28: end procedure
```

new objective function:

$$OF'(r, M) = OF(r) \cdot \left(1 + \zeta \cdot \sum_{ci \in cMr} freqPenList(ci, r)/it\right), \tag{3.2}$$

where M is a specific move, ζ is a penalty uniformly randomly chosen from $[0, 1]$, cMr is the set of customers involved in M to be inserted into a route r, $freqPenList$ is the table of frequencies, which returns how many times a customer ci entered in the route r, and it is the current iteration.

Algorithm 3.3 shows the main scheme for the implemented tabu search. In the first lines of the pseudocode (lines 2–7), the tabu list ($tabuList$) and the table with the most frequent moves ($freqPenList$) are created, and all the parameters explained above are initialized (η, α, γ, and ζ). Furthermore, a maximum number of iterations without improvement is set (line 7), which, in this case, will be the stopping criterion. During the main loop (lines 8–26), the tabu search evaluates all possible moves, selecting the best possible one according

to the constraints imposed by the tabu list. If the current solution is a local optimum, the algorithm selects the best move considering $freqPenList$ and expression (3.2). Once a new solution is created, we check if it increases the cost of the best one found during the tabu search (lines 13–18). Next, α is calibrated based on the solution feasibility (lines 19–23). Moreover, the algorithm adds the selected move information to $freqPenList$ (line 24). The tabu list is updated in line 25 by reducing the survival counter of the stored pairs. Notably, those pairs that reached the iteration limit inside the list are then released, and new pairs involved in the last applied move are included. Finally, after the algorithm reaches the stop condition, the tabu search returns both the best solution visited during the search and the current solution. The latter is from which Algorithm 3.2 will continue to work, perturbing it in the next iteration of the ITS algorithm.

3.3.3 Adaptive large neighborhood search (ALNS)

The adaptive large neighborhood search (ALNS) introduced by Pisinger and Ropke (2007) has shown to provide high-quality solutions for numerous VRP variants. Thus, we have also adapted this algorithm for the problem described in Chapter 2, the MC-TTRP. Considering the large number of components involved in this metaheuristic, we have chosen to organize the present subsection into parts, although this differs in the structure used for the explanation of the ITS.

Initial solution

We start with the solution generated by the adaptation of Clarke–Wright algorithm for the MC-TTRP, which is described in Subsection 3.3.1.

Removal and insertion operators

The proposed ALNS for the MC-TTRP is an iterative process that uses a set of removal $\mathcal{R}$ and insertion $\mathcal{I}$ operations. The strategy consists in selecting $\tilde{\phi}$ customers to remove from and reinsert into the solution according to a chosen pair (R, I) with $R \in \mathcal{R}$ and $I \in \mathcal{I}$. At each iteration, the value $\tilde{\phi}$ is randomly selected between 1% and 8% of the total number of customers using a beta distribution with parameters 2 and 5, whose graphical representation can be seen in Figure 3.8. As observed, the probability of generating smaller values is higher than obtaining the larger ones, allowing the solution to be diversified without destroying it too much.

The removal operators take as inputs a given solution (the candidate solution, which is

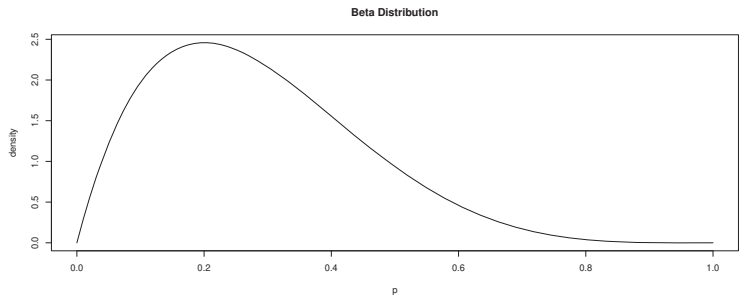


Figure 3.8: Beta distribution to calculate $\tilde{\phi}$.

initialized as the current solution) and the value of $\tilde{\phi}$, and they output the $\tilde{\phi}$ customers to be removed from the solution. We further detail the removal operators below:

- `Random_r`: random choice of $\tilde{\phi}$ customers.
- `Worst`: selection of the $\tilde{\phi}$ customers that add the highest cost in the current solution. That is, those that add a greater increase in distance in the routing configuration. The way to calculate the cost of a customer depends on whether it can act as a trailer parking place. If it is not, its cost is calculated as the sum of its two connecting edges. In case the customer is working as a parking place, then its cost is the maximum of the sums of the pairs of edges in which it is involved. For example, considering the route 0–1–2–6–7–8–2–3–0, the cost of customer 1 would be $c_{01} + c_{12}$, while the cost of customer 2 would be $\max\{c_{12} + c_{26}, c_{82} + c_{23}, c_{12} + c_{23}\}$. In the example of Figure 3.9, the cost of customer 2 would be $\max\{c_{12} + c_{26}, c_{82} + c_{29}, c_{10,2} + c_{23}, c_{12} + c_{23}\}$.

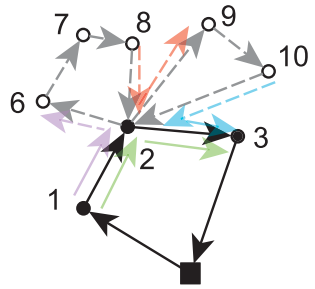


Figure 3.9: Cost of customer 2.

- `Worst_MT`: is an adaptation of the previous operator, where only main tour costs are considered. Thus, we do not differentiate whether a customer is acting as a trailer parking place or not. For the two aforementioned examples, the cost of customer 2 would be computed as $c_{12} + c_{23}$.

- `Shaw`: is introduced in Shaw (1998). A customer i is randomly chosen followed by its $\tilde{\phi}-1$ most related customers. According to Derigs et al. (2013), $\Gamma(i,j)$, the relation between customers i and j, is based on three aspects in the TTRP: geographical position, demands, and customer types, and is calculated as

$$\Gamma(i,j) = f_c \cdot \frac{c_{ij}}{c_{max}} + f_q \cdot \frac{|\sum_{f\in F} d_{if} - \sum_{f\in F} d_{jf}|}{d_{max}} + f_t \cdot t_{ij},$$

  where c_{max} represents the largest distance between two customers, d_{max} the highest demand among all customers, and t_{ij} is a binary variable which takes value 1 if customers i and j are of a different type, and 0 otherwise. Also, f_c, f_q and f_t are parameters measuring the importance of these aspects, and we took $f_c = 0.50$, $f_q = 0.25$, and $f_t = 0.25$.[3]

- `Shaw_Ali`: this operator is a modification of `Shaw` according to Alinaghian and Shokouhi (2018), where the relatedness formula is given by

$$\Gamma'(i,j) = f'_c \cdot c_{ij} + f'_r \cdot l_{ij}.$$

  In this case, l_{ij} takes value -1 if customers i and j are in the same route, and 0 otherwise. Also, f'_c and f'_r are parameters taking both the value 0.50.

- `Proximity-based`: a customer i is randomly chosen followed by its $\tilde{\phi} - 1$ closest customers.

- `Neighborhood`: selection of the $\tilde{\phi}$ customers with the largest difference between its average cost and its position cost. The position cost of a customer is calculated as in `Worst`, and its average cost is obtained by dividing its route length (that is, the length of the route to which this customer belongs) by the number of customers in the route.

- `Neighborhood_MT`: adaptation of `Neighborhood` where the position cost of each customer is computed as in `Worst_MT`.

After choosing the removal operator, we know the customers that must be removed. For the removal of the customers, we use the unstringing and stringing (US) algorithm proposed by Gendreau et al. (1992). We can see some examples of deletion moves in Figure 3.7. Note

[3] We took these values according to Alinaghian and Shokouhi (2018). Note that these authors consider a different relatedness formula, taking into account the different types of products and not considering the type of customers (they study an MC-VRP).

that it is possible to eliminate a customer that is the single customer of its route, thus releasing a vehicle (either a truck or a complete vehicle).

Once we have removed $\tilde{\phi}$ customers from the solution, we have to reinsert them. In the MC-TTRP, we must take into account the type of customers we need to insert and the destination routes (for instance, it is not possible to insert a TC into the main tour of an MVR). The insertion operators will determine in which routes and positions we should insert these customers. However, we must first know which insertions are feasible. This will also depend on whether the destination route is the same as the route of origin (that is, the route from which a specific customer has been removed), or not. We have implemented some insertion moves that combine the GENI procedure proposed by Gendreau et al. (1992) with some specifically created moves for the MC-TTRP. Tables 3.1 and 3.2 present an overview of all the possible insertions for VCs and TCs, respectively.

Table 3.1: Possible insertions for VCs in an MC-TTRP.

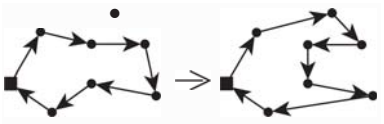
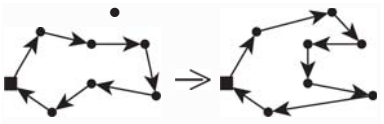
Destination route	Destination route = Origin route?	Type of insertion	Graphical example
PTR	Origin route ✗		-
	Different route ✔	On best position, according to GENI procedure	
	New route ✔	Create route only with this customer, if there are not available trailers but there are available trucks	
PVR	Origin route ✔ We only contemplate this insertion if the origin route was an MVR and the VC was in a sub-tour	Insert this customer on its best position, according to GENI procedure	
	Different route ✔	Insert this customer on its best position, according to GENI procedure	
	New route ✔	Create route only with this customer, if there is available fleet	

Table 3.1: Possible insertions for VCs in an MC-TTRP (continued).

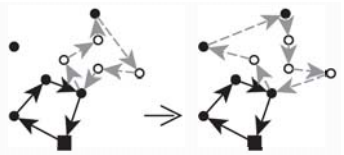
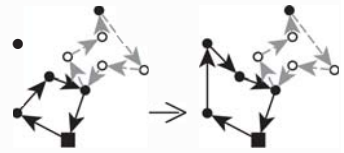
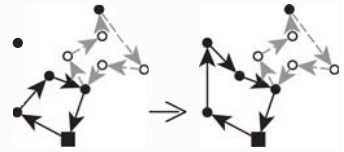
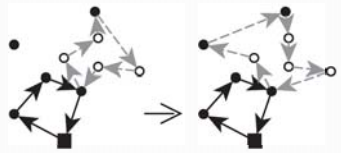
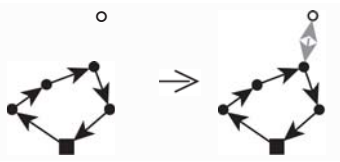
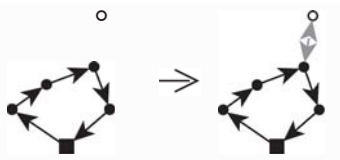
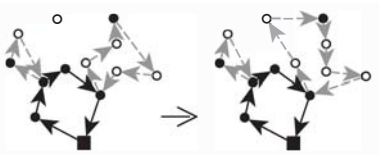
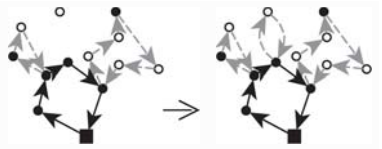
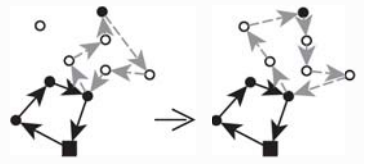
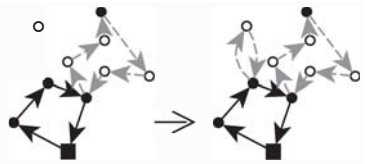
Destination route	Destination route = Origin route?	Type of insertion	Graphical example
MVR	Origin route ✔ This VC could originally be in the main tour or in a sub-tour. We have different insertion possibilities	If the VC was originally in the main tour, insert it on an already existing sub-tour, according to GENI procedure. Choose the sub-tour with the cheapest insertion	
		If the VC was originally in a sub-tour, insert it into the main tour, following the GENI procedure	
	Different route ✔	Insert this VC in the main tour, according to GENI procedure	
		Insert this VC into an existing sub-tour, according to GENI procedure. Choose the cheapest insertion from all the possible sub-tours	
	New route ✗		-

Table 3.2: Possible insertions for TCs in an MC-TTRP.

Destination route	Destination route = Origin route?	Type of insertion	Graphical example
PTR	Origin route ✗		-
	Different route ✔	On best position, according to GENI procedure	
	New route ✔	Create route only with this customer, if there is any available truck	

Table 3.2: Possible insertions for TCs in an MC-TTRP (continued).

Destination route	Destination route = Origin route?	Type of insertion	Graphical example
PVR	Origin route ✔ This means that now the route is PVR but initially it was an MVR (in the removal phase all sub-tours have been eliminated, one of which contained this TC)	Insert this customer as the only customer of a new sub-tour. The trailer parking place will be the VC on the main tour for which this insertion is the cheapest, regarding the VC was not the trailer parking place of this TC before the removal phase	
	Different route ✔ This route will turn into an MVR after this insertion	Insert this customer as the only customer of a new sub-tour. Choose the trailer parking place for which this insertion is the cheapest	
	New route ✗		-
MVR	Origin route ✔ This TC belonged to a sub-tour in this MVR before the removal phase	Insert this TC in a different sub-tour, if any, according to the GENI procedure. Choose the cheapest insertion from all the possible sub-tours	
		Create a new sub-tour with this TC. Choose the VC of the main tour for which this type of insertion is the cheapest, as long as this is not the sub-tour of origin	
	Different route ✔	Insert this TC into an existing sub-tour, according to GENI procedure. Choose the cheapest insertion from all the possible sub-tours	
		Create a sub-tour only with this TC. Choose the trailer parking place for which this insertion is the cheapest	
	New route ✗		-

Now, for the removed customers, we calculate the increment in the solution cost when we insert them into all their possible routes. The insertion operators we detail below will determine in which routes (from those that are possible) we should insert the customers. Note that most of the insertion operators are myopic: an iterative procedure is followed to insert the customers in the order in which they were removed. When a customer is inserted, the number of possible insertions for the remaining customers is restricted.

- `Greedy`: customers are inserted into a route and a position that minimizes the increase in the total cost.
- `Random_i`: customers are randomly inserted in one of their possible routes and positions.
- `p-Best`: customers are randomly inserted in one of their p cheapest insertions. The selection of p depends on the number of possible insertions (n_ins) for a customer. If there is only one possible insertion, $p = 1$; if the number of possible insertions is $\leqslant 10$, then $p = 2$; and if there are more than 10 possible insertions, then $p = \lceil 0.20 \cdot n\_ins \rceil$.
- `Regret-2`: in this case, customers are not inserted in the order they were removed. For the $\tilde{\phi}$ customers, we compute the difference between its best insertion and its second best insertion. Let x_{iq} denote the route and position where customer i has its q-th lowest insertion, and $\Delta_{i,x_{iq}}$ the cost of inserting customer i into x_{iq}. The regret value is in this case computed as $c_i^* = \Delta_{i,x_{i1}} - \Delta_{i,x_{i2}}$. The method chooses to insert customer $\hat{i} = \arg\max_i c_i^*$ into $x_{\hat{i}1}$. If there is a tie, we first insert the customer with the lowest insertion cost. Once a customer has been selected and inserted, we have to repeat the procedure for the remaining ones.
- `GIN`: following Alinaghian and Shokouhi (2018), this operator is an extension of the `Greedy` insertion, in which a degree of freedom is considered when choosing the best insertion position of a certain customer. All the possible insertion costs are added the value $c_{max}\mu\varepsilon$, where c_{max} is the maximum distance between customers, ε is randomly selected from the $U[0, 1]$, and μ is a parameter used for diversification of the solution ($\mu = 0.1$).
- `RIN`: this operator is an extension of the `Regret-2` insertion in the same way that `GIN` is for the `Greedy` operator. As in Alinaghian and Shokouhi (2018), $\mu = 0.1$.
- `Greedy_subtours`: is an adaptation of `Greedy`, in which the possible insertions are limited to those options that involve sub-tours.

- `Random_i_subtours`: is an adaptation of `Random_i`, in which the possible insertions are limited to those options that involve sub-tours.
- `p-Best_subtours`: is an adaptation of `p-Best`, in which the possible insertions are limited to those options that involve sub-tours.
- `Regret-2_subtours`: is an adaptation of `Regret-2`, in which the possible insertions are limited to those options that involve sub-tours.
- `GIN_subtours`: is an adaptation of `GIN`, in which the possible insertions are limited to those options that involve sub-tours.
- `RIN_subtours`: is an adaptation of `RIN`, in which the possible insertions are limited to those options that involve sub-tours.

After having applied one of the insertion operators described above, we return a feasible solution where all customers are present. Note that reinserting some customers may not be possible, in which case we output the solution we had before the removal phase.

Weight initialization

The removal and insertion operators will be selected in each iteration of the ALNS according to some weights. At the beginning of the algorithm, the insertion and removal matrices are initialized. As the procedure is analogous for both matrices, we will explain the initialization of the removal matrix. The dimension of the removal matrix is $4 \times |\mathcal{R}|$. Each column corresponds to a removal operator $R \in \mathcal{R}$ and in the rows we have the number of times operator R has been selected, $Times_R$, its weight, ω_R, its score, π_R, and the probability of the operator being selected, p_R. Naturally, $Times_R$ and π_R are initialized to 0. The weight, ω_R, is initially set to 1 and the probability of selecting operator R is computed as $p_R = \frac{\omega_R}{\sum_{R' \in \mathcal{R}} \omega_{R'}}$.

Tabu criterion

In order to diversify the method, we have considered a tabu criterion that prevents us from choosing the same removal and insertion operators in two consecutive iterations. Thus, at each iteration, we store in a tabu list the selected operators $R \in \mathcal{R}$ and $I \in \mathcal{I}$ with survival counters of 1. That is, in the first iteration of the method we have $|\mathcal{R}|$ and $|\mathcal{I}|$ removal and insertion operators available, respectively. However, in the following iterations there are only $|\mathcal{R}| - 1$ and $|\mathcal{I}| - 1$.

Improvement phase

We have implemented an improvement function that is carried out at each iteration. After reinserting the customers, the algorithm performs an intra-route local search in the candidate solution. This process consists in applying 2-opt, 3-opt, and 4-opt* moves on the resulting routes, including the sub-tours, individually. For each route, the move that leads to the greatest cost reduction, if any, is implemented.

Acceptance criterion

If the cost of the candidate solution is lower than the cost of the current solution, the candidate solution is accepted. Otherwise, we use an aspiration criterion based on simulated annealing: we accept worse solutions with a probability of $e^{\frac{-(\texttt{candidate\_cost-current\_cost})}{T}}$, where T denotes the temperature. At the beginning of the algorithm, the temperature is initialized as

$$T^1 = -\frac{0.005}{\log 0.5} \cdot \texttt{current\_cost},$$

and it will be updated at the end of each iteration as $T^{iter+1} = \delta \cdot T^{iter}$, where the cooling rate, δ, has been set to 0.999.

Weight updating

During the first 100 iterations of the algorithm, the weights for the removal and insertion operators are not updated. In this phase, scores are built for the operators based on the evaluation of the candidate solution obtained in each iteration:

- If the candidate solution improves the best solution, then the scores of the selected removal R and insertion I operators are incremented by σ_1, i.e., $\pi_R = \pi_R + \sigma_1$ and $\pi_I = \pi_I + \sigma_1$, with $\sigma_1 = 20$.
- If the candidate solution improves the current solution, then the selected removal and insertion operators are incremented by $\sigma_2 = 10$.
- If the candidate solution did not improve the current solution, there is still a probability that the operators get a positive score, and this happens if this worse solution is accepted. The acceptance of a worse solution is determined by the aspiration criterion explained above. In such a case, the selected removal and insertion operators are incremented by $\sigma_3 = 2$.

Once we have trained our algorithm for 100 iterations, the weights start to be updated. Throughout the iterations, the removal and insertion operators continue getting scores in the same way as at the beginning of the process. The weights for any operator j are updated as in Pisinger and Ropke (2007):

$$\omega_j^{iter+1} = \lambda \cdot \frac{\pi_j^{iter}}{Times_j^{iter}} + (1 - \lambda) \cdot \omega_j^{iter},$$

where λ has been selected as 0.1.

At each iteration, removal and insertion operators are selected by the use of a roulette wheel (by considering the cumulative probabilities), as long as they are not stored in the tabu list.

Stopping criterion

The algorithm terminates when either the number of maximum iterations or the maximum time allowed is reached.

Comprehensive overview of the proposed ALNS

Algorithm 3.4 outlines our proposed ALNS, of which we will now give a brief description. The input parameters are the solution returned by the CW (`cw_sol`), which is the initial solution, the sets of removal and insertion operators ($\mathcal{R}$ and $\mathcal{I}$, respectively), and a maximum number of iterations (max_iter) and allowed time (max_time) that will be used as the stopping criteria. In the first lines of the pseudocode (lines 2–7), all the elements necessary for the proper functioning of the algorithm are initialized. Then, we enter the main loop (lines 8–31), where the candidate solution is initialized as the current solution. As can be seen in lines 9–13, the candidate solution is transformed at each iteration after having removed and reinserted $\tilde{\phi}$ of its customers, according to the selected removal and insertion operators. Then, the intra-route local search previously described is applied to this solution (line 14). As presented in lines 16–22, the current solution will be updated by the candidate solution as long as the latter has a lower cost, and if not, there is a certain probability that it will be updated as well (see the aspiration criterion explained above). Lines 23–26 check whether the current solution can update the best overall solution. After this, removal and insertion matrices are updated, as well as the temperature (lines 27 and 28). Finally, the algorithm terminates when the stopping conditions are fulfilled, and it outputs the best solution found during the search.

Algorithm 3.4 Adaptive large neighborhood search for the MC-TTRP

procedure ALNS_MC-TTRP(`cw_sol`, $\mathcal{R}$, $\mathcal{I}$, max_iter, max_time)
  `best_sol` ← `cw_sol`
  `current_sol` ← `cw_sol`
  $iter \leftarrow 1, init\_time \leftarrow$ system time
  Initialize removal and insertion matrices, $M_{\mathcal{R}}$ and $M_{\mathcal{I}}$
  Initialize temperature, $T \leftarrow T^1$
  Create new $tabu\_list = \{\}$
  while !stopping conditions **do**
    `candidate_sol` ← `current_sol`
    Choose $\tilde{\phi}$ randomly using β(2,5)
    Select $R \in \mathcal{R}$ and $I \in \mathcal{I}$ using $M_{\mathcal{R}}$ and $M_{\mathcal{I}}$, provided they are not stored in $tabu\_list$
    `candidate_sol` ← Remove and reinsert $\tilde{\phi}$ customers from `current_sol`, according to (R, I)
    Replace the operators in $tabu\_list$ with (R, I)
    `candidate_sol` ← Apply intra-route improvement to `candidate_sol`
    $sol\_improved, sol\_worsened, best\_improved \leftarrow 0$
    if cost(`candidate_sol`) < cost(`current_sol`) **then**
      `current_sol` ← `candidate_sol`
      $sol\_improved \leftarrow 1$
    else
      $diff =$ cost(`candidate_sol`) − cost(`current_sol`)
      with probability $\exp((-diff)/T)$, `current_sol` ← `candidate_sol`, $sol\_worsened \leftarrow 1$
    end if
    if cost(`current_sol`) < cost(`best_sol`) **then**
      `best_sol` ← `current_sol`
      $best\_improved \leftarrow 1$
    end if
    Update matrices $M_{\mathcal{R}}$ and $M_{\mathcal{I}}$ using (R, I), $sol\_improved$, $sol\_worsened$, and $best\_improved$
    Update T
    Check stopping conditions using $iter, init\_time, max\_iter, max\_time$
    $iter = iter + 1$
  end while
  return(`best_sol`)
end procedure

3.3.4 Hybrid ALNS with tabu search (ALNS-TS)

Another algorithm that we have implemented for the second phase of our proposal is a hybrid of ALNS and tabu search (TS), which we have called ALNS-TS. This hybrid methodology has the advantage of combining the diversification capability of ALNS with the intensification provided by TS.

The ALNS-TS starts from the feasible solution returned by the constructive heuristic described in Subsection 3.3.1. The ALNS detailed in Algorithm 3.4 is then applied to this initial solution, adding an extra condition at the end of the main loop: every time the best overall solution is improved, or the number of iterations is a multiple of 100, a call is made to the TS presented in Algorithm 3.3. The idea is to intensify the search by exploring a large number of neighborhoods, rather than limiting it to a single one, as is done by the ALNS at

each iteration. The TS has the ability to accept slightly worse solutions or to move through the infeasible region, thus also performing a diversification of the search. Furthermore, a tabu list is used to avoid stagnation.

While it is true that the TS can navigate through the infeasible solution space, it must return a feasible solution, otherwise our ALNS cannot operate. This has motivated us to design the metaheuristic explained in the following subsection.

3.3.5 Penalized ALNS-TS (PANLS-TS)

The fourth of the algorithms implemented for the second phase of our proposal is also ALNS-based. In particular, it is derived from the ALNS-TS introduced in Subsection 3.3.4. An existing issue in MC-TTRPs is that the search often starts from a solution in which there are very few possible moves because the trucks and trailers are heavily loaded. This sometimes leads to stagnation, as the search often returns to the same solutions. For this reason, we have implemented the penalized ALNS-TS (PALNS-TS), a version of the ALNS-TS capable of handling infeasible solutions, intending to increase the number of possible moves in each iteration of the ALNS itself.

In this way, to measure the total cost of each route, r, along the ALNS, we consider the following objective function:

$$\tilde{OF}(r) = \mathrm{dist}(r) + \tilde{\alpha} \cdot \tilde{\mathrm{Ex}}(r),$$

where $\mathrm{dist}(r)$ is the total distance traveled, $\tilde{\alpha}$ a parameter of penalty, and $\tilde{\mathrm{Ex}}(r)$ denotes the number of extra compartments that are being used in the vehicle covering route r. The $\tilde{\alpha}$ penalty is calibrated in the same way as α is in equation (3.1), considering that violating the constraints now means using more compartments than those available: in the case of feasibility, the penalty is decreased by a factor $1 + \tilde{\gamma}$. Otherwise, if the current solution is using more compartments than those available, $\tilde{\alpha}$ is increased by $1 + \tilde{\gamma}$. The parameter $\tilde{\gamma}$ is set to 0.01 at the beginning of the method. Consequently, this update strategy oscillates between feasible and infeasible solutions.

Therefore, at each iteration, the PALNS-TS has to consider whether or not the insertion it chooses produces infeasibility, and this infeasibility will be limited. In our case, the solution cannot have more than four extra compartments. This parameter is certainly configurable, but we decided not to allow a high value for the infeasibility to avoid the search spending too much time exploring the infeasible space. Thus, our proposal will take into account the $\tilde{\alpha} \cdot \tilde{\mathrm{Ex}}(r)$ score when selecting the insertion move (except for the `Random_i`).

3.4 Computational study

To evaluate the efficiency of our proposals, we analyzed the impact of the MC-TTRP model introduced in Chapter 2 and two-phase heuristics using a modified version of a set of well-known benchmarks from the literature and a real-world problem.

The proposed heuristics described in Section 3.3 were implemented in `R` 4.0.2. To validate their performance, a set of experiments were conducted on the Finisterrae II supercomputer, provided by the Supercomputing Center of Galicia[4] (CESGA), which consists of 306 nodes powered by two deca-core Intel Haswell 2680v3 CPUs with 128 GB RAM connected through an Infiniband FDR network. The mathematical model presented in Subsection 2.3.2 was solved using the `Gurobi` 8.1.0 solver. The code was run on a hexa-core Intel i7-8700 CPU with 16 GB RAM.

The following subsections report the computational study. To the best of our knowledge, there are currently no existing MC-TTRP benchmark problems. Hence, before showing the computational results of our algorithms, Subsection 3.4.1 describes the generation of a set of new MC-TTRP test problems based on the 21 well-known TTRP cases introduced by Chao (2002). Subsection 3.4.2 presents the solutions achieved by the four algorithms on our set of instances. The ITS is the one that shows better results for these problems. Therefore, Subsection 3.4.3 will focus on this heuristic. It describes the solutions obtained both in the instances created by Chao for the TTRP as well as in our generated datasets for the MC-TTRP, from a more detailed perspective. Furthermore, the real-world application described in Subsection 2.3.1 was used to validate this heuristic, comparing the quality of the results achieved to those in the case of the exact method.

3.4.1 Set of instances

To the best of our knowledge, there are no benchmark instances in the literature for the MC-TTRP. Therefore, it was necessary to slightly modify the 21 TTRPs[5] developed by Chao (2002). Thus, to test our algorithms, we generated 21 new MC-TTRP instances by adding compartments to Chao's benchmarks. These test problems were derived from seven VRPs created by Christofides et al. (1979), with 50 to 199 customers by specifying 25%, 50%, and 75% of the customers as truck customers. For further details about the nature of these problems, we recommend referring to the original work of Chao (Chao, 2002).

[4] https://www.cesga.es/.

[5] The 21 test problems of the TTRP (Chao, 2002) are available at https://github.com/LauraDavilaPena/MC-TTRP_heuristics/tree/main/instances.

Table 3.3: Dimensions of the MC-TTRP test problems.

Problem number	Customers		Trucks				Trailers			
	VCs	TCs	Number	Compart.	Capacity of compartments	Total capacity	Number	Compart.	Capacity of compartments	Total capacity
1	38	12								
2	25	25	8	20	5	100	5	10	10	100
3	13	37								
4	57	18								
5	38	37	14	20	5	100	8	10	10	100
6	19	56								
7	75	25								
8	50	50	12	30	5	150	6	10	10	100
9	25	75								
10	113	37								
11	75	75	18	30	5	150	9	10	10	100
12	38	112								
13	150	49								
14	100	99	26	30	5	150	14	10	10	100
15	50	149								
16	90	30								
17	60	60	11	30	5	150	6	10	10	100
18	30	90								
19	75	25								
20	50	50	15	30	5	150	8	10	10	100
21	25	75								

Regarding the characteristics of our MC-TTRPs, Table 3.3 gives a general overview of our instances[6], which present variety in terms of the number of customers and vehicles, vehicle capacity, and compartment configuration. The strategy chosen to generate them was as follows: first, the number of available vehicles is computed as $\lceil 1.5 \cdot av \rceil$, where av denotes the corresponding available vehicle in Chao's TTRPs. Also, the original capacity of each truck and trailer was divided into compartments of capacity 5 and 10, respectively. As a result, depending on the instance, trucks could be split into 20 or 30 compartments, while trailers were all split into 10. Furthermore, we assumed that every customer demands two different types of products, so we equally split the total original demand.

Moreover, we classified these instances into three categories depending on their percentage of truck customers:

- G1: instances 1, 4, 7, 10, 13, 16, and 19 belong to group one.
- G2: instances 2, 5, 8, 11, 14, 17, and 20 belong to group two.
- G3: instances 3, 6, 9, 12, 15, 18, and 21 belong to group three.

[6]Note that every three consecutive instances come from the same VRP.

3.4.2 Results on test instances and comparative study

This subsection presents the results obtained by the four two-phase heuristics methods using the 21 MC-TTRP instances described in Subsection 3.4.1. In this way, Tables 3.4–3.7 report the objective values of each heuristic, where the first column represents the instance number. The second column indicates the cost of the initial solution obtained with our constructive algorithm, that is, the cost of the solution from which all metaheuristics start their search. Naturally, this value is the same for all tables. Column 3 shows the cost of the best solution obtained by the considered algorithm over 10 runs executed for one hour each. Column 4 presents the average cost obtained among these 10 runs. Finally, the last column displays the average of the computational times in seconds (within the $1-$hour threshold) required by each heuristic to find the best solution for each run.

As can be seen from the tables, every method obtains solutions that improve on the initial one for all 21 instances. Only the ALNS seems to be stuck at the beginning of the search when solving problem 13, as the best cost and average cost just reduced the initial objective by 0.014% and 0.001%, respectively. Even so, the overall average improvement is 11.55% and 8.75% for the best cost and average cost. More specifically, best and average costs for groups G1, G2, and G3 reduce the initial cost by 8.82%, 11.48%, and 14.36% and 6.14%, 8.57%, and 11.53%, respectively. Let us now see what are the improvement percentages with respect to the solution provided by the constructive heuristic, for each of the algorithms. Regarding the ITS, 12.82% is obtained for the best cost and 10.80% for the average cost. These values are 8.97% and 5.87% for the ALNS. In the case of ALNS-TS, the best solution improves by 12.04% the cost of the CW solution, and this rate is 9.13% for the average cost. Finally, the PALNS-TS achieves reductions of 12.38% and 9.18% in best cost and average cost. Table 3.8 displays the reduction percentages reached by the individual heuristics, distinguishing between the types of instances. We can therefore deduce that all these metaheuristics obtain greater improvements in the initial cost the higher the percentage of TCs in the instances. In addition, it is also observed that ITS is overall the best performer in terms of solution quality, followed by PALNS-TS.

Returning to Tables 3.4–3.7, if we analyze the computational times employed by each of these metaheuristics, we notice some differences: the ITS took the most time, on average, to find the best solutions for each run, with 45.18 min. In the case of ALNS, ALNS-TS, and PALNS-TS, these values are 33.10, 30.28, and 35.73 min. Considering that all four algorithms have been running for one hour, we can infer that the ALNS-based methods stagnate first, as they barely manage to improve the solution in the second half hour.

Table 3.4: Computational results for 21 test MC-TTRPs, using the ITS.

Problem number	CW cost	Best cost	Avg cost	Avg time (s)
1	662.94	624.74	627.98	769.98
2	832.60	702.20	724.42	1896.85
3	885.79	741.43	750.23	1659.00
4	1013.57	895.62	907.95	2366.98
5	1154.20	986.80	1000.30	2506.17
6	1264.09	1089.63	1109.27	2071.00
7	1013.72	928.40	946.50	2843.47
8	1065.40	997.44	1005.27	2895.57
9	1357.50	1059.51	1105.31	2808.01
10	1336.31	1256.32	1271.75	3314.64
11	1422.93	1339.16	1355.02	3156.77
12	1683.38	1394.58	1422.80	3386.75
13	1670.52	1587.07	1615.12	3389.67
14	1858.74	1721.16	1750.71	3154.42
15	2072.08	1773.33	1817.12	3381.31
16	1681.63	1437.49	1491.31	3322.51
17	1891.04	1549.24	1631.11	3372.20
18	1968.90	1522.39	1589.44	3531.53
19	1026.90	876.95	917.27	2139.40
20	1125.28	965.85	997.90	2578.43
21	1115.85	960.92	977.19	2385.56

Table 3.5: Computational results for 21 test MC-TTRPs, using the ALNS.

Problem number	CW cost	Best cost	Avg cost	Avg time (s)
1	662.94	639.22	650.99	668.23
2	832.60	723.54	750.94	506.27
3	885.79	784.09	799.11	548.17
4	1013.57	918.97	958.61	1203.70
5	1154.20	1038.55	1060.22	1559.39
6	1264.09	1144.05	1174.07	2136.47
7	1013.72	961.45	973.18	3112.95
8	1065.40	1009.89	1027.19	2116.68
9	1357.50	1168.74	1223.42	2855.09
10	1336.31	1309.31	1330.27	1067.31
11	1422.93	1386.01	1417.77	747.72
12	1683.38	1527.34	1600.05	3198.37
13	1670.52	1670.29	1670.50	5.19
14	1858.74	1822.69	1853.28	1029.13
15	2072.08	1948.22	2010.56	3398.17
16	1681.63	1494.54	1573.47	3440.85
17	1891.04	1621.25	1705.61	3049.91
18	1968.90	1688.63	1820.30	3324.04
19	1026.90	840.34	901.45	2942.49
20	1125.28	945.71	1020.84	2262.43
21	1115.85	998.51	1022.30	2537.56

Table 3.6: Computational results for 21 test MC-TTRPs, using the ALNS-TS.

Problem number	CW cost	Best cost	Avg cost	Avg time (s)
1	662.94	624.68	644.48	776.66
2	832.60	715.49	741.21	741.71
3	885.79	753.75	772.21	201.93
4	1013.57	940.74	966.60	1767.42
5	1154.20	1033.03	1053.16	1498.84
6	1264.09	1142.71	1170.45	1586.39
7	1013.72	944.51	953.42	1609.53
8	1065.40	976.37	1003.45	1196.54
9	1357.50	1117.00	1136.63	1181.94
10	1336.31	1248.41	1285.30	2232.39
11	1422.93	1340.97	1373.39	2498.85
12	1683.38	1449.75	1487.54	2120.77
13	1670.52	1579.51	1620.97	3321.52
14	1858.74	1725.11	1782.85	2678.46
15	2072.08	1811.95	1842.95	2185.06
16	1681.63	1460.44	1530.49	2695.77
17	1891.04	1524.75	1615.50	2802.91
18	1968.90	1506.49	1615.31	2513.97
19	1026.90	839.93	891.10	1608.39
20	1125.28	911.67	977.63	1386.74
21	1115.85	979.36	1000.07	1546.38

Table 3.7: Computational results for 21 test MC-TTRPs, using the PALNS-TS.

Problem number	CW cost	Best cost	Avg cost	Avg time (s)
1	662.94	616.27	631.97	1912.83
2	832.60	696.13	721.90	903.99
3	885.79	758.92	807.39	521.30
4	1013.57	890.27	931.01	2300.41
5	1154.20	1000.90	1044.57	2057.88
6	1264.09	1138.10	1204.18	515.74
7	1013.72	938.13	952.79	2087.56
8	1065.40	995.67	1005.53	2039.49
9	1357.50	1111.79	1142.23	2680.97
10	1336.31	1266.12	1297.66	2162.86
11	1422.93	1349.57	1382.59	2404.81
12	1683.38	1432.02	1474.87	2417.27
13	1670.52	1626.91	1654.96	1772.33
14	1858.74	1727.98	1771.72	2567.73
15	2072.08	1838.99	1873.33	2941.79
16	1681.63	1427.55	1502.11	3064.10
17	1891.04	1509.29	1611.10	2657.41
18	1968.90	1514.73	1609.76	2339.43
19	1026.90	834.26	890.18	2932.26
20	1125.28	909.41	953.41	2465.44
21	1115.85	992.21	1010.79	2270.36

Table 3.8: Percentage reduction of the best cost and average cost with respect to the initial cost, by the four metaheuristics, for the three categories of instances.

Instance category	ITS		ALNS		ALNS-TS		PALNS-TS	
	Best	Average	Best	Average	Best	Average	Best	Average
G1	9.42%	7.50%	7.06%	4.33%	9.02%	6.05%	9.77%	6.66%
G2	11.87%	9.81%	9.16%	6.05%	12.18%	8.84%	12.69%	9.58%
G3	17.17%	15.09%	10.70%	7.23%	14.91%	12.50%	14.67%	11.30%

To facilitate the comparison of the objectives achieved by these four methods, Tables 3.9 and 3.10 show the best solutions and the average solutions obtained by each of them, respectively. From Table 3.9, we observe that ITS reaches better solutions in 10 of the 21 instances, while PALNS-TS outperforms the other approaches in 7 problems. Table 3.10 indicates that the former algorithm also attains better average solutions in 16 out of 21 instances.

For all the above reasons, we have decided to take the ITS as a reference and perform for this metaheuristic a more detailed study in the following subsection.

Table 3.9: Comparison of the best solutions obtained for the different metaheuristics.

Problem number	ITS	ALNS	ALNS-TS	PALNS-TS
1	624.74	639.22	624.68	**616.27**
2	702.20	723.54	715.49	**696.13**
3	**741.43**	784.09	753.75	758.92
4	895.62	918.97	940.74	**890.27**
5	**986.80**	1038.55	1033.03	1000.90
6	**1089.63**	1144.05	1142.71	1138.10
7	**928.40**	961.45	944.51	938.13
8	997.44	1009.89	**976.37**	995.67
9	**1059.51**	1168.74	1117.00	1111.79
10	1256.32	1309.31	**1248.41**	1266.12
11	**1339.16**	1386.01	1340.97	1349.57
12	**1394.58**	1527.34	1449.75	1432.02
13	1587.07	1670.29	**1579.51**	1626.91
14	**1721.16**	1822.69	1725.11	1727.98
15	**1773.33**	1948.22	1811.95	1838.99
16	1437.49	1494.54	1460.44	**1427.55**
17	1549.24	1621.25	1524.75	**1509.29**
18	1522.39	1688.63	**1506.49**	1514.73
19	876.95	840.34	839.93	**834.26**
20	965.85	945.71	911.67	**909.41**
21	**960.92**	998.51	979.36	992.21

Table 3.10: Comparison of the average solutions obtained for the different metaheuristics.

Problem number	ITS	ALNS	ALNS-TS	PALNS-TS
1	**627.98**	650.99	644.48	631.97
2	724.42	750.94	741.21	**721.90**
3	**750.23**	799.11	772.21	807.39
4	**907.95**	958.61	966.60	931.01
5	**1000.30**	1060.22	1053.16	1044.57
6	**1109.27**	1174.07	1170.45	1204.18
7	**946.50**	973.18	953.42	952.79
8	1005.27	1027.19	**1003.45**	1005.53
9	**1105.31**	1223.42	1136.63	1142.23
10	**1271.75**	1330.27	1285.30	1297.66
11	**1355.02**	1417.77	1373.39	1382.59
12	**1422.80**	1600.05	1487.54	1474.87
13	**1615.12**	1670.50	1620.97	1654.96
14	**1750.71**	1853.28	1782.85	1771.72
15	**1817.12**	2010.56	1842.95	1873.33
16	**1491.31**	1573.47	1530.49	1502.11
17	1631.11	1705.61	1615.50	**1611.10**
18	**1589.44**	1820.30	1615.31	1609.76
19	917.27	901.45	891.10	**890.18**
20	997.90	1020.84	977.63	**953.41**
21	**977.19**	1022.30	1000.07	1010.79

3.4.3 In-depth analysis of the ITS

Results on TTRP and MC-TTRP test instances

The following computational results help to evaluate the performance of the ITS. Having shown in Section 2.4 that the exact solving of a medium-sized problem can be extremely computation-intensive, we highlight that the main focus of this chapter was to obtain good-quality solutions in reasonable computation time. Moreover, to the best of our knowledge, there are no solution methods for the MC-TTRP in the literature other than those presented in this thesis.

Therefore, to obtain an initial validation of the quality of our CW and ITS, we analyzed their performance with the TTRP instances described above in Table 3.3 but without using compartments. Notably, in this manner, it is possible to compare the solutions obtained by our proposed method with those reported by Chao (2002) and Caramia and Guerriero (2010a).

Table 3.11 summarizes the results obtained for both phases of the algorithms, where the first column represents the instance number, and columns 2, 3 and 4 show the objective value returned by the constructive phase of the three approaches considered, i.e., the distance cov-

Table 3.11: Computational results for 21 test TTRPs.

Problem number	Constructive phase			Improvement phase				Number of vehicles	
	Our CW	Chao	Caramia	Our ITS	Chao	Caramia	BKS	Trucks	Trailers
1	**632.38**	646.02	645.72	**565.01**	565.02	566.80	564.68	5	3
2	711.93	739.90	699.68	635.28	658.07	620.15	612.75	5	3
3	805.34	774.78	770.19	634.16	648.74	632.48	618.04	5	3
4	1002.58	943.47	902.89	809.29	856.20	803.32	798.53	9	5
5	1067.48	1130.85	1035.89	844.46	949.98	842.50	839.62	9	5
6	1193.98	1236.69	1171.50	981.08	1053.23	938.18	933.26	9	5
7	**899.74**	906.31	902.18	852.93	832.26	832.56	830.48	8	4
8	1028.96	971.60	970.45	887.26	900.54	878.87	878.36	8	4
9	1138.93	1106.66	1082.99	**958.50**	971.62	980.42	934.47	8	4
10	1213.15	1159.78	1151.23	1079.73	1073.50	1060.41	1039.07	12	6
11	**1276.23**	1288.74	1288.74	**1123.43**	1170.17	1170.70	1094.11	12	6
12	**1408.93**	1453.82	1440.12	1224.71	1217.01	1178.34	1155.13	12	6
13	1530.66	1481.40	1480.46	1344.82	1364.50	1288.46	1287.18	17	9
14	1617.71	1624.96	1612.34	1444.89	1464.20	1372.52	1353.08	17	9
15	**1706.19**	1858.87	1752.94	1499.29	1540.25	1470.21	1457.61	17	9
16	1349.67	1267.87	1060.26	1015.26	1041.36	1004.69	1002.49	7	4
17	1376.31	1261.17	1120.34	1073.77	1090.46	1042.35	1042.35	7	4
18	1439.34	1366.21	1220.25	1205.46	1141.36	1129.16	1129.16	7	4
19	1141.59	969.96	880.22	852.01	854.02	813.50	813.50	10	5
20	1097.47	1140.47	961.25	908.87	942.39	848.93	848.93	10	5
21	1085.61	1174.43	1009.68	930.70	926.47	909.06	909.06	10	5

ered by all routes once an initial feasible solution is achieved. Note that we have considered the objectives reported by Chao when a descent improvement subroutine is implemented. Concerning the solutions obtained by the constructive stage, our savings-based heuristic allowed the creation of feasible routes, improving on the objectives reported by Chao and Caramia and Guerriero in some cases, and outperforming both of them in five problems.

Furthermore, the next four columns indicate the objective achieved in the improvement phase, where BKS denotes the best known solution values according to Caramia and Guerriero (2010a). For Chao and Caramia and Guerriero's algorithms, values represent their best obtained solutions, with no further details given. In our case, we carried out 500 iterations of the ITS and report the best value over 10 runs. Finally, the last two columns indicate the corresponding available numbers of trucks and trailers.

With respect to computation times, our algorithm requires from 7 to 200 min to achieve the best solutions reported in Table 3.11. Note that our heuristic is efficient to solve the TTRP, but it is actually designed to solve the MC-TTRP. We present the above comparison with existing TTRP solutions in an indicative way to show that our algorithm performs well

Table 3.12: Computational results for 21 test MC-TTRPs.

Problem number	CW objective	ITS objective	No. of used vehicles		Number of routes			Occupancy rate			Time (min)
			Trucks	Trailers	PTR	PVR	MVR	Trucks	Trailers	Total	
1	662.94	624.74	8	4	4	3	1	0.80	1	0.84	6.00
2	832.60	702.20	8	4	4	0	4	0.86	0.90	0.86	7.79
3	885.79	741.43	8	3	5	0	3	0.98	0.80	0.95	23.23
4	1013.57	895.62	13	8	5	5	3	0.79	0.98	0.84	52.71
5	1154.20	986.80	14	7	7	3	4	0.85	0.83	0.84	25.09
6	1264.09	1089.63	14	5	9	0	5	0.98	0.72	0.94	18.93
7	1013.72	928.40	11	6	5	6	0	0.89	0.87	0.89	58.71
8	1065.40	997.44	12	4	8	4	0	0.85	1	0.87	54.46
9	1357.50	1059.51	11	4	7	0	4	0.96	0.90	0.96	46.89
10	1336.31	1256.32	18	9	9	5	4	0.81	0.84	0.81	55.42
11	1422.93	1339.16	18	7	11	4	3	0.91	0.74	0.89	41.76
12	1683.38	1394.58	18	4	14	0	4	0.96	0.80	0.95	56.74
13	1670.52	1587.07	23	13	10	7	6	0.87	0.94	0.88	60.02
14	1858.74	1721.16	26	12	14	7	5	0.85	0.68	0.83	58.67
15	2072.08	1773.33	25	10	15	0	10	0.93	0.68	0.90	56.45
16	1681.63	1437.49	11	6	5	5	1	0.90	0.87	0.90	59.98
17	1891.04	1549.24	11	5	6	0	5	0.91	1	0.92	56.53
18	1968.90	1522.39	11	4	7	0	4	0.95	1	0.96	58.90
19	1026.90	876.95	12	8	4	4	4	0.74	0.80	0.75	47.02
20	1125.28	965.85	13	4	9	0	4	0.76	1	0.78	29.98
21	1115.85	960.92	12	4	8	0	4	0.86	0.80	0.85	22.32

on problems that may be similar.

Returning to the specific problem of interest, Table 3.12 shows the performance of the proposed heuristic, on the 21 instances of the MC-TTRP described in Table 3.3, considering the compartments properly. The first columns indicate the problem number and the objective of the constructive phase. The remaining columns refer to the ITS solution, which corresponds to the best solution found for this method in the computational study of Subsection 3.4.2. Columns 3 to 8 show the objective of the improvement phase (after running the ITS for 1 hour, reporting the best value over 10 runs), vehicles used, and different routes created. The next columns list the vehicle occupancy rate. Column 9 shows the number of truck compartments used divided by the total number of truck compartments in the solution. For instance, the total number of truck compartments in problem 1 was 160 (eight trucks used with 20 compartments each). Column 10 is analogous, considering trailers instead of trucks. The "Total" column is the quotient of the sum of used truck and trailer compartments divided by the total number of compartments in the solution. That is, for problem 1, the total number is 200 (160 truck plus 40 trailer compartments). Finally, the last column refers to the time (in minutes) in which the best reported solution was reached.

To complement the analysis presented in Table 3.12, it should be noted that the occupancy rates are relatively high, with averages of 88%, 86%, and 88% for trucks, trailers, and total, respectively. To provide a better interpretation of the results, we calculated the average rates for groups G1, G2, and G3. The values obtained for trucks were 83%, 86%, and 95%, respectively; thus, we can see that there is a growth in the truck occupancy rate as the number of truck customers increases. The opposite applies in the case of trailers: the rates are 90%, 88%, and 81% for groups G1, G2, and G3, respectively. These results are promising because feasible solutions were achieved at low computational cost.

Application to real-world data

To evaluate the quality of our heuristic algorithm, we considered the data of the real-world example described in Subsection 2.3.1. We solved a set of different-sized problems by combining customers of the agricultural cooperative: P1, P2, and P3 have four customers of each type; P4 and P5 have nine customers, five of which are VCs; and P6, P7, and P8 involve 10 customers, five VCs and five TCs. All of these instances imply two trucks with 13 hoppers that can contain up to 1.5 tons each and one trailer with 15 hoppers with a maximum capacity of 2 tons. Given this information, Table 3.13 gives a comparison between the solutions of both methods. Furthermore, Table 3.14 provides the lower and upper bounds and relative gaps for the exact formulations of those 8 problems by setting a time limit of three hours. It can be seen that problems with 10 customers present a positive gap, which in the particular case of P8 is relatively large.

Table 3.13 shows that our algorithm achieved the exact objective for problems P1, P3, P4, and P5, although the loading procedures were not always the same. Our heuristic did not achieve the exact optimal values for the remaining instances, but there was an increase of at most 2.90% in the objective (which occurred for P8). However, the execution times of the heuristic were significantly shorter (by several orders of magnitude) than those of the exact method. Although optimality convergence was not achieved for all problems, these results are promising because a feasible solution (close to the optimal one) was obtained at a much smaller computational cost (in less than 10 s). For a fair comparison of the times reported in this table, both the model and the heuristic have been run on the Intel i7 mentioned above.

Moreover, the number of vehicles and types of routes were the same for both methods in every problem, except for P5; in this case, our approach served all VCs in an MVR, while TCs were served by a PTR. However, the exact solution allocated one of the TCs to a subtour with customer 3 as the trailer parking place. Figure 3.10 illustrates the results for this instance.

Table 3.13: Results obtained by the exact model versus results obtained with our heuristic algorithm (after 100 iterations of the ITS).

Problem number	Size	Method	Objective	Occupancy rate			Time (s)
				Trucks	Trailers	Total	
P1	8	Exact	189	0.92	0.80	0.88	309.27
		Heuristic	189	0.88	0.87	0.88	5.03
P2	8	Exact	140	0.81	0.73	0.78	6733.20
		Heuristic	142	0.81	0.73	0.78	3.88
P3	8	Exact	106	0.77	0.67	0.73	1275.75
		Heuristic	106	0.77	0.67	0.73	3.90
P4	9	Exact	256	0.85	0.80	0.83	16,766.70
		Heuristic	256	0.81	0.87	0.83	5.58
P5	9	Exact	109	0.81	0.80	0.80	17,511.00
		Heuristic	109	0.62	1	0.76	4.72
P6	10	Exact	222	0.96	0.67	0.85	221,019.00
		Heuristic	223	0.92	0.67	0.83	6.48
P7	10	Exact	237	0.92	0.80	0.88	671,316.00
		Heuristic	238	0.88	0.87	0.88	7.97
P8	10	Exact	207	0.96	0.67	0.85	112,389.00
		Heuristic	213	0.69	1	0.80	6.09

Table 3.14: Lower and upper bounds and gaps for the exact formulation.

Problem number	Lower bound	Upper bound	Relative gap
P1	189	189	0
P2	140	140	0
P3	106	106	0
P4	256	256	0
P5	109	109	0
P6	198	222	0.11
P7	185	237	0.22
P8	73	207	0.65

As another illustrative example of the difference between the exact solution and the heuristic, we show the route configuration for P8 in Figure 3.11. The customers of the sub-tour presented in the optimal solution, 2–9–7–8–2, are part of a PTR in our algorithm's result. Customer 5 was selected the trailer parking place for the sub-tour of the heuristic solution, 5–10–6–5.

Table 3.13 also indicates that, as opposed to the exact result, the heuristic approach gives preference to the loading of trailers, having an average occupancy rate of 86%, while the trucks are 78% full on average. The total occupancy rate does not differ significantly between

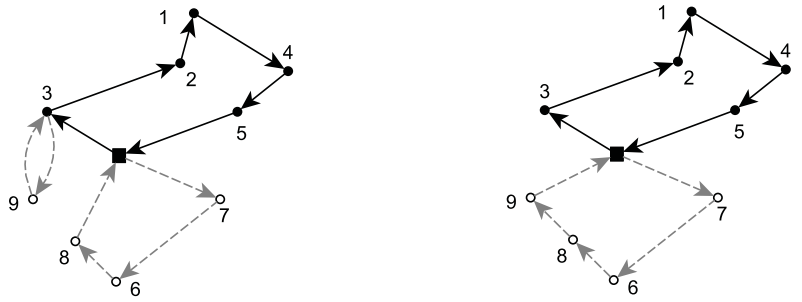


Figure 3.10: Solutions obtained by Gurobi (left) and the two-phase heuristic (right) for problem P5.

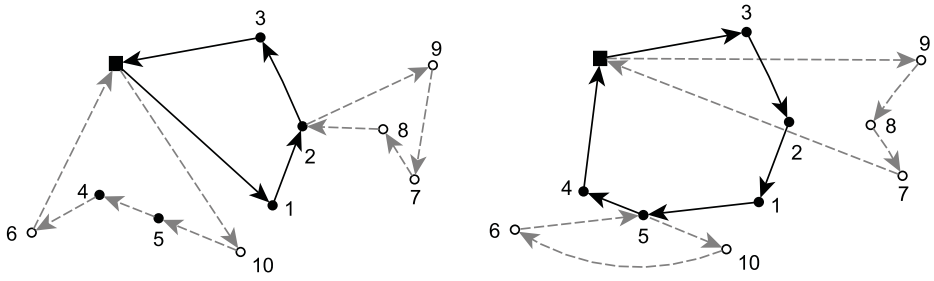


Figure 3.11: Solutions obtained by Gurobi (left) and the two-phase heuristic (right) for problem P8.

these two methods.

3.5 Conclusions and further research

This part of the thesis considered a routing problem for trucks and trailers divided into compartments, called the MC-TTRP. In Chapter 2, we proposed a novel mathematical formulation of this model as an MILP problem and presented a detailed explanation of its constraints. A real-world example of a Spanish agricultural cooperative that distributes feed to its customers was considered to verify the proposed model. Given that exactly solving the problem for large-sized instances is computationally expensive, the use of operational research approximate techniques is necessary to obtain solutions. Therefore, in the present chapter, we introduced and implemented four heuristic algorithms to reduce computation time. Our proposals are two-stage approaches: the first phase iteratively builds an initial solution, based on the savings method of Clarke and Wright, and then the second phase aims to refine the solution. To achieve this, metaheuristics based on tabu search and large neighborhood search were designed: the iterated tabu search (ITS), the adaptive large neighborhood search (ALNS), the hybrid ALNS with tabu search (ALNS-TS), and the penalized ALNS-TS (PALNS-TS). We

conducted a thorough computational study on different datasets. First, we suitably adapted the 21 TTRP benchmark problems of Chao (2002) to consider compartments, leading to 21 challenging test instances. Our heuristics always generated feasible solutions to the test problems, and the objectives obtained showed that our methods can effectively and efficiently solve the MC-TTRP. A comparative study of the four proposed metaheuristics was carried out and it was found that ITS produced better results. Accordingly, an in-depth analysis of the ITS was performed. First, we tailored our method to the TTRP. Chao's benchmark problems were examined and the solutions reported by both Chao (2002) and Caramia and Guerriero (2010a) were compared to ours. We also conducted a more detailed research on the generated instances. Furthermore, this algorithm was applied to the previously mentioned real-world case of a cooperative. A comparison between our results and those provided by the exact formulation showed a significant decrease in computational cost, achieving good-quality solutions to the problems. Considering this, we believe that the proposed heuristic is a promising solution approach for the MC-TTRP.

There is still much room for further research on this variant of the VRP. It would be interesting to apply the model and solution algorithms to other feed producing companies similar to that considered in this study as well as other businesses, such as milk collection and fuel distribution companies. Another open research direction is to extend the model to include modifications, such as stochastic demands, time windows, or heterogeneous vehicle fleets, to address other similar real-world problems. Moreover, in view of the different ingredients that make up our model, including route design, assignment of customers to vehicles, and loading of compartments, exploring the performance of decomposition techniques for solving it could provide good results.

In addition, future work could be to investigate more thoroughly the four proposed algorithms from a comparative point of view. For the 21 generated instances, the ITS was found to be preferable to the other methods. Nevertheless, these instances may not cover the whole instance space. Thus, one possible line of research could be to apply the Instance Space Analysis (ISA, Smith-Miles et al., 2014) methodology, creating new test problems to fill empty regions in the instance space and see whether ITS is actually superior over the entire area or depends on the specific features of each problem. For this purpose, the Melbourne Algorithm Test Instance Library with Data Analytics (MATILDA) tool (Smith-Miles et al., 2019) can be of great support.

The code and instances required to reproduce the results reported herein are available at https://github.com/LauraDavilaPena/MC-TTRP_heuristics.

Part II

On game theory in classification problems

Introduction to game theory and classification problems

Part II of this dissertation deals with the evaluation of the influence that certain features have on a classification problem. This part is organized in a single chapter that proposes an influence measure based on game theory ideas.

There are many situations in which we need to anticipate a certain output by knowing specific inputs. These types of problems arise in a wide range of fields such as science, economics, or health care. For instance, we may be interested in predicting the profits of a certain company, or in knowing whether or not a given patient suffers from a disease. The latter case, in which the response is categorical, is framed within classification problems.

A *classification problem* consists of predicting the value of a qualitative response variable for one or more individuals, making use of the values we know of certain variables (features) of such individuals. Those predictions are based on the knowledge obtained through a training sample of individuals whose values of the features and of the response variable are known. Classification problems have been extensively studied in the field of statistics, but can also be addressed by using machine learning techniques. Numerous classifiers have been proposed and analyzed in the machine learning literature (see, for example, Fernández-Delgado et al., 2014).

An important aspect when dealing with classification problems involves identifying the most decisive features for accurately predicting the response. Several methodologies have been adopted to rank these explanatory variables (see Štrumbelj et al., 2009), most of which are model-dependent, i.e., they rely on the predictor that has been used. Nevertheless, explanation methods that are independent of the classifier in question have been also developed. In particular, Štrumbelj and Kononenko (2010) introduced one of those methods using the Shapley value of cooperative games (Shapley, 1953).

A *cooperative game* is a pair (N, v), where N is the finite set of players, and $v\colon 2^N \to \mathbb{R}$ is the characteristic function of the game, which satisfies $v(\varnothing) = 0$. We usually interpret $v(S)$ as the gain that coalition $S \subseteq N$ can obtain. Also, $G(N)$ represents the set of all

cooperative games with set of players N. In general, we identify (N, v) with its characteristic function, v.

An extensively addressed problem in cooperative games is how to allocate $v(N)$ among the cooperating agents. One of the most important allocation rules is the Shapley value, $\Phi\colon G(N) \to \mathbb{R}^N$, which represents a fair compromise for the players and it is defined by the following expression:

$$\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!}\,(v(S \cup \{i\}) - v(S)),$$

for all $v \in G(N)$ and $i \in N$. The Shapley value has many applications in a wide range of fields. Just to give a few instances, Liu et al. (2020) uses the Shapley value for water resource allocation in multinational river basins, Saavedra-Nieves and Saavedra-Nieves (2020) proposes a new quota system for the milk market that is based again on the Shapley value, and Li and Chen (2020) makes use of the Shapley value in their study of alliance formation in an assembly system where several upstream complementary suppliers produce components and sell them to a downstream manufacturer. Algaba et al. (2019b) is a recent review of the Shapley value, its variants, and its applications.

Different axiomatic characterizations of the Shapley value have been proposed through some of its properties. We will not go into detail in the present thesis, but the reader is referred to the work of González-Díaz et al. (2010) for further study of cooperative games and, more specifically, of this value.

In Chapter 4 we will propose an influence measure of the different features in a classification problem. To do so, the Shapley value will be used. We provide properties that satisfies such a measure and characterize it axiomatically. In addition, a computational study is conducted and an application of the measure to a COVID-19 database is presented.

Chapter 4

An influence measure for classification problems

This chapter deals with an important subject in classification problems addressed by machine learning techniques: the evaluation of the influence of each of the features on the classification of individuals. Specifically, a measure of that influence is introduced using the Shapley value of cooperative games. In addition, an axiomatic characterization of the proposed measure is provided based on properties of efficiency and balanced contributions. Furthermore, some experiments have been designed in order to validate the appropriate performance of such measure. Finally, the methodology introduced is applied to a sample of COVID-19 patients to study the influence of certain demographic or risk factors on various events of interest related to the evolution of the disease.

The contents of this chapter are collected in Davila-Pena et al. (2022b).

Contents

4.1 Introduction

One of the fundamental elements when dealing with classification problems is to be provided with a training sample, that is, a set of observations for which we know how they have been classified and the value of all the features. In this chapter, we make use of some classification techniques imported from machine learning to develop a methodological tool for the

exploratory analysis of a training sample of the type described above. Specifically, our objective is to define a sensible measure to evaluate the influence of the features on the value of the response variable.

The relevance of our objective can be illustrated with a real problem of applied research that we recently faced. During the first wave of COVID-19 in Spain, we had access to a database of 10,454 patients from Galicia (a region in the northwest of Spain) infected with COVID-19 from March 6, 2020 to May 7, 2020. Knowing the characteristics of individuals that significantly increase their probability of needing access to certain health infrastructures is highly useful for health authorities to make the right decisions. Therefore, we set out to use these data to find out which were the values of the features that most influenced the worsening of an infected patient's condition, so that such patient had to be hospitalized, had to be admitted to the intensive care unit (ICU) or even died. In Section 4.4 we apply the methodology we introduced and analyzed in Sections 4.2 and 4.3 to explore this database.

The machine learning literature has been more focused on classification than on analyzing the influence of features, although the latter is a problem of growing interest (see, for example, Section 4 of Carrizosa et al., 2021). In the emerging literature on this topic (cf. Burkart and Huber, 2021), we have identified the following shortcoming: we have not found a global measure of the influence of features on the response variable in a classification problem that is both theoretically well-founded and empirically contrasted. We thus fill the gap by introducing and analyzing both theoretically and empirically an influence measure based on the Shapley value of cooperative games (Shapley, 1953). In the following paragraphs, we present an overview of the current research background by discussing some of the works that we consider to be closest to our study.

When analyzing a database, it is very likely to come across features that only cause noise and are not really influential for the response. For this reason, the selection of features is a prior problem to the study of influences we discuss here since it is convenient to start from an already selected set of features and then comparatively study their influences. The problem of feature selection has given rise to a remarkable literature. For instance, Ghaddar and Naoum-Sawaya (2018) introduces an iterative approach to address feature selection in classification using support vector machines and applies it to a case of medical tumor diagnosis. Jothi et al. (2021) attempts to identify individuals who may be suffering from mental illness by implementing the Shapley value as a feature selection of a data mining classifier system.

In the context of classification, Štrumbelj and Kononenko (2010) introduces a general procedure to assess the importance that the various features have had in the classification of a particular individual. Our approach is closely related to that paper but it is essentially differ-

ent mainly because it is not locally oriented: we do not attempt to evaluate the influence of each feature on the classification of a particular individual, but rather to evaluate the influence of each feature value on the response variable. The Štrumbelj and Kononenko's procedure has been extended to regression models in Štrumbelj and Kononenko (2011). Although the evaluation of features' contributions to model predictions is of great relevance, other papers have been devoted to determining the importance of features with respect to model performance. According to the latter approach, Casalicchio et al. (2019) describes a procedure, based on the Shapley value, that allows to fairly distribute the overall performance of a model among the features involved. Regarding our contribution, we propose a global measure in the context of classification and analyze it in some depth, both from a theoretical and empirical point of view. Furthermore, instead of focusing on how features affect the performance of a model trained on a dataset, our method is aimed at measuring the influence of the features on whether the response variable takes a certain value in a classification problem; to this end, it considers the knowledge provided by a dataset.

Probably the nearest paper to the subject of our research is Datta et al. (2015). In that paper, the authors also study how influential the various features are in a classification problem. They theoretically base their measure of influence in the binary case, that is, when both the features and the response variable take only two possible values. However, their measure of influence can also be used in the general non-binary case. Another difference with our approach is that they start from a set of observed cases of the feature vectors and an already fixed classifier, and study the influence of each feature for that classifier. In our approach we start from a training sample of individuals for whom we have observed their values of the features and of the response variable; we intend to know the influence of the feature values on the response in the population from which the training sample has been drawn. It is certainly possible to use the approach of Datta et al. (2015) to address our problem: train a classifier with the training sample, and then apply Datta et al.'s measure of influence. In fact, in Section 4.3 we compared the latter approach with our own and show that overall our procedure presents a better performance. In particular, it is seen that our measure, unlike that of Datta et al. (2015), distinguishes between positive and negative influences, thus improving the interpretability of the results. We refer the reader to Section 4.3 for a further discussion on this topic.

A common point of Štrumbelj and Kononenko (2010), Datta et al. (2015), and our work is that all three make extensive use of cooperative game theory tools, especially the Shapley value. The Shapley value is a rule for distributing the profits generated by a collection of cooperating agents and it has many applications in a wide range of fields, as we have shown

previously. The objective of fairly distributing the benefits generated by the cooperation of a set of agents is essentially the same as fairly evaluating the contribution of the variables involved in a classification problem. In the field of game theory, the Shapley value has proven to be an exceptionally valuable tool for distributing such benefits. On the one hand, the Shapley value has been defined to always provide fair allocations. Furthermore, it usually has specific properties in the particular problems to which it is applied, which are often interpretable in a very insightful way. On the other hand, the Shapley value has an explicit formula that allows it to be calculated or approximated in a computationally affordable way, and it has been widely studied. In our particular case, the fact that the Shapley value depends on all possible subgroups of features means that somehow it accounts for all potential interactions between features when calculating our influence measure. In this way, this tool allows us to better understand the importance that the several features have on a classification problem. Thus, the last few years there has been an explosion of papers using the Shapley value to improve the interpretability of complex machine learning models.

A particularly relevant reference in this latter context is Lundberg and Lee (2017), which introduces SHAP (SHapley Additive exPlanations), a unified framework for interpreting predictions and providing theoretical foundation in several prediction models. This methodology is used, for example, in Smith and Alvarez (2021), which analyzes a COVID-19 database collected at the early stages of the pandemic in Wuhan (China). The research conducted in our work does not fit within this framework because SHAP focuses on determining the importance of features for each particular prediction. As already mentioned, we instead study the features' importance not for each specific individual but for the whole dataset.

To summarize, the contribution of our paper is framed within a topic of great importance in machine learning today, which is the explainability and interpretability of models. Specifically, we propose a model-agnostic measure of the influence of the features involved in a classification problem. To do so, we make use of the Shapley value, just as it is increasingly done in the artificial intelligence literature. For our influence measure, we first provide a theoretical justification based on the properties that characterize it, and we then show its good performance with a computational study. Finally, we illustrate the versatility of our measure through the analysis of a COVID-19 database in Galicia (Spain).

The organization of this chapter is as follows. Section 4.2 presents the influence measure and discusses its theoretical basis, including an axiomatic characterization. In Section 4.3, various experiments are carried out to validate in practice the behavior of our measure, which is also compared with another approach from the literature. Section 4.4 uses the measure to explore data from a sample of COVID-19 patients to detect features that affect mortality, ICU

admission, and patient hospitalization, and to evaluate the influence of such features. Finally, Section 4.5 summarizes the main conclusions of this work.

4.2 Axiomatic characterization

We start this section by formally establishing what we mean by *classification problem*. In one such problem we have a vector of features $X = (X_1, \ldots, X_k)$ and a response variable Y. $K = \{1, \ldots, k\}$ denotes the set of indices of the features. Each feature X_j, $j \in K$, takes values in a finite set $\mathcal{A}_j$ and Y takes values in a finite set $\mathcal{B}$. We also have a training sample $\mathcal{M} = \{(X^i, Y^i)\}_{i=1}^n$, where $X^i = (X_1^i, \ldots, X_k^i)$ and Y^i are the observed values of the features and the response variable corresponding to individual i. A classification problem is thus characterized by a triplet $(X, Y, \mathcal{M})$.

A *classifier* trained with sample $\mathcal{M}$ is a map $f^{\mathcal{M}}$ that assigns to every $a \in \mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ (an observation of X) a probability distribution over $\mathcal{B}$, i.e., $f^{\mathcal{M}}(a) = (f_b^{\mathcal{M}}(a))_{b \in \mathcal{B}}$ with $f_b^{\mathcal{M}}(a) \geqslant 0$, for all $b \in \mathcal{B}$, and $\sum_{b \in \mathcal{B}} f_b^{\mathcal{M}}(a) = 1$. Each $f_b^{\mathcal{M}}(a)$ is the estimated probability that an individual whose observed values of the features are given by a belongs to group b of the response variable Y. From now on, $\mathcal{A}_V$, a_V, X_V, and X_V^i will denote the restrictions of $\mathcal{A}$, a, X, and X^i to the variables of V, respectively (for all $V \subseteq K$).

Our goal in this section is to use classification techniques to define a measure that allows us to study the influence of the features on the response variable. The formal definition of an influence measure is the one included below.

Definition 4.1. An **influence measure** for $(X, Y, \mathcal{M})$ is a map I that assigns to every $a_R \in \mathcal{A}_R$ ($R \subseteq K$), $b \in \mathcal{B}$, and $T \subseteq K$ ($T \neq \varnothing$) a vector $I(a_R, b, T) = (I_l(a_R, b, T))_{l \in T} \in \mathbb{R}^T$. The vector $I(a_R, b, T)$ provides an evaluation of the influence that each feature X_l ($l \in T$) has on whether the response is worth b when X_R is worth a_R and we only take into account the features $\{X_l\}_{l \in T}$.

Remark 4.1. Note that R represents the subset of indices corresponding to those features whose values are to be fixed and, therefore, we will only take into account those observations that present these values. On the other hand, T is the non-empty subset of K corresponding to those features whose influences are to be studied. The appropriate selection of these subsets allows this influence measure to be used with considerable versatility, as illustrated in Section 4.4.

Throughout this section, we present an influence measure based on the Shapley value of cooperative games. To strengthen the theoretical support for our measure, we introduce it

axiomatically, i.e., we provide a collection of desirable properties and then show that there exists a unique measure that satisfies them. As we show below, two properties are sufficient to characterize our measure.

Regarding the problem that concerns us, our objective is to know how much certain values of some variables contribute to the prediction of a given set of individuals. In particular, for a single individual, $a = (a_1, \ldots, a_k)$, we consider the difference between the average prediction of our classifier when only the feature values corresponding to T are known, and the average prediction when no feature value is known. Thus, we take:

$$\triangle_{f_b^{\mathcal{M}}}(a, T) = \frac{1}{|\mathcal{A}_{K\setminus T}|} \sum_{a'_{K\setminus T} \in \mathcal{A}_{K\setminus T}} f_b^{\mathcal{M}}(a_T, a'_{K\setminus T}) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} f_b^{\mathcal{M}}(a'),$$

where $(a_T, a'_{K\setminus T})$ denotes the k-dimensional feature vector whose j-th component is a_j if $j \in T$ or a'_j if $j \in K\setminus T$.

The first desirable property we consider establishes that a measure of influence distributes among the features corresponding to T their total influence on the response variable (when X_R equals a_R). One way to estimate that total influence using the classifier $f^{\mathcal{M}}$ is given by the following expression:

$$\frac{1}{n^b_{a_R}} \sum_{(X^i, Y^i) \in \mathcal{M}^b_{a_R}} \triangle_{f_b^{\mathcal{M}}}(X^i, T), \tag{4.1}$$

where $\mathcal{M}^b_{a_R}$ denotes the subsample of $\mathcal{M}$ formed by the observations (X^i, Y^i) with $X^i_R = a_R$ and $Y^i = b$, and $n^b_{a_R}$ denotes the size of the subsample $\mathcal{M}^b_{a_R}$.

Notice that expression (4.1) can be interpreted as an estimation of the variability of the response variable due to the T features (using $f^{\mathcal{M}}$). Therefore, the first property we ask for an influence measure is the $f^{\mathcal{M}}$-Efficiency below.

***$f^{\mathcal{M}}$*-Efficiency.** An influence measure I satisfies $f^{\mathcal{M}}$-Efficiency if, for every $(X, Y, \mathcal{M})$, every $a_R \in \mathcal{A}_R$ ($R \subseteq K$), $b \in \mathcal{B}$, and $T \subseteq K$ ($T \neq \varnothing$), it holds that $\sum_{l \in T} I_l(a_R, b, T)$ is equal to the amount in expression (4.1).

The second property that we consider is a fairness property that treats all features in a balanced way. Informally, it states that given two of these features, the effect of ignoring one to the measure of the influence of the other is identical for both features. Note that the marginal loss or gain of influence that the inclusion or exclusion of one feature causes to another feature is due to the dependency that exists between the two. The fact that the dependence between features is symmetrical makes advisable the property of balanced contributions.

Balanced Contributions. An influence measure satisfies Balanced Contributions if, for every

$(X, Y, \mathcal{M})$, every $a_R \in \mathcal{A}_R$ $(R \subseteq K)$, $b \in \mathcal{B}$, $T \subseteq K$ $(T \neq \varnothing)$, and $l, m \in T$ with $l \neq m$,

$$I_l(a_R, b, T) - I_l(a_R, b, T \backslash \{m\}) = I_m(a_R, b, T) - I_m(a_R, b, T \backslash \{l\}).$$

Now we state and prove the main mathematical result of this section. It provides a characterization and a formal expression of an influence measure that satisfies all the properties introduced above.

Theorem 4.2. *There exists a unique influence measure for $(X, Y, \mathcal{M})$ that satisfies the properties of $f^{\mathcal{M}}$-Efficiency and Balanced Contributions. For all $a_R \in \mathcal{A}_R$ $(R \subseteq K)$, $b \in \mathcal{B}$, $T \subseteq K$ $(T \neq \varnothing)$ and $l \in T$, this measure (that we denote by I^{Φ}) is given by*

$$I_l^{\Phi}(a_R, b, T) = \frac{1}{n_{a_R}^b} \sum_{(X^i, Y^i) \in \mathcal{M}_{a_R}^b} \Phi_l(v_{X^i}^b|_T), \tag{4.2}$$

where Φ denotes the Shapley value, $v_{X^i}^b$ denotes the game with set of players K given by

$$v_{X^i}^b(S) = \frac{1}{|\mathcal{A}_{K \backslash S}|} \sum_{a'_{K \backslash S} \in \mathcal{A}_{K \backslash S}} f_b^{\mathcal{M}}(X_S^i, a'_{K \backslash S}) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} f_b^{\mathcal{M}}(a'), \tag{4.3}$$

for all $S \subseteq K$, and $v_{X^i}^b|_T$ denotes the restriction of the game $v_{X^i}^b$ to the subsets of T.[1]

Proof. Existence. To show that I^{Φ} satisfies $f^{\mathcal{M}}$-Efficiency, take $a_R \in \mathcal{A}_R$ $(R \subseteq K)$, $b \in \mathcal{B}$, and $T \subseteq K$ $(T \neq \varnothing)$. Shapley (1953) proves that the Shapley value of cooperative games satisfies an efficiency property. In our case, this property implies that

$$\sum_{l \in T} \Phi_l(v_{X^i}^b|_T) = v_{X^i}^b(T).$$

Applying this result we obtain that:

$$\begin{aligned} &\sum_{l \in T} I_l^{\Phi}(a_R, b, T) \\ &= \sum_{l \in T} \frac{1}{n_{a_R}^b} \sum_{(X^i, Y^i) \in \mathcal{M}_{a_R}^b} \Phi_l(v_{X^i}^b|_T) \\ &= \frac{1}{n_{a_R}^b} \sum_{(X^i, Y^i) \in \mathcal{M}_{a_R}^b} \sum_{l \in T} \Phi_l(v_{X^i}^b|_T) \end{aligned}$$

[1] The game in (4.3) results to be the same as the one used in Štrumbelj and Kononenko (2010) to assess the importance of the various features in the classification of a particular individual in a classification problem.

$$
\begin{aligned}
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} v^b_{X^i}(T) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \left(\frac{1}{|\mathcal{A}_{K\setminus T}|} \sum_{a'_{K\setminus T}\in\mathcal{A}_{K\setminus T}} f^{\mathcal{M}}_b(X^i_T, a'_{K\setminus T}) - \frac{1}{|\mathcal{A}|} \sum_{a'\in\mathcal{A}} f^{\mathcal{M}}_b(a') \right) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \triangle_{f^{\mathcal{M}}_b}(X^i, T),
\end{aligned}
$$

which equals expression (4.1).

To show that I^Φ satisfies Balanced Contributions, let $a_R \in \mathcal{A}_R$ ($R \subseteq K$), $b \in \mathcal{B}$, $T \subseteq K$ ($T \neq \varnothing$), and $l, m \in T$ with $l \neq m$. Myerson (1980) proves that the Shapley value of cooperative games satisfies a property of balanced contributions. In our case, this property implies that

$$\Phi_l(v^b_{X^i}|_T) - \Phi_l(v^b_{X^i}|_{T\setminus\{m\}}) = \Phi_m(v^b_{X^i}|_T) - \Phi_m(v^b_{X^i}|_{T\setminus\{l\}}).$$

Applying this result we obtain that:

$$
\begin{aligned}
& I^\Phi_l(a_R, b, T) - I^\Phi_l(a_R, b, T\setminus\{m\}) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \Phi_l(v^b_{X^i}|_T) - \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \Phi_l(v^b_{X^i}|_{T\setminus\{m\}}) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \left(\Phi_l(v^b_{X^i}|_T) - \Phi_l(v^b_{X^i}|_{T\setminus\{m\}})\right) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \left(\Phi_m(v^b_{X^i}|_T) - \Phi_m(v^b_{X^i}|_{T\setminus\{l\}})\right) \\
&= \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \Phi_m(v^b_{X^i}|_T) - \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} \Phi_m(v^b_{X^i}|_{T\setminus\{l\}}) \\
&= I^\Phi_m(a_R, b, T) - I^\Phi_m(a_R, b, T\setminus\{l\}).
\end{aligned}
$$

Uniqueness. We show uniqueness by induction on the size of T. Suppose that I^1 and I^2 are two influence measures satisfying $f^{\mathcal{M}}$-Efficiency and Balanced Contributions. If $|T| = 1$, by $f^{\mathcal{M}}$-Efficiency,

$$I^1(a_R, b, T) = \frac{1}{n^b_{a_R}} \sum_{(X^i,Y^i)\in\mathcal{M}^b_{a_R}} v^b_{X^i}(T) = I^2(a_R, b, T).$$

Assume now that $I^1(a_R, b, S) = I^2(a_R, b, S)$ for all $S \subseteq T$ with $1 \leqslant |S| < |T|$. Then, by

Balanced Contributions, for all $l, m \in T$, $l \neq m$,

$$I_l^1(a_R, b, T) - I_m^1(a_R, b, T) = I_l^2(a_R, b, T) - I_m^2(a_R, b, T). \tag{4.4}$$

Using $f^{\mathcal{M}}$-Efficiency,

$$\sum_{l \in T} I_l^1(a_R, b, T) = \sum_{l \in T} I_l^2(a_R, b, T). \tag{4.5}$$

By (4.4) and (4.5) it is obtained that:

$$I_l^1(a_R, b, T) = I_l^2(a_R, b, T) \text{ for all } l \in T.$$

This last expression gives the uniqueness. □

4.3 Empirical results

In this section, we show the performance of the proposed influence measure (4.2) by means of a computational study. Three different experiments have been carried out using the software `R`. The objective of such simulations is to corroborate that the results obtained by the methodology introduced in the current work are in accordance with the expected ones. Furthermore, these results are compared with those obtained by the influence measure introduced in Datta et al. (2015), which counts the number of times that a modification in a feature results in a different classification. We provide the formal definition of such an influence measure below.

Definition 4.3. Given a training set $\mathcal{M} = \{(X^i, Y^i)\}_{i=1}^n$ and a classifier $f^{\mathcal{M}}$, the influence of the j-th feature is

$$\chi_j(f^{\mathcal{M}}) = \sum_{a' \in \{X^i\}} \sum_{\substack{a_j \in \mathcal{A}_j : \\ (a'_{-j}, a_j) \in \{X^i\}}} \min\left\{ \left| \arg\max_{b \in \mathcal{B}} f_b^{\mathcal{M}}(a'_{-j}, a_j) - \arg\max_{b \in \mathcal{B}} f_b^{\mathcal{M}}(a') \right|, 1 \right\},$$

where (a'_{-j}, a_j) denotes the vector $(a'_1, \ldots, a'_{j-1}, a_j, a'_{j+1}, \ldots, a'_k)$, $\mathcal{B} \subset \mathbb{N}$, and $\{X^i\}$ denotes $\{(X_1^i, \ldots, X_k^i)\}_{i=1}^n$.

The classifier used in this chapter is Breiman's random forest classifier (Breiman, 2001), implemented in Weka[2] and used through `RWeka`[3]. This choice is motivated by the excellent result of the random forest type classifiers (see, for example, Fernández-Delgado et al., 2014). The code was run on a quad-core Intel i7-8665U CPU with 16GB RAM.

[2] http://www.cs.waikato.ac.nz/ml/weka.

[3] https://cran.r-project.org/web/packages/RWeka/index.html.

The procedure adopted in the experiments is as follows. We start from a sample of individuals from which their attributes and response are known, $\mathcal{M} = \{(X^i, Y^i)\}_{i=1}^n$. Right after, such sample is used to train a previously chosen classifier, obtaining $f^{\mathcal{M}}$. To evaluate the influence of feature X_j on the response Y taking the value b, the quantities $I_j^{\Phi}(a_j, b, K)$ and $\sum_{l \in K} I_l^{\Phi}(a_j, b, K)$ are computed and analyzed for all $a_j \in \mathcal{A}_j$.

For the first experiment, a sample of 1000 instances with four binary features was generated, $\{X_1, X_2, X_3, X_4\}$. Such attributes take the values 0 and 1 with probability 0.5 (hence, $a_j \in \mathcal{A}_j = \{0, 1\},\ j \in K$). In half of the instances, the value of Y coincides with the value of X_1, while in the remaining instances the value of Y coincides with the value of X_2; note thus that $b \in \mathcal{B} = \{0, 1\}$. The following step is to select those observations whose assigned class was $b = 1$. Afterward, for each attribute $X_j,\ j \in K$, and each of its possible values, we study the influence that such feature had on the response when it took such value. Since the procedure by which the class has been generated is known, it is evident that the influence of attributes X_3 and X_4 should be independent of their values. Furthermore, the value 1 for features X_1 and X_2 should have a stronger influence in the classification than the value 0. Table 4.1 and Figure 4.1 present the results obtained for this simulation, which took a runtime of 9.3 minutes.

Table 4.1: Results for simulation 1.

$X_j, j \in K$	a_j	$\sum_{l \in K} I_l^{\Phi}(a_j, b, K)$	$I_j^{\Phi}(a_j, b, K)$
X_1	0	-0.002	-0.250
	1	0.344	0.247
X_2	0	-0.019	-0.260
	1	0.361	0.260
X_3	0	0.268	0.000
	1	0.268	0.000
X_4	0	0.258	-0.010
	1	0.277	0.010

Indeed, it can be observed that for attributes X_1 and X_2 the value $I_j^{\Phi}(a_j, b, K)$ is positive when $a_j = 1$ and negative when $a_j = 0$, which means that features X_1 and X_2 taking the value 1 works in favour of the response resulting in 1, unlike what happens if these features are worth 0. Note also that $\sum_{l \in K} I_l^{\Phi}(a_j, b, K)$ is the total influence of the four features on the response being 1 when feature X_j takes the value a_j. In view of the results obtained, for features X_1 and X_2 the quantities $I_j^{\Phi}(a_j, b, K)$ and $\sum_{l \in K} I_l^{\Phi}(a_j, b, K)$ are closer when $a_j = 1$ than when $a_j = 0$. Thus, the total influence on the response being 1 when either X_1 or X_2 is

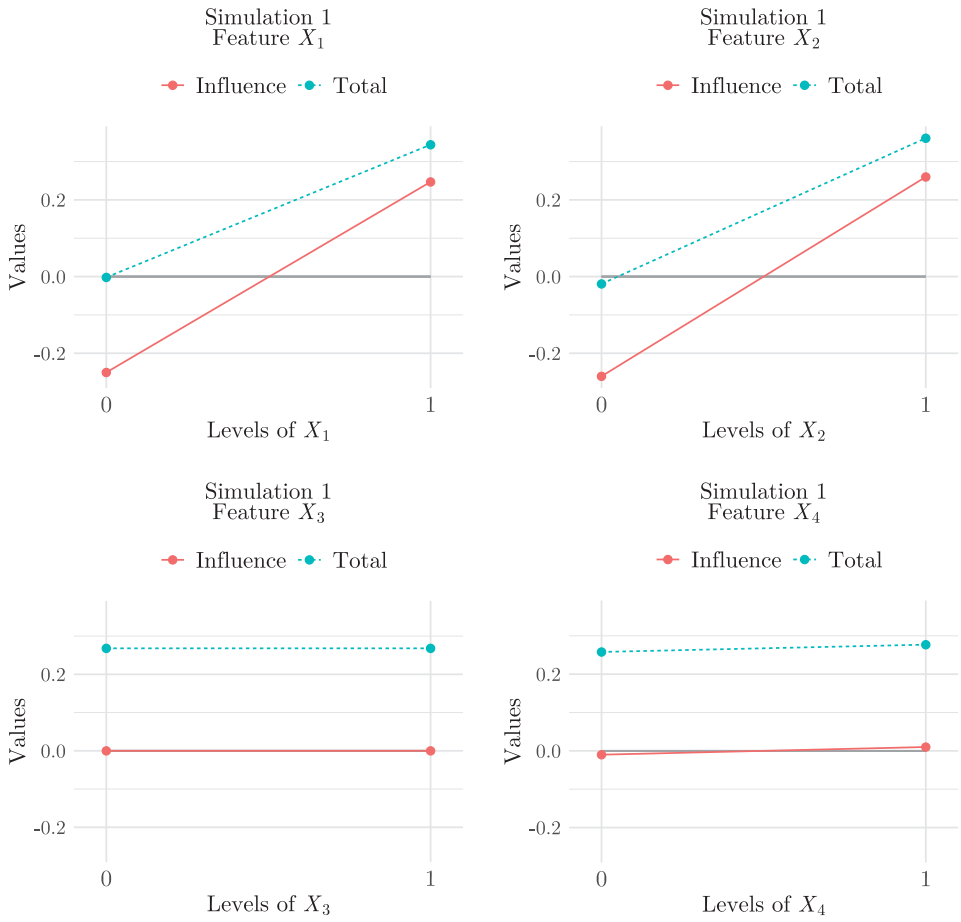


Figure 4.1: Influence and total influence for the features (Simulation 1).

1 is in fact due to these specific attributes taking the value 1. In the case of features X_3 and X_4, their influence is near 0 whatever value they take.

Applying the procedure in Datta et al. (2015) to the previous experiment, we obtain the measure $(0.50, 0.50, 0.25, 0.25)$. As expected, features X_1 and X_2 present a higher influence than X_3 and X_4. Just as we have already mentioned, Datta et al.'s procedure measures the number of times that a change in a specific attribute produces a different response. Thus, it only takes positive values, which prevents us from knowing the direction of the influence. In our case, setting features X_1 and X_2 to 0 works against the response being 1, and this is made clear by the negative sign of their influences.

The second experiment differs from the previous one in the procedure to assign the class to the instances. The response is now generated as a binary vector that takes the values 0 and 1 with probability 0.5, independently of the attributes. The goal of this simulation is to show that the influence of the features in the classification of the instances with response $b = 1$

does not depend on the features' values. Table 4.2 and Figure 4.2 present the results obtained for this simulation. The computational time was 12.4 minutes.

Table 4.2: Results for simulation 2.

$X_j, j \in K$	a_j	$\sum_{l \in K} I_l^{\Phi}(a_j, b, K)$	$I_j^{\Phi}(a_j, b, K)$
X_1	0	-0.001	-0.009
	1	0.017	0.011
X_2	0	-0.012	-0.019
	1	0.026	0.023
X_3	0	0.007	-0.002
	1	0.010	0.006
X_4	0	0.002	-0.003
	1	0.014	0.005

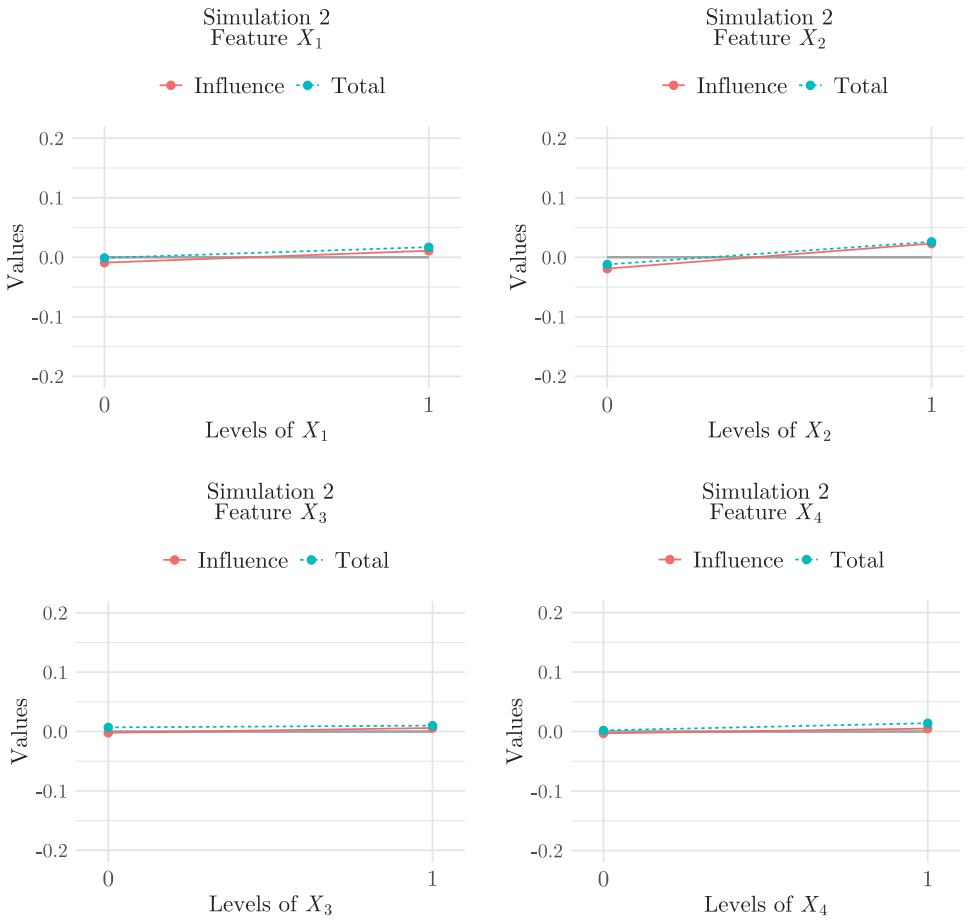


Figure 4.2: Influence and total influence for the features (Simulation 2).

Again, the outcomes are as expected: for each feature, there are barely any differences in

the values $I_j^{\Phi}(a_j, b, K)$ and $\sum_{l\in K} I_l^{\Phi}(a_j, b, K)$ when a_j changes. In this case, Datta et al.'s measure resulted in $(0.375, 0.375, 0.375, 0.375)$. The response is not influenced by any one attribute more than the others. However, because the class was generated independently of the features, one would expect their influence to be zero.

Finally, we have considered the non-binary case. Now, the four attributes can take the values 0, 1 and 2 with equal probability, and the class of the response is computed as follows: in $1/3$ of the instances, it is the value of attribute X_1 that determines the response; while in the remaining $2/3$, it is attribute X_2 that determines it. Table 4.3 and Figure 4.3 illustrate the results. This took a runtime of 13.3 minutes.

Table 4.3: Results for simulation 3.

$X_j, j \in K$	a_j	$\sum_{l\in K} I_l^{\Phi}(a_j, b, K)$	$I_j^{\Phi}(a_j, b, K)$
X_1	0	0.364	-0.091
	1	0.455	0.233
	2	0.360	-0.120
X_2	0	0.105	-0.172
	1	0.495	0.445
	2	0.005	-0.245
X_3	0	0.421	-0.012
	1	0.391	0.012
	2	0.424	0.016
X_4	0	0.410	0.042
	1	0.385	-0.041
	2	0.435	0.020

The outcomes obtained show that changes in features X_3 and X_4 do not affect the response being $b = 1$, and their influence is almost zero whatever their values. Nevertheless, the value 1 of attributes X_1 and X_2 has a positive influence, which is larger in the case of the latter. On the contrary, when these attributes take the values 0 and 2, their influence is negative. This speaks against the class resulting in 1. In this case, the influence measure of Datta et al. is $(0.321, 1.827, 0.296, 0.296)$. This result shows that X_2 is the most influential feature, and that X_1 is more relevant than X_3 and X_4. Nevertheless, this measure does not properly capture the magnitude of how much more influential attribute X_1 is in comparison to X_3 and X_4.

In view of the previous results, our methodology seems to be appropriate to study the influence that the different feature values have on the classification of individuals. Since the experiments are satisfactory, this analytic tool can be applied to real-life problems. Consequently, this procedure has been employed on a real dataset concerning COVID-19 patients,

whose results are presented in the next section.

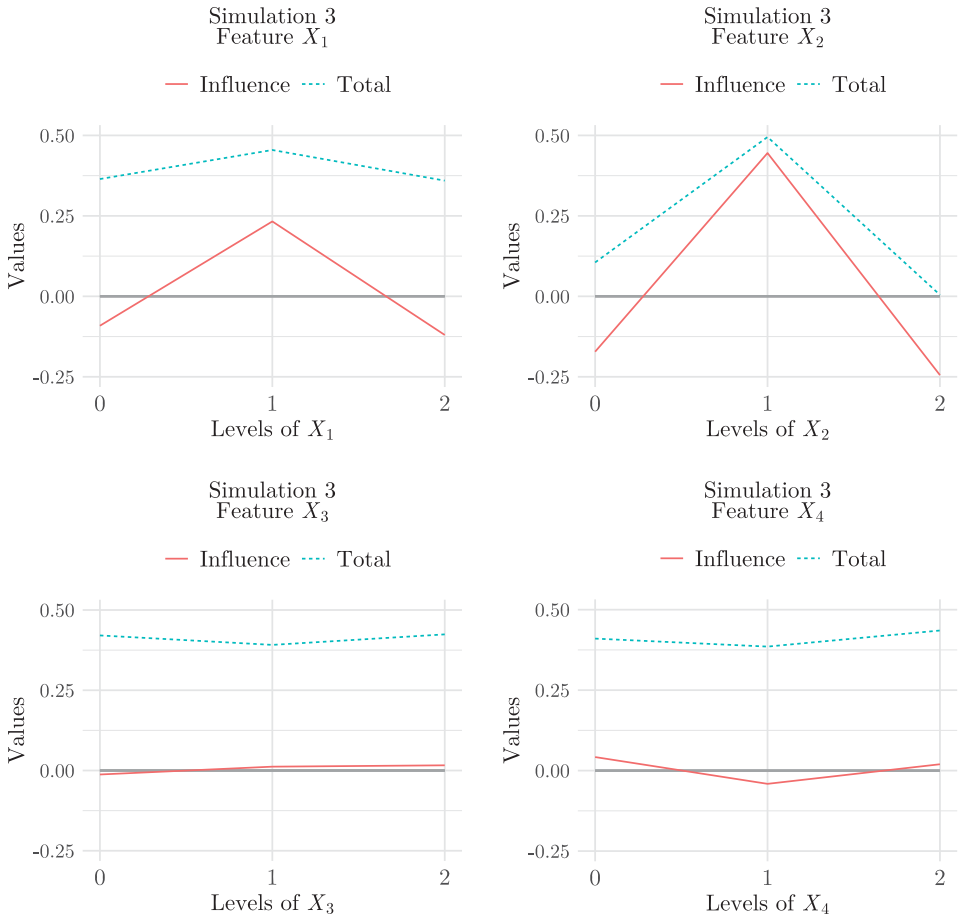


Figure 4.3: Influence and total influence for the features (Simulation 3).

4.4 Application of our influence measure to COVID-19 data

This section analyzes a database of 10,454 patients from Galicia (a region in the northwest of Spain) infected with COVID-19 from March 6, 2020 to May 7, 2020. The objective is to study the influence of various patients' characteristics in three binary response variables of special interest: the need for hospitalization, the need for ICU admission, and the eventual decease. The emphasis is not on the predictive classification of new patients, but on the analysis of the characteristics that influenced the patients whose complete history is known to have a positive response in the binary variables indicated. On the other hand, what follows is not intended to be an exhaustive study of these data to draw definitive conclusions about

the evolution of COVID-19, but simply an illustration of some of the uses of the measure of influence we introduced in Section 4.2.

The features that have been considered in this study are the following:

- **Sex**: 0 (woman), 1 (man).
- **Age**: 0 (0–49 y/o), 1 (50–64 y/o), 2 (65–79 y/o), 3 (80 y/o and over).
- **Cardiovascular diseases**: 0 (no diseases), 1 (mild diseases), 2 (severe diseases: ischemia with angina, infarction, stroke).
- **Respiratory diseases**: 0 (no diseases), 1 (mild diseases), 2 (severe diseases: malignancy, COPD, pneumonia).
- **Metabolic diseases**: 0 (no diseases), 1 (mild diseases), 2 (severe diseases: malignancy, insulin-dependent diabetes).
- **Urinary diseases**: 0 (none or mild diseases), 1 (severe diseases: malignancy, kidney failure).

The binary response variables considered in this application are:

- **Decease (exitus)**: 0 (no), 1 (yes).
- **ICU admission**: 0 (no), 1 (yes).
- **Need for hospitalization**: 0 (no), 1 (yes).

Next, we applied the methodology outlined in Section 4.2 to measure the influence of the features in the classification with respect to the binary response variables. For instance, the interest would reside in selecting those individuals who resulted in decease (that is, $\texttt{decease} = 1$) when our purpose is to know the most influential attributes for the exitus. Note that to estimate the influence of feature X_j on Y, we use the influence that X_j has in the classification of the elements of the sample $\mathcal{M}$ using an excellent classifier since it is precisely trained with the sample $\mathcal{M}$. As in the previous section, we use the random forest classifier introduced by Breiman (2001) and implemented in `R` through the `RWeka` library.

Let $\{X_1 = \texttt{sex}, X_2 = \texttt{age}, X_3 = \texttt{cardi}, X_4 = \texttt{resp}, X_5 = \texttt{meta}, X_6 = \texttt{uri}\}$ be the set of features. We start the analysis by presenting Figures 4.4, 4.5 and 4.6, which display the influence and total influence of the different features' values on the three classification problems. Let us explain in more detail what the graphics in the figures show. In each of the graphics a response variable is chosen and set its value to 1, and also a feature is chosen. The graphic shows in red (solid) the measure of influence of the chosen feature when we set its value to each of the possible values it can take (feature influence), and in blue (dashed) the

sum of the measures of influence of all the features (total influence). The objective of these figures is to identify what we call *influence scenarios*. An influence scenario is detected when the total influence shown in the corresponding graphic deviates noticeably from zero.

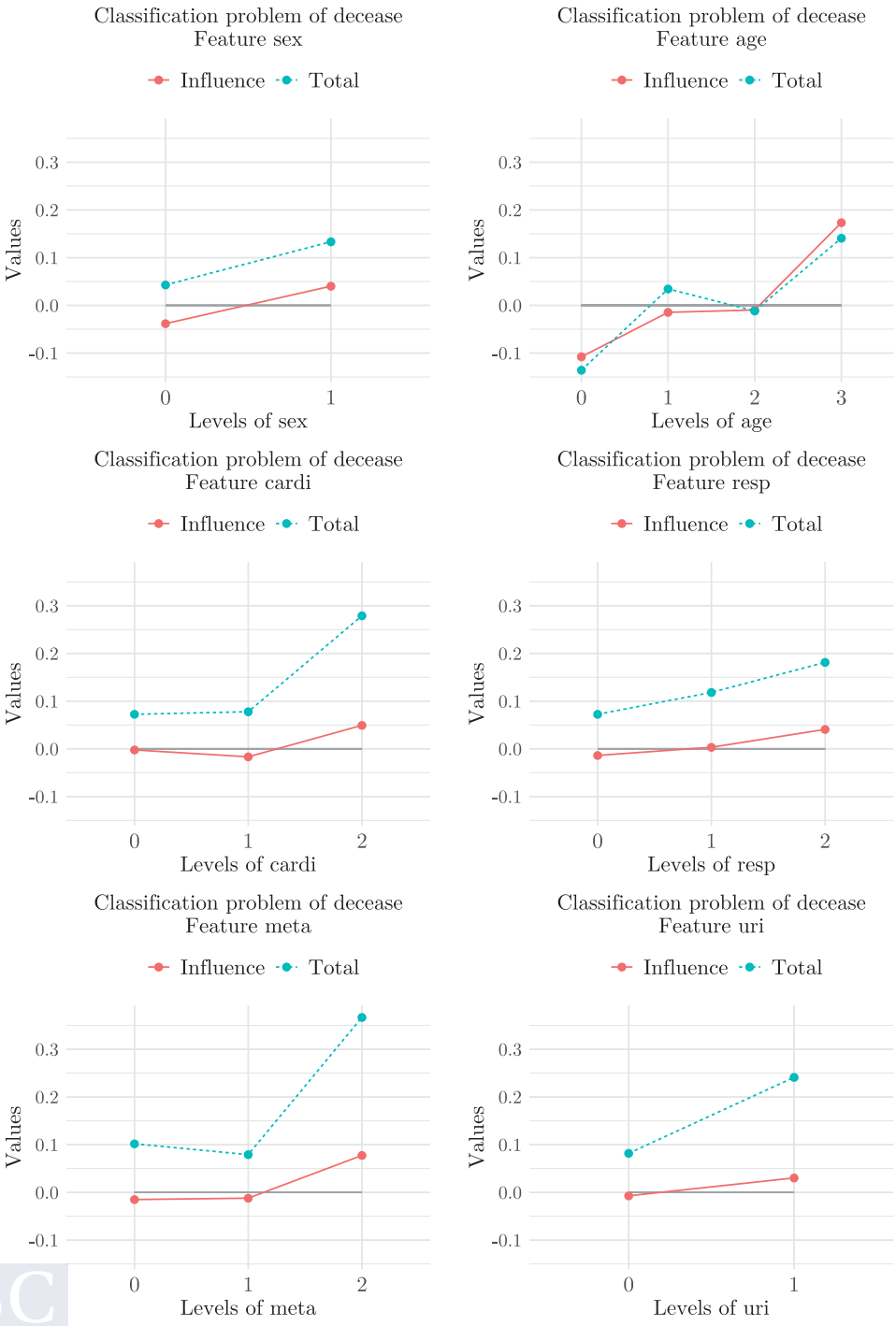


Figure 4.4: Influence and total influence for the features on the decease.

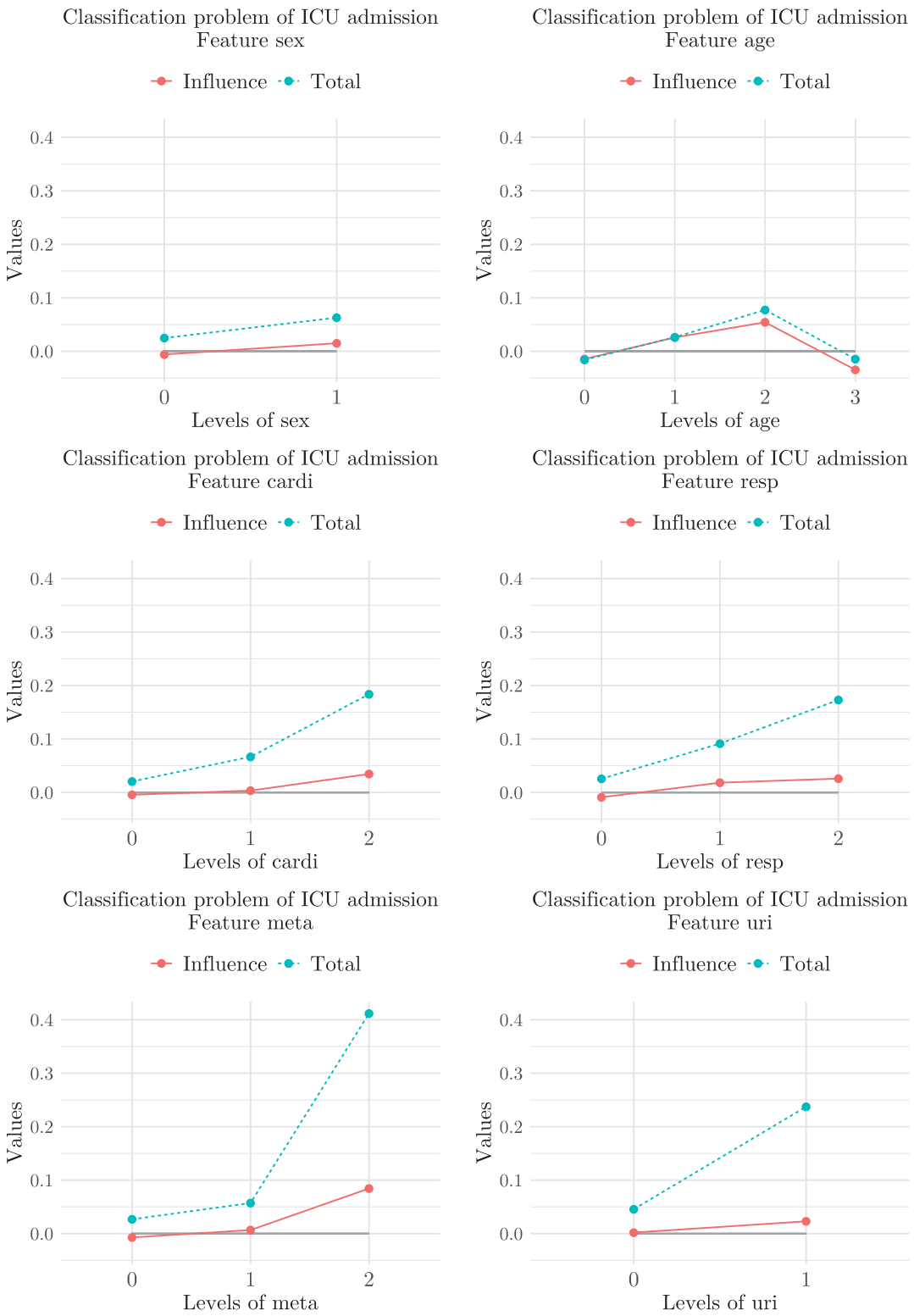


Figure 4.5: Influence and total influence for the features on the ICU admission.

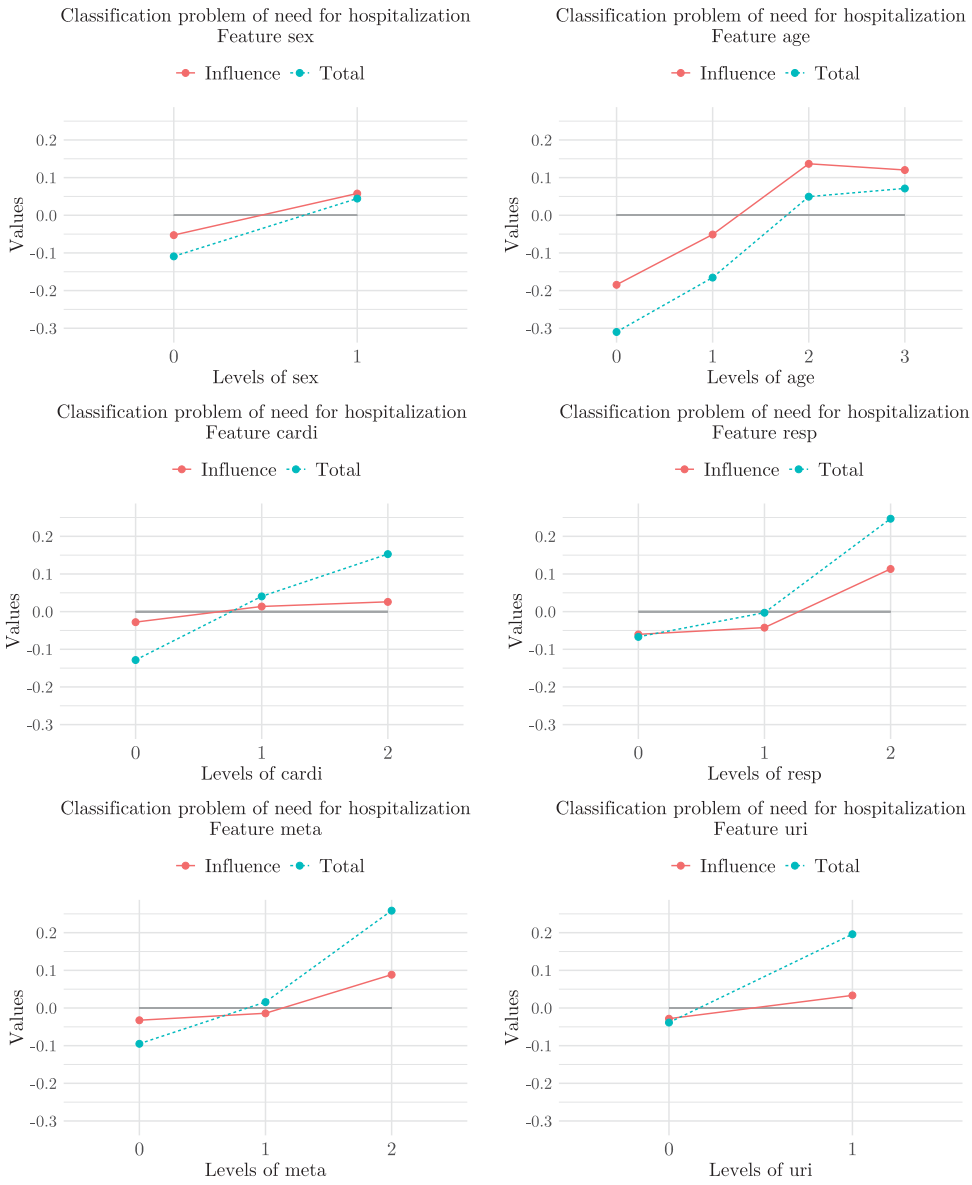


Figure 4.6: Influence and total influence for the features on the need for hospitalization.

For example, in Figure 4.4 several influence scenarios can be identified. The first is the case of `age`, both when it is worth 0 and when it is worth 3. There are two influence scenarios here that allow us to state that in the case of young individuals (`age` $= 0$) and in the case of old individuals (`age` $= 3$) we detect an important influence of the features on mortality, negative in the first case and positive in the second. We can observe that in this graphic the red and blue lines (`age` influence and total influence, respectively) are very close, which means that this total influence is mainly due to age.

Other influence scenarios that can be inferred from the figure are those corresponding to the feature `cardi` being 2 and the feature `meta` being 2. Note, however, that in such

scenarios the red and blue lines are noticeably separated, which means that the significant total influence detected is not primarily due to the features chosen in each case. Therefore, for each of these two scenarios, Table 4.4 presents the value of the influence measure for all the features, in order to identify which ones are influencing the most.

Table 4.4: Influence measure. Decease = 1.

	sex	age	cardi	resp	meta	uri	Total
cardi = 2	0.025	**0.151**	**0.049**	0.016	0.014	0.023	0.279
meta = 2	0.035	**0.142**	0.034	0.062	**0.078**	0.017	0.367

From Table 4.4 it can be seen that age is the most influential feature in these two scenarios, although the features chosen in each case (cardi and meta, respectively) are the second most influential.

Figure 4.5 shows, surprisingly, a minor influence of age on ICU admissions. This is probably because in the first wave of COVID-19 in Spain, a considerable number of elderly died in nursing homes before they could even be hospitalized or admitted to ICU. In any case, age generates an influence scenario when it is worth 2. As in Figure 4.4, in the case of age the blue and red lines are very close, showing that the total influence in this particular situation is mainly due to age.

Another influence scenario presented in Figure 4.5 is the one corresponding to the meta feature being equal to 2. In that case, the blue and red lines are far apart, so we show in Table 4.5 the value of the influence measure for all features. It can be observed that all features are influential, although the most influential are, in this order, age and metabolic diseases.

Table 4.5: Influence measure. ICU admission = 1.

	sex	age	cardi	resp	meta	uri	Total
meta = 2	0.064	**0.098**	0.072	0.071	**0.084**	0.022	0.412

Figure 4.6 allows us to identify other influence scenarios, among which we highlight those corresponding to age equal to 0, meta equal to 2 and resp equal to 2. In this case, although the blue and red lines tend to coincide more in the age feature, they are considerably separated in all the influence scenarios. Therefore, we show in Table 4.6 the value of the influence measure for all features in the three scenarios.

Table 4.6: Influence measure. Need for hospitalization = 1.

	sex	age	cardi	resp	meta	uri	Total
age $= 0$	-0.001	**-0.184**	-0.013	**-0.047**	-0.030	-0.034	-0.310
resp $= 2$	0.039	**0.085**	-0.009	**0.113**	0.014	0.005	0.247
meta $= 2$	0.026	**0.095**	0.054	0.012	**0.089**	-0.017	0.247

Again, age remains a highly influential feature in the occurrence of hospitalization in all the influence scenarios we have detected. In the first scenario, when age is 0, what we observe is that the marked tendency towards less hospitalization when patients are young is mainly due to their youth, although we also detect an important influence of good health in terms of respiratory ailments. In the influence scenario when resp $= 2$, the measure indicates that respiratory diseases are the most influential in the need for hospitalization, even more so than age. Somehow we detect that respiratory pathologies, in addition to age, are considerably influential in the need for hospitalization of COVID-19 patients.

In light of the above, it is evident that the most influential feature in all the response variables considered is age: young people are less likely to need hospitalization and admission to the ICU, as well as to die from COVID-19; the only exception we detected is that elderly people who die have a tendency to die quickly, even before being admitted to the ICU.

With this in mind, we could look further for other influential features by eliminating the age effect. That is, we can remove age from the list of features (i.e., following the notation in Section 4.2, $T = K\backslash\{2\}$, where $X_2 =$ age) and calculate the corresponding measure of influence. Through this approach, the expectation is that fewer influential scenarios will be detected; but in detected cases, the most influential features after age may come to light. We perform this analysis for the subsample in which we have the largest number of observations: the one corresponding to need for hospitalization equal to 1.

Figure 4.7 seems to confirm the considerable influence of respiratory diseases on the need for hospitalization of COVID-19 patients. Indeed, the only positive influence scenario detected occurs when resp $= 2$. Note also that, in this case, the blue and red lines are close, so that the total influence detected is mostly due to respiratory pathologies.

There is another scenario of influence when cardi $= 0$. In this case, it is striking that the red line is close to the point $(0, 0)$. This seems to indicate that in healthy individuals regarding cardiac functions an important influence on the decrease in hospitalizations is detected, but however such a decrease is not due to the feature cardi. To detect which is the most influential feature in this case, we show in Table 4.7 the value of the measure of influence when cardi $= 0$ and any other of the pathologies considered is also 0. Notice that in these three cases, feature resp is the most influential by far. Once again, the data we handle seem

to confirm the important influence of the presence of respiratory pathologies on the need for hospitalization of COVID-19 patients.

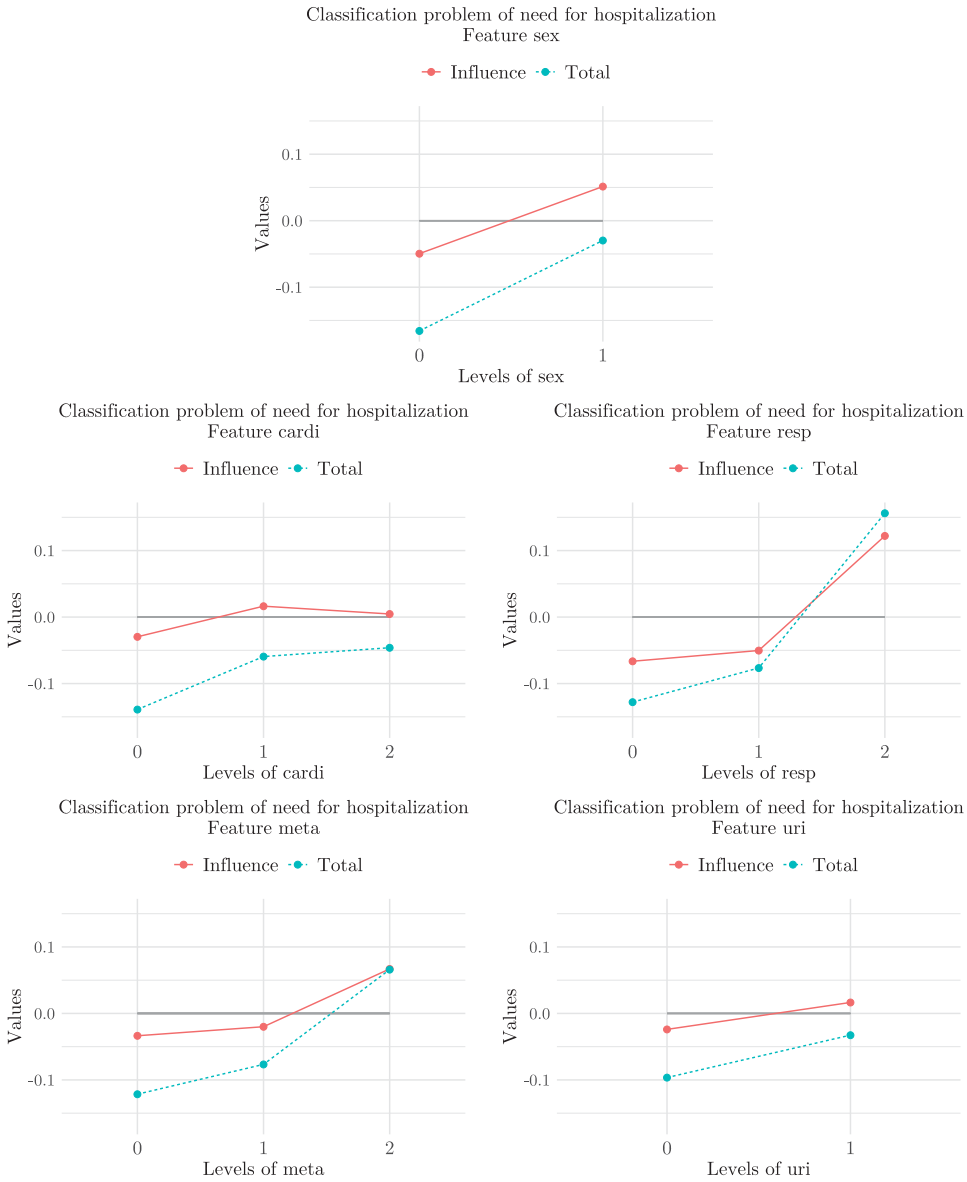


Figure 4.7: Influence and total influence for features $K\backslash\{2\}$ on the need for hospitalization.

Table 4.7: Influence measure without considering age. Need for hospitalization = 1.

	sex	cardi	resp	meta	uri	Total
cardi = 0, resp = 0	0.006	-0.028	**-0.063**	-0.025	-0.046	-0.156
cardi = 0, meta = 0	0.009	-0.038	**-0.052**	-0.033	-0.037	-0.151
cardi = 0, uri = 0	0.005	-0.030	**-0.054**	-0.022	-0.042	-0.143

4.5 Conclusions and further research

This chapter introduces a new general measure of the influence that various features of a set of individuals have on their classification. For the construction of such measure, we consider several ideas taken from game theory. In particular, we define a cooperative game (whose players are the features considered) and apply a game theoretical solution, known as the Shapley value. An axiomatic characterization theorem for the proposed influence measure is stated and mathematically proved. The properties used in this result are adaptations of Shapley value's properties in the general context of cooperative games, and are highly desirable from the exploratory data analysis point of view. To test the scope and adequacy of the proposed influence measure, a control experiment that provides a very satisfactory result is designed. Our proposal is also compared with the influence measure defined in Datta et al. (2015), which also uses ideas from game theory, leading our method to more suitable outcomes from the interpretability perspective. Section 4.4 provides an application of our measure to the study of a Spanish database of patients infected with COVID-19 from the first wave of the pandemic, between March and May 2020. The aim of this application is to determine which demographic features, as well as previous pathologies, are the most influential in the classification of a patient regarding their potential need for hospitalization, admission to the intensive care unit, or death. Initial results obtained present a promising future for the technique proposed here as a decision support tool. It serves, in particular, to alert medical professionals of the importance of certain patient characteristics, such as age or prior pathologies, as opposed to the lesser importance or influence of others. Such characteristics potentially pose an added difficulty in patients with a given disease, which should be taken into account both in the care and treatment that these patients should receive and in the planning of resources destined for them.

As for future lines of research, we believe that additional work on the recently introduced measure of influence is worthwhile. We cite, for example, the desirability of further analyzing the sensitivity of the results provided by the measure of influence according to the classifiers used, exploring its relation with other statistical techniques of multivariate analysis, as well as extending it to continuous scenarios. It would also be possible to complete the application presented using data from successive waves of the COVID-19 pandemic and taking into account the existence of different virus strains.

Part III

On cooperative Operations Research problems

Introduction to cooperative sequencing and minimum cost spanning tree problems

Part III of this dissertation is based on a specific multi-agent cooperative Operations Research problem, called the graph machine scheduling problem (GMS-problem). It is organized into two closely linked chapters that study the aforementioned problem from an optimization and cost allocation point of view, respectively.

The GMS-problem arises from combining ideas of both the minimum cost spanning tree problems and sequencing problems, of which we will give a brief introduction below. The objectives pursued in dealing with each of these situations will also be outlined.

In *minimum cost spanning tree problems* (MCSTPs), several agents, located at different geographical areas, are interested in a certain service, which will be provided by a common supplier, called the source. The agents are served by means of connections that have an associated (positive) cost, and it is not relevant whether they are connected to the source directly or indirectly. The objective of the MCSTP is to connect all the agents to the source at the lowest possible cost. In this way, the focus is on determining the so-called minimum cost spanning trees (MCSTs). A tree is a set of edges such that there is a single path from the source to each of the agents, and the cost of a tree is the sum of the costs of all the edges belonging to it. There are numerous real-world situations that can be modeled as an MCSTP. For example, when a set of people are connected to a central computer or to a cable television system (Claus and Kleitman, 1973). In Bergantiños and Lorenzo (2004), a problem is presented in which a group of inhabitants requires the construction of a water supply network coming from a common reservoir. MCSTs can be computed in a polynomial time and the most common methods are Kruskal's algorithm (Kruskal, 1956) and Prim's algorithm (Prim, 1957).

In *deterministic one-machine sequencing problems*, a set of jobs needs to be processed on a machine. Each of these jobs is identified with one agent and has associated with it: a processing time, i.e., the time needed by the machine to process that specific job, and a cost function, which indicates how costly it is for that agent to spend a unit of time in the system.

The completion time is defined as the total time that a certain agent spends in the system until it is processed. The main objective in sequencing problems consists in finding an optimal order, i.e., an order on the jobs that minimizes the total aggregated cost of all agents. The cost of an agent will naturally depend on its completion time, and this dependence is linear in the classical model (Smith, 1956).

There are, however, numerous variants of sequencing problems that allow for an adaptation to real situations. One of them is the sequencing problem with precedence constraints, where some jobs need to be processed before others. Baker (1971), Sidney (1975), and Hamers et al. (2005) analyze this type of situations. Baker and Su (1974) studies the problem where jobs have ready times and due dates. This latter feature is also incorporated in Borm et al. (2002). In Cabo et al. (2015), a problem in which the machine can process more than one job is addressed. The above works are restricted to a single machine, but sequencing problems with multiple machines are also treated in the literature, see Cheng and Gupta (1989), Lee (1991), Alidaee and Ahmadian (1993), Serafini (1996), Schmidt (2000), and Shen et al. (2013). More recently, sequencing problems have been considered in the field of robotics. Alatartsev et al. (2015) is a survey on this latter topic.

As we have just discussed, for both MCSTPs and sequencing problems, solving an optimization problem is fundamental. However, a further issue of great relevance in a cooperative setting is the allocation of costs among the different agents involved. This serves to establish and maintain cooperation between the agents.

In the case of MCSTPs, Claus and Kleitman (1973) is the first to consider this issue. Bird (1976), Kar (2002), and Dutta and Kar (2004) introduced different rules that are still very popular today. Nevertheless, other rules have been subsequently proposed (see Bergantiños and Vidal-Puga, 2007a). Numerous works have focused on approaching the cost allocation problem from the game theory perspective. Bird (1976) associated a cooperative game with transferable utility to each MCSTP, leading to the MCST games. Works such as Granot and Huberman (1981, 1984), Curiel (1997), Dutta and Kar (2004), Fernández et al. (2004), Norde et al. (2004), Bergantiños and Vidal-Puga (2007b), and Estévez-Fernández and Reijnierse (2014) were devoted to studying this class of games. Furthermore, Kar (2002), Bergantiños and Vidal-Puga (2007a), Bergantiños and Lorenzo-Freire (2008a, 2008b), Bergantiños and Gómez-Rúa (2010, 2015), and Gómez-Rúa and Vidal-Puga (2017) introduced and axiomatically characterized cost allocation rules for MCST games. Examples of papers that study MCST cost allocation rules without relying on an intermediate cooperative game include Feltkamp et al. (1994), Moretti et al. (2004), Dutta and Kar (2004), Tijs et al. (2006), Bergantiños and Kar (2010), Bergantiños et al. (2010, 2011), and Bergantiños and Vidal-Puga (2011,

2015). Bergantiños and Vidal-Puga (2021) is a recent review.

In most cooperative sequencing problems, an initial order is assumed as a starting point and the focus is on the allocation of cost savings with respect to this initial order. For the classical model, Curiel et al. (1989) introduces and axiomatically characterizes an allocation rule, the *equal gain splitting rule* (EGS-rule), based on neighbor switches to derive an optimal order from the initial one, which was later generalized in Hamers et al. (1996). Saavedra-Nieves et al. (2020) and Schouten et al. (2021) propose allocation rules for sequencing situations with non-linear cost functions. Sequencing problems have been extensively dealt with from a game-theoretic approach: as in MCSTPs, we can associate a cooperative game with a sequencing problem, resulting in sequencing games. Papers that address sequencing games include Curiel et al. (1989), Tijs et al. (1984), Borm et al. (2002), Calleja et al. (2002, 2006), Hamers et al. (2005), Slikker (2005), Estévez-Fernández et al. (2008), Çiftçi et al. (2013), Curiel (2015), Musegaas et al. (2018), Saavedra-Nieves et al. (2020) and Schouten et al. (2021).

As already mentioned, the GMS-problem as considered in this part of the thesis stems from a combination of concepts taken from the MCSTPs and sequencing problems. In our setting, we start from a graph $(N \cup \{0\}, E)$ where N is the set of nodes corresponding to the agents, 0 is the source node that must serve all agents, and E is a set of edges connecting the nodes. Each edge has a connection time, and each agent has a cost depending on the time it gets connected. We will aim, on the one hand, to find the optimal ordering of the agents such that the total connection costs are minimized; and on the other hand, to find a fair allocation of these costs.

In Chapter 5, the GMS-problem is tackled from an optimization perspective. Firstly, a situation from the field of logistics is presented which fits into our model. We illustrate that, for general GMS-problems, finding an optimal order is complex even for very simple situations, which leads us to consider specific subproblems that can explicitly be solved. These happen to be particular cases of sequencing problems with precedence constraints, as analyzed in Sidney (1975). We propose a variant of Sidney's algorithms to solve particular cases. This variant will contain specific features that will be essential for the cost allocation analysis in Chapter 6.

Chapter 6 focuses on the allocation issue: an allocation rule (the κ rule) is introduced to distribute the costs of an optimal order among the players involved. This rule relies on its construction on the algorithms described in the previous chapter. The calculation of this rule is illustrated by means of several examples of GMS-problems.

Both chapters are collected in Davila-Pena et al. (2022a).

Chapter 5

The graph machine scheduling problem (GMS-problem)

This chapter studies a new Operations Research problem, which we have called the graph machine scheduling problem (GMS-problem). The GMS-problem combines ideas from the minimum cost spanning tree problem and sequencing problem. Given that solving a general GMS-problem is complex, we restrict attention to GMS-problems on trees and propose a recursive method to solve these tree GMS-problems. The recursive procedure is described in detail, and examples are used to illustrate various aspects of the procedure. The contents of this chapter are collected in Davila-Pena et al. (2022a).

Contents

5.1 Introduction

As argued in Bergantiños et al. (2014), there are many real-life problems that require the construction of infrastructures to connect a set of agents to a source, either directly or indirectly. One of them is the urban supply of water from a general reservoir to certain points of interest (agents), which involves building pipelines throughout a city. Installing pipes between two points takes a certain amount of time. The first problem that arises in this kind of situation is the question of where the pipelines should go. The objective will thus be to connect all the agents to the network in such a way that the total time involved is minimized. To address this

problem, the standard MCST setting has been widely applied (see, for example, Curiel, 1997 or Bergantiños and Lorenzo, 2004).

However, it is often essential for the agents to be provided with water at all times (e.g., a hospital), so they have to contract an external service company for as long as the water supply does not reach them. Thus, each agent has an associated coefficient that indicates the cost per unit of time in the system, i.e., per unit of time for which the pipes that connect it to the source are not yet constructed. Therefore, the cost of an agent will be determined by its coefficient cost per unit of time multiplied by the total time required to connect that specific agent. This total time will depend on when this agent is connected to the source: for example, if agent 2 is connected to the source via agent 1, then the pipeline connecting the source to agent 1 must be constructed first, followed by the pipeline connecting agent 1 to agent 2. Hence, the total time required to connect agent 2 to the source would be the sum of the construction times of the pipelines connecting the source to agent 1, and agent 1 to agent 2. Thus, the issue is to minimize the total aggregate costs instead of just the total construction time for the project as a whole.

Situations such as those described above result in a new type of problem, which we have called the graph machine scheduling problem (GMS-problem). One issue we would like to highlight is the proximity of our problem to two other problems already studied in the literature. First, the GMS-problem is closely related to the MCSTP. However, in a GMS-problem the costs are computed in a completely different way. In particular, the order in which the edges are constructed has a substantial effect on the cost in our setup, whereas this order is irrelevant in calculating the costs in an MCSTP. Second, the GMS-problem is deeply linked to sequencing problems with precedence constraints. Although sequencing problems with precedence constraints have been treated in the literature before (see Hamers et al., 2005), the approach under which we will study them here has, to the best of our knowledge, never been adopted.

The remainder of this chapter is organized as follows. Section 5.2 motivates the GMS-problem through a particular example and describes the general problem in detail. The difficulty in obtaining an optimal order for the general model results in the consideration of $2-$lines, $n-$lines, and tree GMS-problems. Section 5.3 provides solution algorithms for specific subproblems, along with proofs of their optimality. Several examples are presented to illustrate these procedures.

5.2 Problem description and motivation

The challenge of optimally designing a water supply system is to determine the order in which to install the pipelines, connecting each of the agents to the source, so that the total costs are minimized. We will now summarize the situation we are facing, as well as the specific problems we want to tackle. Further on, we will formally define the GMS-problem.

The problem we address in this part of the dissertation is the following. There is a node 0 that provides a certain service, a set of nodes N that must be (directly or indirectly) connected to 0 in order to receive the service, and a set of edges $E \subseteq \{\{i,j\} \mid i,j \in N \cup \{0\}, i \neq j\}$ such that $(N \cup \{0\}, E)$ is a connected graph. The edges are initially "inactive" and each has an activation or connection time. In addition, each node has an associated parameter indicating the unit cost of its connection time to node 0. The objective pursued is twofold:

(i) First, we intend to choose an *activation order* on the nodes of N so that the sum of the costs over all nodes, the *total connection cost*, is as low as possible. This order will imply a set of edges to be activated that connects all nodes to 0. Once we have found an optimal solution to the previous question, we know the costs required to connect each node of N to 0 according to that solution. Let us call the sum of such costs the *optimal connection cost*.

(ii) The second objective is to propose a fair allocation of the optimal connection cost among the nodes of N.

Issue (i) will be tackled in the present chapter, leaving issue (ii) for Chapter 6. However, let us now treat a simple example that motivates the need to address these two issues, and also to better appreciate the complexity of the problem at hand.

Example 5.1. Consider a GMS-problem with three cities that need to be connected to a node providing a service as soon as possible. The graph in Figure 5.1 presents the possible connections between the nodes and the times required to activate each of these connections. The cost parameters associated with the three cities are 1, i.e., the cost that corresponds with the connection time of each city coincides with such time. As mentioned above, the first objective is to connect the three cities to the service node (the 0 node) while minimizing the total connection cost. Table 5.1 displays the possible orders on the nodes, along with their corresponding edges to be sequentially activated, the individual cost vector (whose coordinates represent the cost of cities 1, 2, and 3, respectively), and the total aggregated costs. By considering the 6 possible orders, one sees that the optimal order is (213). Correspondingly, the edges are

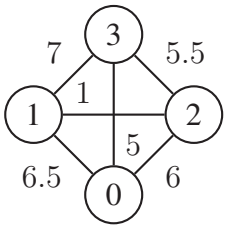


Figure 5.1: Example of a GMS-problem.

Table 5.1: Possible orders for the GMS-problem from Figure 5.1.

Order	Edges to be activated	Individual cost vector	Total cost
(123)	$\{0,1\},\{1,2\},\{0,3\}$	$(6.5, 7.5, 12.5)$	26.5
(132)	$\{0,1\},\{0,3\},\{1,2\}$	$(6.5, 12.5, 11.5)$	30.5
(213)	$\{0,2\},\{2,1\},\{0,3\}$	$(7, 6, 12)$	25
(231)	$\{0,2\},\{0,3\},\{2,1\}$	$(12, 6, 11)$	29
(312)	$\{0,3\},\{0,1\},\{1,2\}$	$(11.5, 12.5, 5)$	29
(321)	$\{0,3\},\{3,2\},\{2,1\}$	$(11.5, 10.5, 5)$	27

activated in the order $\{0,2\},\{2,1\}$, and $\{0,3\}$. The optimal connection cost is 25, which is obtained by adding the costs of the nodes, i.e., 6 for player 2, $6+1$ for player 1, and $6+1+5$ for player 3. The fact that (213) is optimal is somehow surprising. The most natural choice, from the point of view of a central planner, would be to assume that the sequence of edges to be activated should be $\{0,3\}$, $\{3,2\}$, and $\{2,1\}$, i.e., to select at each step the edge with the lowest time. This however results in the order (321), which has an associated cost of 27. As we can see, taking this myopic approach does not yield an optimal order in general. Nevertheless, as mentioned above, optimal cost allocation is an objective to be pursued. In this way, this myopic order could be used as a reference order for cost allocation by appropriately subtracting the savings $27-25=2$ from the reference costs $(11.5, 10.5, 5)$. △

Below we will formally present the proposed problem, as well as some essential definitions to address it.

A *GMS-problem* can be summarized by a tuple $(N, 0, E, \gamma, \alpha)$, where N is a finite set of jobs or players, 0 represents the machine (source), E is a set of available (precedence) edges between players and source, i.e., $E \subseteq \{\{i,j\} \mid i,j \in N \cup \{0\}, i \neq j\}$, such that $(N \cup \{0\}, E)$ is a connected graph, $\gamma\colon E \to \mathbb{R}_+$ with γ_{ij} representing the activation time of the edge $\{i,j\} \in E$, and, finally, $\alpha\colon N \to \mathbb{R}_+$, with $\alpha(i)$ representing the linear cost coefficient to spend one time unit in the system for player $i \in N$.

The main assumption is that a player $i \in N$ can only be processed by the machine if all players on a path in E from i to the machine have been processed before. A processing order

is described by a bijection $\sigma\colon N \to \{1,\dots,|N|\}$, and $\Pi(N)$ denotes the set of all processing orders. A processing order $\sigma \in \Pi(N)$ is called feasible if the aforementioned condition is met for all players. Given $\sigma \in \Pi(N)$ and $i \in N$, let $P_\sigma(i) = \{j \in N \mid \sigma(j) < \sigma(i)\}$ denote the set of predecessors of i in σ. Also, let $P^0_\sigma(i) = P_\sigma(i) \cup \{0\}$. Formally, $\sigma \in \Pi(N)$ is feasible if there exists $j \in P^0_\sigma(i)$ such that $\{i,j\} \in E$ for all $i \in N$. Let $\mathcal{F}(N)$ denote the set of all feasible orders.

Definition 5.1. Let $(N, 0, E, \gamma, \alpha)$ be a GMS-problem, and let $\sigma \in \mathcal{F}(N)$. Given $i \in N$, we define the **completion time of player i with respect to σ**, $C_i(\sigma)$, as follows:

$$C_i(\sigma) = \sum_{k \in P_\sigma(i)} C_k(\sigma) + \min\left\{\gamma_{ij} \mid j \in P^0_\sigma(i) \text{ and } \{i,j\} \in E\right\}.$$

Definition 5.2. Let $(N, 0, E, \gamma, \alpha)$ be a GMS-problem, and let $\sigma \in \mathcal{F}(N)$. Given $i \in N$, we define the **cost of player i with respect to σ**, $c_i(\sigma)$, as follows:

$$c_i(\sigma) = \alpha(i) \cdot C_i(\sigma).$$

Let $c(\sigma) = (c_i(\sigma))_{i \in N}$ denote the individual cost vector with respect to σ.

Definition 5.3. Let $(N, 0, E, \gamma, \alpha)$ be a GMS-problem, and let $\sigma \in \mathcal{F}(N)$. We define the **total cost of σ**, $TC(\sigma)$, as follows:

$$TC(\sigma) = \sum_{i \in N} c_i(\sigma). \tag{5.1}$$

The current chapter aims to determine an optimal order $\hat{\sigma} \in \mathcal{F}(N)$ that minimizes total costs among all feasible processing orders. It is important to note that the problem of finding an optimal order for general graph structures $(N \cup \{0\}, E)$ is hard. The optimal solution of the example presented in Figure 5.1 already shows some unexpected peculiarities, highlighting the potential complexity of the GMS-problem. Example 5.1 could be restricted to a tree and the optimization of the resulting problem would still not be straightforward. The following example illustrates this aspect.

Example 5.2. Consider the GMS-problem of Figure 5.2, in which there is only one possible connection between any two nodes. Thus, it is obvious which edges are to be connected, however, the order in which to perform these activations is not trivial. In this case, there are only two feasible orders for the cities: (321) and (213), which have associated costs of 28 and 25, respectively. Hence, (213) is the optimal order. It is therefore necessary to first activate the edge $\{0,2\}$, then $\{2,1\}$, and finally $\{0,3\}$. However, from a central planner's perspective,

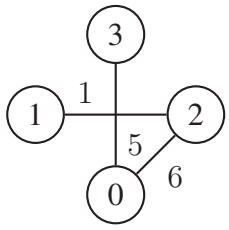


Figure 5.2: Example of a tree GMS-problem.

and in analogy with Example 5.1, the myopic order obtained for this problem would be (321), which we have already seen it is not optimal. $\triangle$

The above example shows that obtaining an optimal order for GMS-problems is not a straightforward task, even when considering simpler graphs. Therefore, we will focus on GMS-problems such that $(N \cup \{0\}, E)$ is a tree. Still in this case, Example 5.2 demonstrates that the resulting optimization problem is interesting and challenging. Again, we see that the myopic order need not be optimal. However, restricting the GMS-problem to trees will enable us to identify subproblems that are manageable from the point of view of optimization, and thus we will be able to obtain solution algorithms.

5.3 Solving GMS-problems on trees: a recursive approach

As already mentioned in Section 5.2, finding an optimal order for the GMS-problem can be complex in general. In the following, we will limit ourselves to situations in which the graph is a tree. It turns out that the corresponding optimization problem is equivalent to a sequencing problem with precedence constraints studied in Sidney (1975). We modify the algorithm of Sidney mainly for allocation purposes, which will be discussed in Chapter 6. We explain this variant of Sidney's algorithm in detail in this section.

It should be stressed that by restricting the problem to trees, there will be only one path between any two nodes. With the purpose of simplifying the notation, the GMS-problems treated from now on will be denoted by a tuple $(N, 0, E, \gamma, \alpha)$, where γ is now a function on N, specifically $\gamma\colon N \to \mathbb{R}_+$ with $\gamma(i)$ representing the processing time of the player $i \in N$. In this way, note that equation (5.1) can be also formulated as

$$TC(\sigma) = \sum_{k=1}^{|N|} \sum_{\{j \in N | \sigma(j) \geqslant k\}} \alpha(j) \cdot \gamma(\sigma^{-1}(k)).$$

5.3.1 The $2-$lines GMS-problem

A GMS-problem $(N, 0, E, \gamma, \alpha)$ is called a $2-$*lines GMS-problem* if there exists a partition $\langle A, B\rangle$ of N with $A = \{a_1, \ldots, a_{\tilde{s}}\}$ and $B = \{b_1, \ldots, b_{\tilde{t}}\}$ with $\tilde{s} + \tilde{t} = |N|$, such that

$$E = \big\{\{0, a_1\}, \{a_1, a_2\}, \ldots, \{a_{\tilde{s}-1}, a_{\tilde{s}}\}\big\} \cup \big\{\{0, b_1\}, \{b_1, b_2\}, \ldots, \{b_{\tilde{t}-1}, b_{\tilde{t}}\}\big\}.$$

The sets A and B are called branches. For this particular case, a feasible order is described by a bijection $\sigma\colon A \cup B \to \{1, 2, \ldots, \tilde{s} + \tilde{t}\}$ such that

$$\sigma(a_k) < \sigma(a_l) \Rightarrow k < l;$$
$$\sigma(b_k) < \sigma(b_l) \Rightarrow k < l.$$

Let $\mathcal{F}(A \cup B)$ denote the set of all such feasible orders. A graphical representation of a $2-$lines GMS-problem can be seen in Figure 5.3.

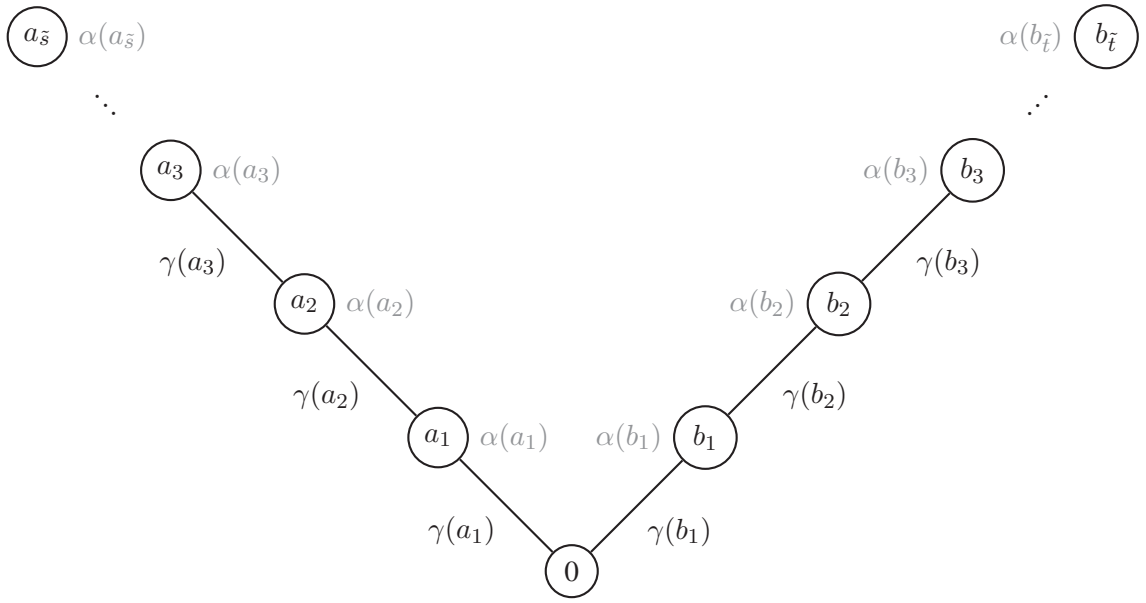


Figure 5.3: Graphical representation of a $2-$lines GMS-problem.

Definition 5.4. Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem, and let $\sigma \in \mathcal{F}(A \cup B)$. We define a **segment of A from h to l** as the following subset of A:

$$Q_{hl} = \{a_h, a_{h+1}, \ldots, a_l\},$$

where $1 \leqslant h \leqslant l \leqslant \tilde{s}$. Analogously, we define a **segment of B from h to l** as the following

subset of B:

$$R_{hl} = \{b_h, b_{h+1}, \ldots, b_l\},$$

where $1 \leqslant h \leqslant l \leqslant \tilde{t}$.

When we do not need to specify which branch a certain segment belongs to, we will use the notation X or Y. A segment from the beginning of a branch is called a **head**.

Definition 5.5. Given a segment X, we define the **cost weighted average time per edge** of X, $CAT(X)$, as

$$CAT(X) = \frac{\sum_{i \in X} \gamma(i)}{\sum_{i \in X} \alpha(i)}.$$

Definition 5.6. Given a segment X, we define its **urgency**, $U(X)$, as

$$U(X) = \frac{1}{CAT(X)}.$$

Given a $2-$lines GMS-problem, $(N, 0, E, \gamma, \alpha)$, our objective is to solve the following problem:

$$\begin{aligned} \min \quad & TC(\sigma) \\ \text{s.t.} \quad & \sigma \in \mathcal{F}(A \cup B). \end{aligned}$$

First, we informally present the outline of the proposed solving algorithm. The idea behind this algorithm is to consider a segment from the beginning of a branch (called a *pivot*, consisting of *merge candidates*), whose CAT (called *pivot's cost*) will be compared to the CAT of some other segment from the beginning of the other branch (called *opposite cost*). This pivot will be changing throughout the process regarding the results of these comparisons. We aim to merge those candidates to the source and start over with a smaller graph. The steps to follow are described below:

1. Select the node with the cheapest cost to the source directly (that is, the one with the highest urgency). If there is a tie, choose branch A. This would be our merge candidate. The other node would be our opposite node.

2. Given the pivot and its depth/length (i.e., the number of merge candidates), increase the depth of the opposite nodes if possible (otherwise, go to step 3), compute their CAT, and compare it to the pivot's cost. Two options arise:

   (a) If pivot's cost is not more expensive, go back to step 2.

(b) If the opposite cost is cheaper, exchange the merge candidates with the opposite nodes (which now constitute the new pivot). Automatically, the depth/length of the new opposite nodes is also decreased to that of the old pivot. Go to step 2 with the new pivot.

3. If it is not possible to increase the depth of the opposite nodes, then you have reached the end of the opposite branch. Hence, merge your candidates to the source. Go back to step 1.

In Algorithm 5.1 we formally present the algorithm described above. Steps 1–3 constitute an iteration of the algorithm. The output of Algorithm 5.1 is a *merge order*:

$$\hat{\sigma} = (M_1, M_2, \ldots, M_{m-1}, M_m),$$

with $m \geqslant 2$, where $M_1, M_2, \ldots, M_m$ are called *merge segments*. Of course, a merge order $\hat{\sigma}$ corresponds to one order on all players. If we do not want to highlight the merge segments, we will use the notation $\hat{\tau}$ for this order. It could be possible that both M_k and M_{k+1} belong to the same branch (because they might be merged at different steps).

Let $P_1, P_2, \ldots, P_p$ represent the pivots. These pivots do not necessarily need to be merged. But there is a relation between the merge segments and the pivots: all merge segments are pivots. Also, we know that for each iteration consecutive pivots come from different branches. There can also be duplicated pivots. Example 5.3 shows a detailed example of this algorithm, in which all these features are specified.

Algorithm 5.1 Algorithm to solve a $2-$lines GMS-problem

0. Initialize $k = 1, i = 1$.

1. Consider the $2-$lines GMS-problem $(N, 0, E, \gamma, \alpha)$. Initialize $l = 1, l' = 1$.
   - If $\dfrac{\gamma(a_1)}{\alpha(a_1)} \leqslant \dfrac{\gamma(b_1)}{\alpha(b_1)}$, select $P_i = \{a_1\}$.
   - If $\dfrac{\gamma(a_1)}{\alpha(a_1)} > \dfrac{\gamma(b_1)}{\alpha(b_1)}$, select $P_i = \{b_1\}$.

2. (a) If $P_i \subseteq A$, take $l = l + 1$.
   - If $l \leqslant \tilde{t}$, compare $CAT(P_i)$ to $CAT(R_{1l})$.
     - If $CAT(P_i) \leqslant CAT(R_{1l})$, go back to step 2.
     - If $CAT(P_i) > CAT(R_{1l})$, then $i = i + 1$, $P_i = R_{1l}$. Go back to step 2.
   - If $l > \tilde{t}$, go to step 3.

   (b) If $P_i \subseteq B$, take $l' = l' + 1$.
   - If $l' \leqslant \tilde{s}$, compare $CAT(P_i)$ to $CAT(Q_{1l'})$.
     - If $CAT(P_i) \leqslant CAT(Q_{1l'})$, go back to step 2.
     - If $CAT(P_i) > CAT(Q_{1l'})$, then $i = i + 1$, $P_i = Q_{1l'}$. Go back to step 2.
   - If $l' > \tilde{s}$, go to step 3.

3. Set $M_k = P_i$.

   (a) If $M_k \subseteq A$.
   - If $A \backslash M_k = \varnothing$, then $M_{k+j} = \{b_j\}$ for all $j \in \{1, \ldots, \tilde{t}\}$. The algorithm is finished.
   - Otherwise, let $A = r(A \backslash M_k)$, where $r \colon A \backslash M_k \to A$ is a renumbering function such that $r(a_h) = a_{h-l'}$, for all $h \in \{l' + 1, \ldots, \tilde{s}\}$. Let $\gamma(a_h) = \gamma(a_{h+l'})$ and $\alpha(a_h) = \alpha(a_{h+l'})$ for all $h \in \{1, \ldots, \tilde{s} - l'\}$. Set $N = A \cup B$, $k = k + 1$ and $i = i + 1$. Go back to step 1.

   (b) If $M_k \subseteq B$.
   - If $B \backslash M_k = \varnothing$, then $M_{k+j} = \{a_j\}$ for all $j \in \{1, \ldots, \tilde{s}\}$. The algorithm is finished.
   - Otherwise, let $B = \tilde{r}(B \backslash M_k)$, where $\tilde{r} \colon B \backslash M_k \to B$ is a renumbering function such that $r(b_h) = b_{h-l}$, for all $h \in \{l + 1, \ldots, \tilde{t}\}$. Let $\gamma(b_h) = \gamma(b_{h+l})$ and $\alpha(b_h) = \alpha(b_{h+l})$ for all $h \in \{1, \ldots, \tilde{t} - l\}$. Set $N = A \cup B$, $k = k + 1$ and $i = i + 1$. Go back to step 1.

Example 5.3. Let us consider the $2-$lines GMS-problem presented in Figure 5.4. We start by initializing $k = 1, i = 1$.

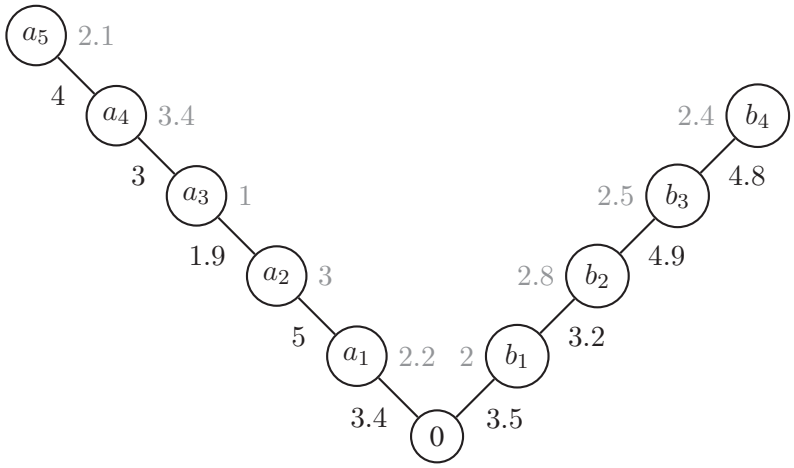


Figure 5.4: Example of a $2-$lines GMS-problem.

First iteration

1. Consider the aforementioned $2-$lines GMS-problem. Initialize $l = 1, l' = 1$. Note that:

$$\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{3.4}{2.2} < \frac{3.5}{2} = \frac{\gamma(b_1)}{\alpha(b_1)},$$

hence select $P_1 = \{a_1\}$.

2.1. Since $P_1 \subseteq A$, take $l = 2$. Consider $R_{12} = \{b_1, b_2\}$, and note that:

$$CAT(P_1) = \frac{3.4}{2.2} > \frac{3.5 + 3.2}{2 + 2.8} = CAT(R_{12}).$$

Then, $i = 2$ and we change the pivot to $P_2 = R_{12}$.

2.2. Since $P_2 \subseteq B$, take $l' = 2$. Consider $Q_{12} = \{a_1, a_2\}$. Note that:

$$CAT(P_2) = \frac{3.5 + 3.2}{2 + 2.8} < \frac{3.4 + 5}{2.2 + 3} = CAT(Q_{12}),$$

so we maintain the pivot.

2.3. Since $P_2 \subseteq B$, take $l' = 3$. Consider $Q_{13} = \{a_1, a_2, a_3\}$. Note that:

$$CAT(P_2) = \frac{3.5 + 3.2}{2 + 2.8} < \frac{3.4 + 5 + 1.9}{2.2 + 3 + 1} = CAT(Q_{13}),$$

so we maintain the pivot.

2.4. Since $P_2 \subseteq B$, take $l' = 4$. Consider $Q_{14} = \{a_1, a_2, a_3, a_4\}$. Note that:

$$CAT(P_2) = \frac{3.5 + 3.2}{2 + 2.8} > \frac{3.4 + 5 + 1.9 + 3}{2.2 + 3 + 1 + 3.4} = CAT(Q_{14}).$$

Then, $i = 3$ and we change the pivot to $P_3 = Q_{14}$.
2.5. Since $P_3 \subseteq A$, take $l = 3$. Consider $R_{13} = \{b_1, b_2, b_3\}$. Note that:

$$CAT(P_3) = \frac{3.4 + 5 + 1.9 + 3}{2.2 + 3 + 1 + 3.4} < \frac{3.5 + 3.2 + 4.9}{2 + 2.8 + 2.5} = CAT(R_{13}),$$

so the pivot does not change.
2.6. Since $P_3 \subseteq A$, take $l = 4$. Consider $R_{14} = \{b_1, b_2, b_3, b_4\}$. Note that:

$$CAT(P_3) = \frac{3.4 + 5 + 1.9 + 3}{2.2 + 3 + 1 + 3.4} < \frac{3.5 + 3.2 + 4.9 + 4.8}{2 + 2.8 + 2.5 + 2.4} = CAT(R_{14}),$$

so the pivot does not change.
2.7. Since $P_3 \subseteq A$, take $l = 5$. Note that $l > \tilde{t} = 4$, so we have already compared the CAT of our pivot, which belongs to branch A, to the $CATs$ of all possible heads from branch B.
3. $M_1 = P_3$, so we merge our pivot to the source. Since $M_1 \subseteq A$ and $A \backslash M_1 = \{a_5\} \neq \varnothing$, we need to renumber the nodes and take $k = 2$, $i = 4$. This ends the first iteration.

Second iteration

1. Consider the $2-$lines GMS-problem of Figure 5.5a. Initialize $l = 1$, $l' = 1$. Note that:

$$\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{4}{2.1} > \frac{3.5}{2} = \frac{\gamma(b_1)}{\alpha(b_1)},$$

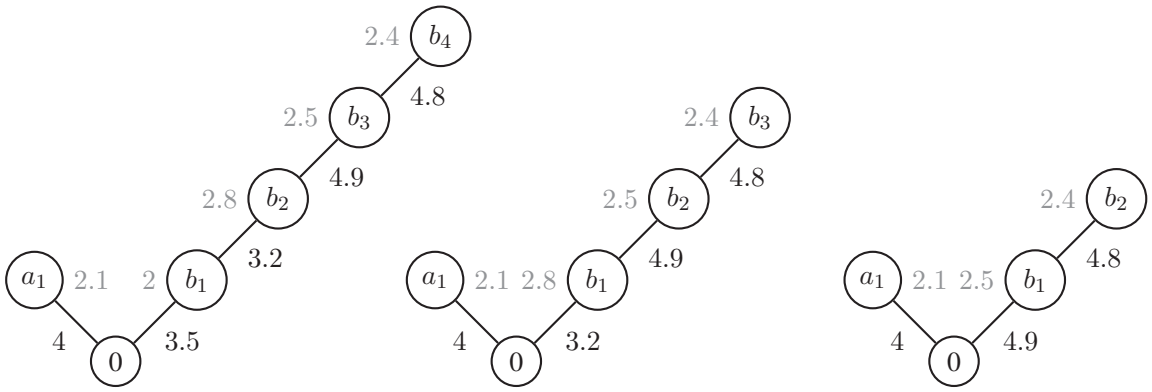
hence select $P_4 = \{b_1\}$.
2.1. Since $P_4 \subseteq B$, take $l' = 2$. Note that $l' > \tilde{s} = 1$, so we have reached the end of branch A.
3. $M_2 = P_4$, so we merge our pivot to the source. Since $M_2 \subseteq B$ and $B \backslash M_2 = \{b_2, b_3, b_4\} \neq \varnothing$, we need to renumber the nodes and take $k = 3$, $i = 5$. This ends the second iteration.

Third iteration

1. Consider the $2-$lines GMS-problem of Figure 5.5b. Initialize $l = 1$, $l' = 1$. Note that:

$$\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{4}{2.1} > \frac{3.2}{2.8} = \frac{\gamma(b_1)}{\alpha(b_1)},$$

hence select $P_5 = \{b_1\}$.
2.1. Since $P_5 \subseteq B$, take $l' = 2$. Note that $l' > \tilde{s} = 1$, so we have reached the end of branch A.



(a) Second iteration. *(b) Third iteration.* *(c) Fourth iteration.*

Figure 5.5: Resulting $2-$lines GMS-problems throughout the algorithm.

3. $M_3 = P_5$, so we merge our pivot to the source. Since $M_3 \subseteq B$ and $B \backslash M_3 = \{b_3, b_4\} \neq \varnothing$, we need to renumber the nodes and take $k = 4$, $i = 6$. This ends the third iteration.

Fourth iteration

1. Consider the $2-$lines GMS-problem of Figure 5.5c. Initialize $l = 1$, $l' = 1$. Note that:

$$\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{4}{2.1} < \frac{4.9}{2.5} = \frac{\gamma(b_1)}{\alpha(b_1)},$$

hence select $P_6 = \{a_1\}$.

2.1. Since $P_6 \subseteq A$, take $l = 2$. Consider $R_{12} = \{b_1, b_2\}$. Note that:

$$CAT(P_6) = \frac{4}{2.1} < \frac{4.9 + 4.8}{2.5 + 2.4} = CAT(R_{12}),$$

so the pivot is maintained.

2.2. Since $P_6 \subseteq A$, take $l = 3$. Note that $l > \tilde{t} = 2$, so we have reached the end of branch B.

3. $M_4 = P_6$, so we merge our pivot to the source. Since $M_4 \subseteq A$ and $A \backslash M_4 = \varnothing$, then $M_5 = \{b_1\}$ and $M_6 = \{b_2\}$. The algorithm is finished.

The algorithm outputs the following merge order:

$$\hat{\sigma} = (\underbrace{a_1 a_2 a_3 a_4}_{M_1} \underbrace{b_1}_{M_2} \underbrace{b_2}_{M_3} \underbrace{a_5}_{M_4} \underbrace{b_3}_{M_5} \underbrace{b_4}_{M_6}),$$

which has an associated cost of:

$$\begin{aligned}TC(\hat{\sigma}) &= \alpha(a_1)\cdot\gamma(a_1) + \alpha(a_2)\cdot\big(\gamma(a_1)+\gamma(a_2)\big) + \alpha(a_3)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)\big)\\ &\quad + \alpha(a_4)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)\big)\\ &\quad + \alpha(b_1)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)+\gamma(b_1)\big)\\ &\quad + \alpha(b_2)\cdot(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)+\gamma(b_1)+\gamma(b_2))\\ &\quad + \alpha(a_5)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)+\gamma(b_1)+\gamma(b_2)+\gamma(a_5)\big)\\ &\quad + \alpha(b_3)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)+\gamma(b_1)+\gamma(b_2)+\gamma(a_5)+\gamma(b_3)\big)\\ &\quad + \alpha(b_4)\cdot\big(\gamma(a_1)+\gamma(a_2)+\gamma(a_3)+\gamma(a_4)+\gamma(b_1)+\gamma(b_2)+\gamma(a_5)+\gamma(b_3)+\gamma(b_4)\big)\\ &= 2.2\cdot 3.4 + 3\cdot 8.4 + 1\cdot 10.3 + 3.4\cdot 13.3 + 2\cdot 16.8 + 2.8\cdot 20 + 2.1\cdot 24 + 2.5\cdot 28.9\\ &\quad + 2.4\cdot 33.7 = 381.33.\end{aligned}$$

Note that Sidney-components (see Hamers et al., 2005) are not the same as merge segments. For this particular example, it can be checked that the order obtained in terms of Sidney-components would be as follows:

$$\hat{\tau} = (\underbrace{a_1a_2a_3a_4}_{S_1}\,\underbrace{b_1b_2}_{S_2}\,\underbrace{a_5}_{S_3}\,\underbrace{b_3b_4}_{S_4}). \qquad \triangle$$

Next, we will show a series of theoretical results that will allow us to prove the optimality of Algorithm 5.1.

Remark 5.1. Given a set $Z \subseteq N$, we will use the notation $\gamma[Z] = \sum_{i\in Z}\gamma(i)$ and $\alpha[Z] = \sum_{i\in Z}\alpha(i)$. Hence, if X is a segment we can write $CAT(X) = \frac{\gamma[X]}{\alpha[X]}$.

Proposition 5.1. Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem. Let X and Y be two segments that belong to the same branch, and let Z be a segment from the other branch. If $CAT(X) > CAT(Y)$, then

$$\tau = (\sim, X, Z, Y, \sim)$$

is not optimal.

Proof. Let us consider:

$$\tau_1 = (\sim, X, Y, Z, \sim);$$
$$\tau_2 = (\sim, Z, X, Y, \sim).$$

Note that

$$\begin{aligned} TC(\tau) - TC(\tau_1) &= \alpha[Y] \cdot \gamma[Z] - \alpha[Z] \cdot \gamma[Y] \\ &= \alpha[Y] \cdot \alpha[Z] \cdot \frac{\gamma[Z]}{\alpha[Z]} - \alpha[Z] \cdot \alpha[Y] \cdot \frac{\gamma[Y]}{\alpha[Y]} \\ &= \alpha[Y] \cdot \alpha[Z] \cdot (CAT(Z) - CAT(Y)), \end{aligned} \tag{5.2}$$

and

$$\begin{aligned} TC(\tau) - TC(\tau_2) &= \alpha[Z] \cdot \gamma[X] - \alpha[X] \cdot \gamma[Z] \\ &= \alpha[Z] \cdot \alpha[X] \cdot \frac{\gamma[X]}{\alpha[X]} - \alpha[X] \cdot \alpha[Z] \cdot \frac{\gamma[Z]}{\alpha[Z]} \\ &= \alpha[Z] \cdot \alpha[X] \cdot (CAT(X) - CAT(Z)). \end{aligned} \tag{5.3}$$

Suppose for the sake of contradiction that τ is optimal. Then, both τ_1 and τ_2 cannot have lower total costs than τ. This means that

$$TC(\tau) - TC(\tau_1) \leqslant 0 \quad \text{and} \quad TC(\tau) - TC(\tau_2) \leqslant 0.$$

From (5.2), this would imply that

$$CAT(Z) \leqslant CAT(Y); \tag{5.4}$$

while from (5.3), this would lead to

$$CAT(X) \leqslant CAT(Z). \tag{5.5}$$

By combining (5.4) and (5.5),

$$CAT(X) \leqslant CAT(Z) \leqslant CAT(Y),$$

leading to $CAT(X) \leqslant CAT(Y)$, which is a contradiction. Thus, τ cannot be optimal. □

Definition 5.7. Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for a $2-$lines GMS-problem. And let τ be a feasible order. Let $k \in \{1, 2, \ldots, m\}$. A **component of M_k in τ** is a maximal connected subset of M_k with respect to τ. Denote by M_k/τ the **set of components of M_k in τ**, and let $M_k/\tau = \{G_1, G_2, G_3, \ldots, G_{m_k}\}$ be the different

components in the order they appear in τ, that is:

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_3, \ldots, G_{m_k}, \sim).$$

Naturally, all merge segments and their components are segments.

Lemma 5.1. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for a $2-$lines GMS-problem. Take $k \in \{1, 2, \ldots, m\}$ such that $|M_k| > 1$. Let τ be a feasible order such that $\left|M_k/\tau\right| > 1$. Then, the following holds:*

i) $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1})$;

ii) $CAT(G_{m_k}) < CAT(M_k)$;

iii) $CAT(G_{m_k}) < CAT(G_1)$.

Proof. If $|M_k| > 1$, then M_k is not the first pivot of Algorithm 5.1. In order for M_k to become the new pivot and hence, the merge segment, it must have lower CAT than the previous pivot. Furthermore, the CAT of the previous pivot is less or equal to the CAT of the combination of the first $m_k - 1$ components of M_k. This might be a direct comparison, but it could also be an indirect comparison via several other pivots. That is,

$$CAT(M_k) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1}). \tag{5.6}$$

Then,

$$\begin{aligned} CAT(M_k) &= \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]} \\ &\underset{(5.6)}{<} \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]} \\ &= CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1}), \end{aligned}$$

which can be rewritten, by using cross-multiplication, to:

$$\begin{aligned} &(\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]) \cdot (\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]) \\ &< (\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]) \cdot (\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]) . \end{aligned}$$

Consequently,

$$\begin{aligned} &\gamma[G_{m_k}] \cdot (\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]) \\ &< (\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]) \cdot \alpha[G_{m_k}], \end{aligned} \tag{5.7}$$

and hence,

$$\frac{\gamma[G_{m_k}]}{\alpha[G_{m_k}]} < \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]}.$$

Thus, $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1})$, proving i).

To prove ii), we add $\gamma[G_{m_k}] \cdot \alpha[G_{m_k}]$ on both sides of equation (5.7):

$$\begin{aligned}&\gamma[G_{m_k}] \cdot (\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]) + \gamma[G_{m_k}] \cdot \alpha[G_{m_k}] \\ &< (\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]) \cdot \alpha[G_{m_k}] + \gamma[G_{m_k}] \cdot \alpha[G_{m_k}],\end{aligned}$$

which results in

$$\begin{aligned}&\gamma[G_{m_k}] \cdot (\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]) \\ &< (\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]) \cdot \alpha[G_{m_k}].\end{aligned}$$

Consequently,

$$\frac{\gamma[G_{m_k}]}{\alpha[G_{m_k}]} < \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]},$$

and thus $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k}) = CAT(M_k)$, proving ii).

To prove iii), note that $CAT(G_{m_k}) < CAT(M_k) < CAT(G_1)$, where the first inequality follows from ii) and the second inequality from Algorithm 5.1, see also equation (5.6). □

Lemma 5.2. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for a $2-$lines GMS-problem. Take $k \in \{1, 2, \ldots, m\}$ such that $|M_k| > 1$. And let τ be a feasible order such that $|M_k/\tau| > 1$. Then, there exists $k \in \{2, 3, \ldots, m_k\}$ such that $CAT(G_{k-1}) > CAT(G_k)$.*

Proof. Suppose for the sake of contradiction that $CAT(G_{k-1}) \leqslant CAT(G_k)$ for all $k \in \{2, 3, \ldots, m_k\}$. That is,

$$CAT(G_1) \leqslant CAT(G_2) \leqslant \cdots \leqslant CAT(G_{m_k-1}) \leqslant CAT(G_{m_k}).$$

This implies that $CAT(G_1) \leqslant CAT(G_{m_k})$, contradicting iii) of Lemma 5.1. Hence, there exists $k \in \{2, 3, \ldots, m_k\}$ such that $CAT(G_{k-1}) > CAT(G_k)$. □

Lemma 5.3. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for a $2-$lines GMS-problem. It holds that the elements of $M_k, k \in \{1, \ldots, m\}$, are consecutive in any optimal order.*

Proof. Let τ be an optimal order. Let us suppose that there exists $k \in \{1, \ldots, m\}$ such that players from M_k are separated by other players in τ. Let us also consider the set of

components of M_k in τ, i.e., $M_k/\tau = \{G_1, G_2, \ldots, G_{m_k}\}$, so we would have:

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_{m_k}, \sim).$$

From Lemma 5.2, we know that there exists $\tilde{k} \in \{2, \ldots, m_k\}$ such that $CAT(G_{\tilde{k}-1}) > CAT(G_{\tilde{k}})$. Let us rewrite τ as follows:

$$\tau = (\sim, G_{\tilde{k}-1}, Z, G_{\tilde{k}}, \sim),$$

where Z is a segment from the other branch. From Proposition 5.1, τ is not optimal, which is a contradiction. Hence, $\left|M_k/\tau\right| = 1$ for all $k \in \{1, \ldots, m\}$, that is, the nodes in M_k must be consecutive in τ. □

Proposition 5.2. Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem. Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for such problem. There always exists an optimal order that starts with M_1.

Proof. Let us consider $\tau \in \mathcal{F}(A \cup B)$ such that τ does not start with M_1. We will first prove that there always exists τ^* starting with M_1 such that $TC(\tau^*) \leqslant TC(\tau)$.

We distinguish between two cases: i) $|M_1/\tau| = 1$ and ii) $|M_1/\tau| > 1$. Let us assume w.l.o.g. that $M_1 \subseteq A$. During Algorithm 5.1, we compared the CAT of this pivot (which ended up being a merge segment) with the $CATs$ of all possible segments $R_{1l}, l \in \{1, \ldots, \tilde{t}\}$. It holds that:

$$\begin{aligned} CAT(M_1) &\leqslant CAT(R_{11}); \\ CAT(M_1) &\leqslant CAT(R_{12}); \\ &\vdots \\ CAT(M_1) &\leqslant CAT(R_{1\tilde{t}}). \end{aligned} \tag{5.8}$$

In the first case, M_1 is a connected component in τ. If τ does not start with M_1, then it must start with some players from branch B, followed by M_1:

$$\tau = (R_{1\ell}, M_1, \sim),$$

where $1 \leqslant \ell \leqslant \tilde{t}$. Now, consider the following order:

$$\tau_1 = (M_1, R_{1\ell}, \sim),$$

in which we have swapped the positions of M_1 and $R_{1\ell}$. Then,

$$\begin{aligned} TC(\tau) - TC(\tau_1) &= \alpha[M_1] \cdot \alpha[R_{1\ell}] \cdot CAT(R_{1\ell}) - \alpha[R_{1\ell}] \cdot \alpha[M_1] \cdot CAT(M_1) \\ &= \alpha[M_1] \cdot \alpha[R_{1\ell}] \cdot (CAT(R_{1\ell}) - CAT(M_1)) \underset{(5.8)}{\geqslant} 0, \end{aligned} \tag{5.9}$$

and hence, τ is not better than τ_1.

In the second case, if $|M_1/\tau| > 1$, then we can write τ as

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_3, \ldots, G_{m_1}, \sim),$$

with $m_1 > 1$, where $M_1/\tau = \{G_1, G_2, G_3, \ldots, G_{m_1}\}$ denotes the set of components of M_1 in τ. From Lemma 5.3, τ cannot be optimal since the elements of M_1 are not consecutive. In particular, we know there exists a bijection

$$\begin{aligned} \rho\colon \{1, 2, \ldots, m\} &\to \{1, 2, \ldots, m\} \\ i &\mapsto \rho(i) = j, \end{aligned}$$

such that

$$\tau' = (M_{\rho(1)}, \ldots, M_{\rho(m)})$$

is an optimal order. Take $\hat{i} \in \{1, \ldots, m\}$ such that $\rho(\hat{i}) = 1$. Let us consider

$$\tau^* = (M_{\rho(\hat{i})}, M_{\rho(1)}, M_{\rho(2)}, \ldots, M_{\rho(\hat{i}-1)}, M_{\rho(\hat{i}+1)}, \ldots, M_{\rho(m)}).$$

We know that

$$TC(\tau^*) \leqslant TC(\tau') < TC(\tau),$$

where the first inequality follows from (5.9) and the second inequality from Lemma 5.3. Thus, there always exists an optimal order starting with M_1. □

Proposition 5.3. Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem, and let $\tilde{\sigma}^N = (M_1, \ldots, M_m)$ be the output of Algorithm 5.1. Let τ^N be an optimal order. If τ^N starts with M_1, it holds that $\tau^N|_{N\setminus M_1}$ is an optimal order for the subproblem with $N\setminus M_1$ players.

Proof. W.l.o.g., let us assume that $M_1 \subseteq A$. If $M_1 = A$, then it is clear that $\tau^N = (M_1, b_1, b_2, \ldots, b_{\tilde{t}})$, where $\tau^N|_{N\setminus M_1} = (b_1, b_2, \ldots, b_{\tilde{t}})$ is the optimal order of the $1-$line GMS-problem with set of players $N\setminus M_1 = \{b_1, b_2, \ldots, b_{\tilde{t}}\}$. Let us suppose now that $M_1 \subsetneq A$. For the sake of contradiction, assume that $\tau^N|_{N\setminus M_1}$ is not an optimal order for the aforemen-

tioned subproblem. Then, there would exist $\hat{\tau}^{N\setminus M_1}$ such that

$$TC(\hat{\tau}^{N\setminus M_1}) - TC(\tau^N|_{N\setminus M_1}) < 0. \tag{5.10}$$

Let us consider the following order:

$$\hat{\tau}^N = (M_1, \hat{\tau}^{N\setminus M_1}).$$

Note that

$$TC(\hat{\tau}^N) - TC(\tau^N) = TC(\hat{\tau}^{N\setminus M_1}) - TC(\tau^N|_{N\setminus M_1}) \underset{(5.10)}{<} 0,$$

which is a contradiction because τ^N is optimal. Thus, $\tau^N|_{N\setminus M_1}$ is an optimal order for the problem with set of players $N\setminus M_1$. □

Lemma 5.4. *Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem, and let $\tilde{\sigma}^N = (M_1, \ldots, M_m)$ be the output of Algorithm 5.1. Let $\tau^{N\setminus M_1}$ be an optimal order for the problem with players set $N\setminus M_1$. It holds that $\tau^N = (M_1, \tau^{N\setminus M_1})$ is an optimal order for $(N, 0, E, \gamma, \alpha)$.*

Proof. From Proposition 5.2, we know there exists an optimal order $\hat{\tau}^N$ for $(N, 0, E, \gamma, \alpha)$ that starts with M_1, so $\hat{\tau}^N = (M_1, \hat{\tau}^N|_{N\setminus M_1})$. From Proposition 5.3, $\hat{\tau}^N|_{N\setminus M_1}$ is an optimal order for the subproblem with set of players $N\setminus M_1$. For the sake of contradiction, let us suppose that τ^N is not optimal. In such a case,

$$TC(\hat{\tau}^N) - TC(\tau^N) = TC(\hat{\tau}^N|_{N\setminus M_1}) - TC(\tau^{N\setminus M_1}) < 0,$$

which contradicts $\tau^{N\setminus M_1}$ being optimal for the problem with set of players $N\setminus M_1$. Thus, τ^N is an optimal order for $(N, 0, E, \gamma, \alpha)$. □

Theorem 5.8. *Let $(N, 0, E, \gamma, \alpha)$ be a $2-$lines GMS-problem, and let $\hat{\tau}$ be the order provided by Algorithm 5.1. Thus, $TC(\hat{\tau}) \leqslant TC(\tau)$ for all $\tau \in \mathcal{F}(A \cup B)$.*

Proof. In order to prove this, we will apply induction in the number of players, $|N|$.

Let us first suppose that $|N| = 2$. We present this situation in Figure 5.6. In such a case,

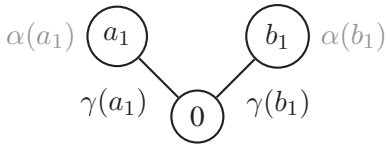


Figure 5.6: A $2-$lines GMS-problem with 2 players.

there are two possible orders, $\tau_1 = (a_1, b_1)$ and $\tau_2 = (b_1, a_1)$. Note that:

$$TC(\tau_1) = (\alpha(a_1) + \alpha(b_1)) \cdot \gamma(a_1) + \alpha(b_1) \cdot \gamma(b_1);$$
$$TC(\tau_2) = (\alpha(b_1) + \alpha(a_1)) \cdot \gamma(b_1) + \alpha(a_1) \cdot \gamma(a_1),$$

and thus

$$TC(\tau_1) - TC(\tau_2) = \alpha(b_1)\cdot\gamma(a_1) - \alpha(a_1)\cdot\gamma(b_1) = \alpha(b_1)\cdot\alpha(a_1)\cdot\left(\frac{\gamma(a_1)}{\alpha(a_1)} - \frac{\gamma(b_1)}{\alpha(b_1)}\right). \quad (5.11)$$

Algorithm 5.1 would compare $\frac{\gamma(a_1)}{\alpha(a_1)}$ to $\frac{\gamma(b_1)}{\alpha(b_1)}$ in order to choose the first merge segment, which in this case will consist of a single node. From (5.11), we can see that the optimal order will be determined by the exact same comparison, thus Algorithm 5.1 leads to an optimal order.

Now assume that Algorithm 5.1 leads to an optimal solution if the number of players is $k < |N|$.

Now, take $k = |N|$. Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 corresponding to $\hat{\tau}$. Naturally, $\hat{\sigma}|_{N\setminus M_1} = (M_2, \ldots, M_m)$ will be an output of our procedure for the problem with set of players $N\setminus M_1$. Using our induction hypothesis, $\hat{\sigma}|_{N\setminus M_1}$ is optimal for such subproblem. From Lemma 5.4, the order $(M_1, \hat{\sigma}|_{N\setminus M_1})$ is optimal. Clearly, $\hat{\sigma} = (M_1, \hat{\sigma}|_{N\setminus M_1})$, finishing the proof. □

5.3.2 The $n-$lines GMS-problem

A GMS-problem $(N, 0, E, \gamma, \alpha)$ is called an *$n-$lines GMS-problem* if there exists a partition $\langle A^1, \ldots, A^n\rangle$ of N with $A^k = \{a_1^k, \ldots, a_{\tilde{s}_k}^k\}$ for all $k \in \{1, \ldots, n\}$ with $\sum_{k=1}^n \tilde{s}_k = |N|$, such that

$$E = \bigcup_{k=1}^{n} \left\{\{0, a_1^k\}, \{a_1^k, a_2^k\}, \ldots, \{a_{\tilde{s}_k-1}^k, a_{\tilde{s}_k}^k\}\right\}.$$

As for the $2-$lines GMS-problems, sets $A^k, k \in \{1, \ldots, n\}$, are called branches. A feasible order is described by a bijection $\sigma\colon N \to \{1, 2, \ldots, |N|\}$ such that $\sigma(a_h^k) < \sigma(a_l^k) \Rightarrow h < l$, for all $k \in \{1, \ldots, n\}$. Let $\mathcal{F}(N)$ denote the set of all such feasible orders. A graphical representation of an $n-$lines GMS-problem can be seen in Figure 5.7.

Naturally, the $2-$lines GMS-problem introduced in Subsection 5.3.1 is a particular case of this problem. All the definitions regarding the segments and $CATs$ can be trivially extended to this generalization.

Given an $n-$lines GMS-problem, our objective is to obtain an optimal order. To this end, we have proposed an algorithm that combines the concept of recursion with Algo-

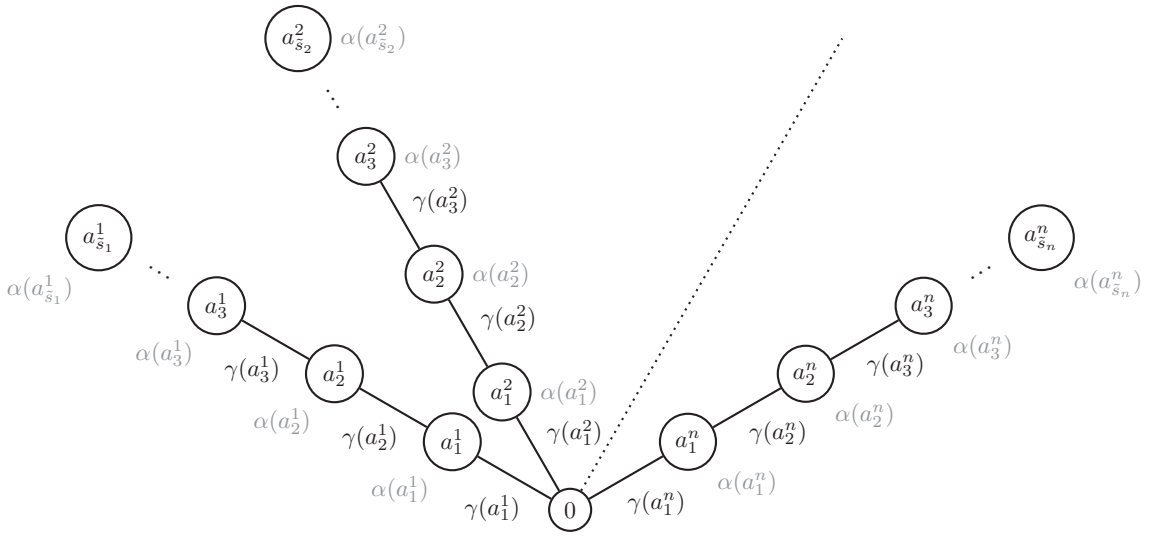


Figure 5.7: Graphical representation of an $n-$lines GMS-problem.

rithm 5.1. This procedure is based on the following idea: given $(N, 0, E, \gamma, \alpha)$ an $n-$lines GMS-problem, we can consider a $2-$lines GMS-problem, $(A^h \cup A^l, 0, E|_{A^h \cup A^l}, \gamma, \alpha)$, where $h, l \in \{1, \ldots, n\}$. Note that $E|_{A^h \cup A^l}$ is the restriction of E to the players of $A^h \cup A^l$. $\hat{\sigma}_{hl}$ will denote the output of Algorithm 5.1 for such a subproblem in which the merge segments are specified. $\hat{\tau}_{hl}$ will refer to the corresponding order on all players. Furthermore, $A^{hl}_{\hat{\tau}}$ will denote the branch formed by the nodes from A^h and A^l following the order specified by $\hat{\tau}_{hl}$. We informally present the proposed algorithm below.

Algorithm 5.2 Algorithm to solve an $n-$lines GMS-problem

1. Consider the $n-$lines GMS-problem $(N, 0, E, \gamma, \alpha)$. Select branches A^1 and A^2.
2. Apply Algorithm 5.1 to solve the corresponding $2-$lines GMS-problem, $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. This leads to an optimal order, $\hat{\tau}_{12}$. Replace A^1 and A^2 with the branch $A^{12}_{\hat{\tau}}$. We get a new problem with one branch less. Renumber the branches adequately.
3. (a) If there are still more than two branches left, go back to step 1.
   (b) If there are two branches left, apply Algorithm 5.1. The order obtained is the solution.

The following example illustrates how to apply Algorithm 5.2 to solve an $n-$lines GMS-problem.

Example 5.4. Let us consider the $3-$lines GMS-problem presented in Figure 5.8. We will avoid renumbering the nodes for a better understanding of the algorithmic process. Also, we will denote the branches by A, B, and C for clarity, instead of A^1, A^2, and A^3, respectively.

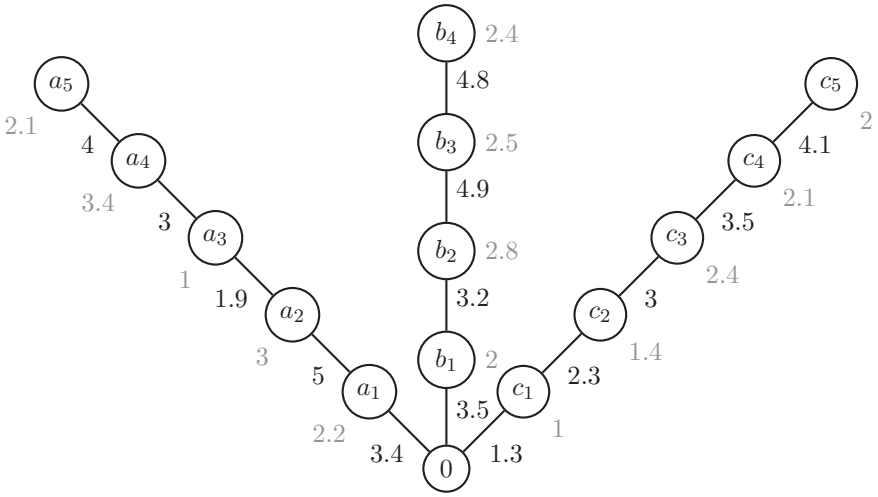


Figure 5.8: Example of a $3-$lines GMS-problem.

As indicated in Algorithm 5.2, we need to select branches A and B. The $2-$lines GMS-problem $(A \cup B, 0, E|_{A\cup B}, \gamma, \alpha)$ has been solved in Example 5.3, leading to the optimal order $\hat{\tau}_{12} = (a_1a_2a_3a_4b_1b_2a_5b_3b_4)$. We replace branches A and B with one branch respecting the order of $\hat{\tau}_{12}$, as illustrated in Figure 5.9.

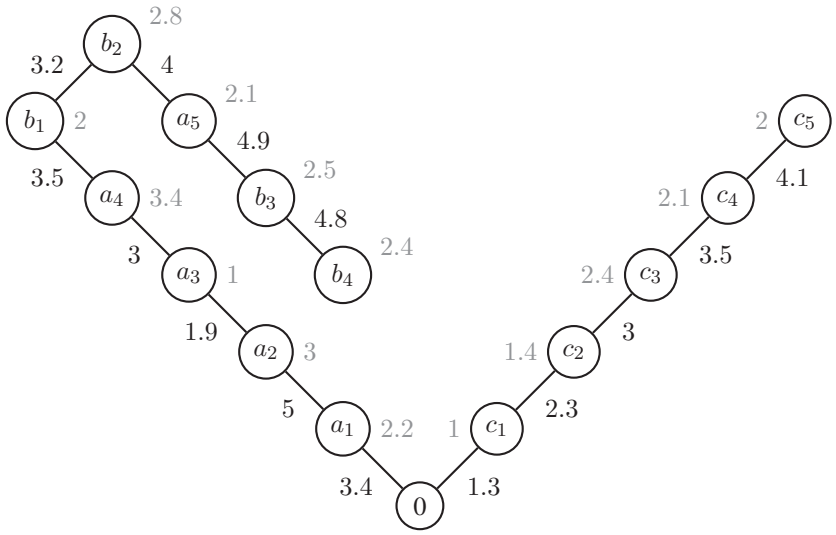


Figure 5.9: Resulting $2-$lines GMS-problem after solving $(A \cup B, 0, E|_{A\cup B}, \gamma, \alpha)$.

We need to solve now this $2-$lines GMS-problem by applying Algorithm 5.1. The output

would be:

$$\hat{\tau} = (c_1 a_1 a_2 a_3 a_4 c_2 c_3 b_1 b_2 c_4 a_5 b_3 b_4 c_5),$$

finishing Algorithm 5.2. △

In order to prove the optimality of Algorithm 5.2, we present the following results.

Lemma 5.5. *Let $(N, 0, E, \gamma, \alpha)$ be an $n-$lines GMS-problem, and $\hat{\sigma}_{12} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 5.1 for the $2-$lines GMS-problem $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. It holds that the elements of $M_k, k \in \{1, \ldots, m\}$, are consecutive in any optimal order for $(N, 0, E, \gamma, \alpha)$.*

Proof. See Proof of Lemma 5.3. □

Proposition 5.4. Let $(N, 0, E, \gamma, \alpha)$ be an $n-$lines GMS-problem, and let τ_{12}^* be the output of Algorithm 5.1 for the $2-$lines GMS-problem $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. There exists an optimal order $\hat{\tau}$ for $(N, 0, E, \gamma, \alpha)$ such that $\hat{\tau}(i) < \hat{\tau}(j)$ for all $i, j \in A^1 \cup A^2$ for which $\tau_{12}^*(i) < \tau_{12}^*(j)$.

Proof. Let $\sigma_{12}^* = (M_1, \ldots, M_m)$ be the output of Algorithm 5.1 for the $2-$lines GMS-problem $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$ that has τ_{12}^* as its associated order.

Let τ' be an optimal order for $(N, 0, E, \gamma, \alpha)$. From Lemma 5.5, we know that the elements of $M_k,\ k \in \{1, \ldots, m\}$, are consecutive in τ'. Let us assume that τ' does not respect the relative order induced by τ_{12}^*, and let $M_k, M_l,\ k, l \in \{1, \ldots, m\}$, be the first merge segments such that $\tau_{12}^*(M_k) > \tau_{12}^*(M_l)$ but $\tau'(M_k) < \tau'(M_l)$. Note that M_k and M_l necessarily belong to different branches (otherwise, τ' would not be feasible), and due to the way they have been chosen we also know that

$$CAT(M_l) \leqslant CAT(M_k). \tag{5.12}$$

Thus, between M_k and M_l in τ', players from all branches except that of M_l can be present. Let $M_k^1, M_k^2, \ldots, M_k^q$ be the maximal connected segments from the branch of M_k that are between M_k and M_l in τ', and let $Z^1, Z^2, \ldots, Z^q, Z^{q+1}$ be the (potential) maximal connected sets of nodes from $\mathcal{A} \backslash \{A^1 \cup A^2\}$ in τ'. There are four possible cases:

i) $\tau' = (\sim, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, Z^{q+1}, M_l, \sim)$,

ii) $\tau' = (\sim, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, M_l, \sim)$,

iii) $\tau' = (\sim, M_k, M_k^1, Z^1, M_k^2, \ldots, Z^{q-1}, M_k^q, Z^q, M_l, \sim)$,

iv) $\tau' = (\sim, M_k, M_k^1, Z^1, M_k^2, \ldots, Z^{q-1}, M_k^q, M_l, \sim)$.

We will prove the result for i) since the other cases are analogous. We will first show that:

$$\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} \leqslant CAT(M_k^{\tilde{q}}) \leqslant \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]}, \tag{5.13}$$

for all $\tilde{q} \in \{1, \ldots, q\}$. Let us suppose that (5.13) does not hold, that is, there exists $\tilde{q} \in \{1, \ldots, q\}$ such that

$$\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} > CAT(M_k^{\tilde{q}}), \tag{5.14}$$

or

$$CAT(M_k^{\tilde{q}}) > \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]}. \tag{5.15}$$

Let us take the order τ_1, which is a modification of τ' in which $Z^{\tilde{q}}$ and $M_k^{\tilde{q}}$ are swapped. In such a case,

$$TC(\tau') - TC(\tau_1) = \alpha[M_k^{\tilde{q}}] \cdot \alpha[Z^{\tilde{q}}] \cdot \left(\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} - CAT(M_k^{\tilde{q}}) \right) \underset{(5.14)}{>} 0,$$

which contradicts τ' from being optimal. Thus, (5.14) cannot hold. Analogously, take the order τ_2, which is a modification of τ' in which $M_k^{\tilde{q}}$ and $Z^{\tilde{q}+1}$ are swapped. In such a case,

$$TC(\tau') - TC(\tau_2) = \alpha[Z^{\tilde{q}+1}] \cdot \alpha[M_k^{\tilde{q}}] \cdot \left(CAT(M_k^{\tilde{q}}) - \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]} \right) \underset{(5.15)}{>} 0,$$

which contradicts τ' from being optimal. Thus, (5.15) cannot hold. This proves (5.13). Using similar arguments, it can be shown that

$$CAT(M_k) \leqslant \frac{\gamma[Z^1]}{\alpha[Z^1]} \quad \text{and} \quad \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} \leqslant CAT(M_l). \tag{5.16}$$

From (5.13) and (5.16), we have that

$$CAT(M_k) \leqslant \frac{\gamma[Z^1]}{\alpha[Z^1]} \leqslant CAT(M_k^1) \leqslant \cdots \leqslant \frac{\gamma[Z^q]}{\alpha[Z^q]} \leqslant CAT(M_k^q) \leqslant \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} \leqslant CAT(M_l). \tag{5.17}$$

By taking (5.12) and (5.17), we obtain that $CAT(M_k) = CAT(M_l)$, which in consequence leads to

$$CAT(M_k) = \frac{\gamma[Z^1]}{\alpha[Z^1]} = CAT(M_k^1) = \cdots = \frac{\gamma[Z^q]}{\alpha[Z^q]} = CAT(M_k^q) = \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} = CAT(M_l). \tag{5.18}$$

Let us now consider

$$\hat{\tau} = (\sim, M_l, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, Z^{q+1}, \sim),$$

which results from moving M_l to the front of M_k in τ'. Note that

$$\begin{aligned}
TC(\tau') - TC(\hat{\tau}) = \; & \alpha[M_l] \cdot \left(\alpha[M_k] \cdot CAT(M_k) + \sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] \cdot CAT(M_k^{\tilde{q}}) + \sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \cdot \frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} \right) \\
& - \alpha[M_l] \cdot \left(\alpha[M_k] + \sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] + \sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \right) \cdot CAT(M_l) \\
= \; & \alpha[M_l] \cdot \alpha[M_k] \cdot (CAT(M_k) - CAT(M_l)) \\
& + \alpha[M_l] \cdot \sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] \cdot (CAT(M_k^{\tilde{q}}) - CAT(M_l)) \\
& + \alpha[M_l] \cdot \sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \cdot \left(\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} - CAT(M_l) \right) \\
\underset{(5.18)}{=} \; & 0,
\end{aligned}$$

thus implying that $\hat{\tau}$ is an optimal order. □

The above result guarantees that in an $n-$lines GMS-problem, the first two branches can be replaced by one branch that maintains the relative order from solving the correspondent subproblem.

Theorem 5.9. *Let $(N, 0, E, \gamma, \alpha)$ be an $n-$lines GMS-problem, and let $\hat{\tau}$ be an order provided by Algorithm 5.2. Thus, $TC(\hat{\tau}) \leqslant TC(\tau)$ for all $\tau \in \mathcal{F}(N)$.*

Proof. We will prove this by induction in the number of branches, n.

If $n = 2$, Algorithm 5.2 coincides with Algorithm 5.1, which we know it leads to an optimal solution from Theorem 5.8.

Let us suppose that Algorithm 5.2 leads to an optimal solution for all $k < n$.

Now, take $k = n$. We can select branches A^1 and A^2 and apply Algorithm 5.1 to obtain a relative order, τ_{12}. From Proposition 5.4, there exists an optimal order $\hat{\tau}$ for $(N, 0, E, \gamma, \alpha)$ that maintains the order induced from branches A^1 and A^2. We can convert these two branches into one, A_τ^{12}, reducing the dimension of our problem by 1. Now we have an $(n-1)-$lines GMS-problem. By the induction hypothesis, Algorithm 5.2 leads to an optimal solution, τ^*. It is straightforward to prove that τ^* is also an optimal solution for the $n-$lines GMS-problem. Let us suppose it is not. In such a case, there would exist an optimal order $\tilde{\tau}$ such that $TC(\tilde{\tau}) < TC(\tau^*)$. Furthermore, considering that $\hat{\tau}$ is an optimal order for

$(N, 0, E, \gamma, \alpha)$ that maintains the induced relative order in branches A^1 and A^2, it will also be an optimal order for the $(n-1)-$lines GMS-problem in which A^1 and A^2 have been converted into A^{12}_τ. Thus, $TC(\hat{\tau}) = TC(\tilde{\tau}) < TC(\tau^*)$, contradicting the induction hypothesis. □

5.3.3 The tree GMS-problem

Once we have proven the optimality of Algorithm 5.2, let us now turn to trees. A GMS-problem $(N, 0, E, \gamma, \alpha)$ is called a *tree GMS-problem* if $(N \cup \{0\}, E)$ is a tree. Clearly, the problems we have seen above are particular cases of this one. The methodology proposed so far will be the basis for solving an optimization problem in these more general structures.

Definition 5.10. Let $(N, 0, E, \gamma, \alpha)$ be a tree GMS-problem. We define the **degree of a node** $a \in N$, $\deg(a)$, as the number of edges reaching that node.

Definition 5.11. Let $(N, 0, E, \gamma, \alpha)$ be a tree GMS-problem. A **sub-source** will be either the source, 0, or a node that is reached by three or more edges. Let $\mathcal{S}$ be the set of sub-sources,

$$\mathcal{S} = \{0\} \cup \{s \in N \mid \deg(s) \geqslant 3\}.$$

Given the sub-sources of a tree, we are interested in knowing their *level*.

Definition 5.12. Let $(N, 0, E, \gamma, \alpha)$ be a tree GMS-problem. The **level of a sub-source** $s \in \mathcal{S}$ is the number of sub-sources in the path between 0 and s, including 0. Thus, the sub-source 0 is the only sub-source with level 1. The level function, ℓ, is defined as follows:

$$\begin{aligned} \ell\colon \mathcal{S} &\to \mathbb{N} \\ s &\mapsto \ell(s) = |\{\text{sub-sources from } 0 \text{ to } s\} \cup \{0\}|. \end{aligned}$$

We assume an ordering on the sub-sources, from level 1 to the highest, v. Given a level $l \in \{2, \ldots, v\}$, there are m_l sub-sources. Thus, we can write

$$\mathcal{S} = \{0, s^2_1, \ldots, s^2_{m_1}, s^3_1, \ldots, s^3_{m_2}, \ldots, s^v_1, \ldots, s^v_{m_v}\},$$

where s^l_k denotes the $k-$th sub-source from level l. Figure 5.10 provides an illustration of the sub-sources of a tree and their levels.

That is, we have the set of sub-sources in increasing order by their level. The theoretical results seen for the $n-$lines GMS-problem can be extended to the general case of trees. In particular, given a tree GMS-problem, the elements of the merge segments obtained when

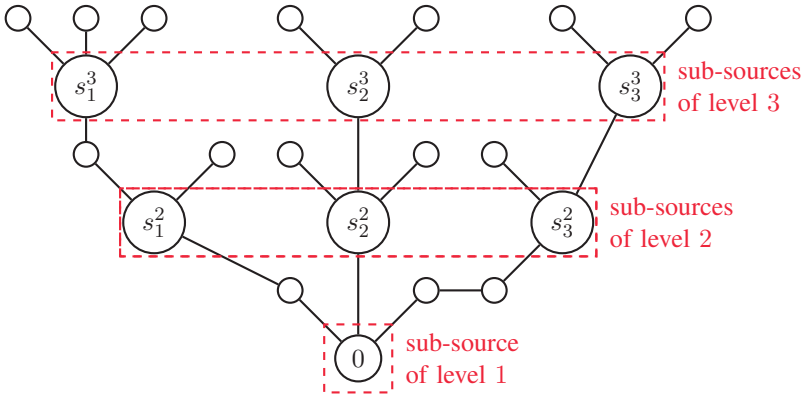


Figure 5.10: Sketch of the sub-sources of a tree and their levels.

solving a $2-$lines GMS-problem of the highest level remain consecutive in any optimal order. Furthermore, there always exists an optimal order that maintains the order induced by the aforementioned subproblem. With all these ingredients, it is immediate to prove the optimality of Algorithm 5.3, in which we present a brief outline of a proposed solution for the tree GMS-problem. As can be observed, a recursive methodology is adopted, starting with the $n-$lines GMS-problems at the highest level. For each of them, the $2-$lines GMS-problems that comprise it are solved recursively until these $n-$lines are converted into a single line, thus reducing the dimension. Precisely, an iteration of this algorithm consists in reducing by 1 the highest level of the problem. Therefore, in view of Algorithm 5.3, a tree GMS-problem can be solved in v iterations.

Algorithm 5.3 Algorithm to solve a tree GMS-problem

1. Consider the tree GMS-problem $(N, 0, E, \gamma, \alpha)$.
2. Repeat for all $k \in \{1, \ldots, m_v\}$ (i.e., the sub-sources from the highest level). Consider the n_k^v-lines GMS-problem arising from s_k^v. Apply Algorithm 5.2 to obtain $\tau_{s_k^v}$. Replace the n_k^v branches arising from s_k^v with one branch maintaining the order of $\tau_{s_k^v}$. Note that we have a new tree GMS-problem with one level less since all the subproblems of the highest level have been converted into lines. Renumber the nodes adequately. The highest level has now been reduced by 1.
3. (a) If $\mathcal{S} \neq \{0\}$, go back to step 1.
   (b) If $\mathcal{S} = \{0\}$, solve the resulting $n-$lines GMS-problem with Algorithm 5.2. The order obtained is the solution.

The following example shows how to solve a tree GMS-problem by applying Algorithm 5.3.

Example 5.5. Let us consider the tree GMS-problem presented in Figure 5.11.

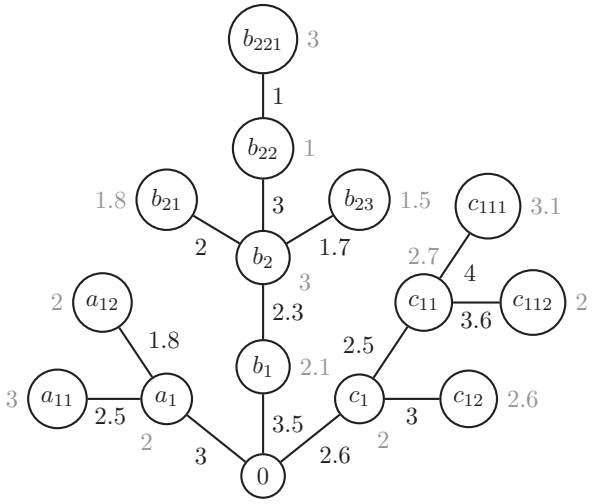


Figure 5.11: Example of a tree GMS-problem.

As can be seen, there are five sub-sources: 0, which is the only sub-source of level 1; a_1, b_2, and c_1, which are the sub-sources of level 2; and c_{11}, which is the sub-source of level 3, the highest level of this problem. Let us apply Algorithm 5.3 to solve it. First, we select c_{11}, which is the only sub-source of the highest level, and consider the $2-$lines GMS-problem arising from it, as shown in Figure 5.12. We apply Algorithm 5.1 and obtain $\tau_{c_{11}} = (c_{111}c_{112})$. We need to replace these two branches with one branch maintaining the order of $\tau_{c_{11}}$. Figure 5.13 shows the resulting tree GMS-problem, whose highest level is now reduced to 2.

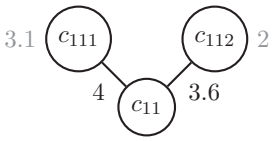


Figure 5.12: $2-$lines GMS-problem of level 3.

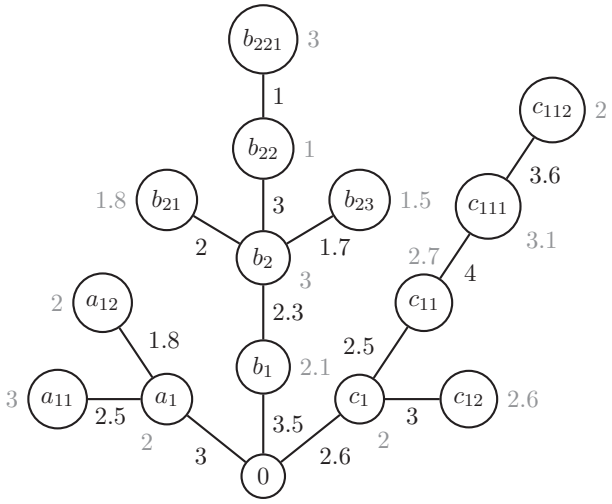


Figure 5.13: Resulting tree GMS-problem after the first iteration.

Now, we need to consider the $2,3,2-$lines GMS-problems arising from a_1, b_2, and c_1, respectively, which are the sub-sources of level 2. Figure 5.14 shows these problems.

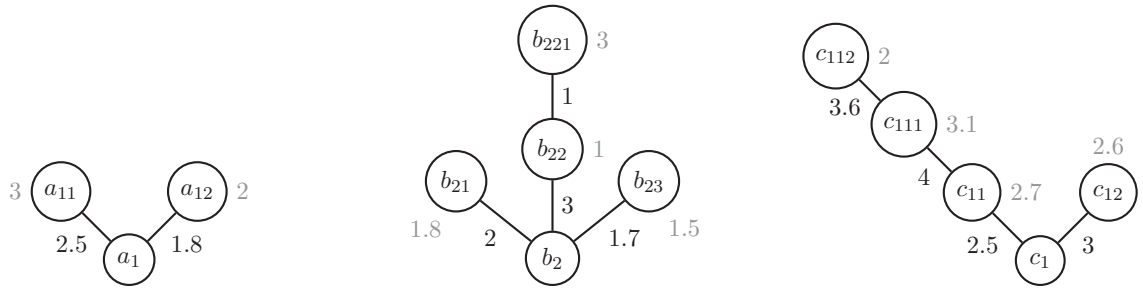


Figure 5.14: n-lines GMS-problems of level 2.

By applying Algorithm 5.2 to these problems, we obtain the orders $\tau_{a_1} = (a_{11}a_{12})$, $\tau_{b_2} = (b_{22}b_{221}b_{21}b_{23})$, and $\tau_{c_1} = (c_{11}c_{12}c_{111}c_{112})$. Thus, we replace each of these subproblems with one branch following the order specified by τ_{a_1}, τ_{b_2}, and τ_{c_1}, resulting in the $3-$lines GMS-problem of Figure 5.15.

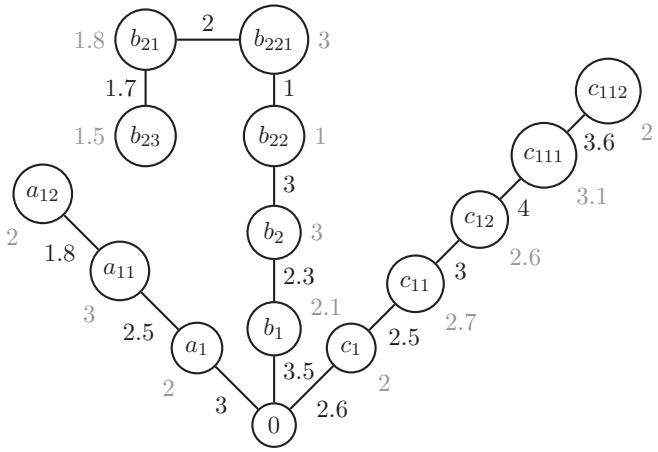


Figure 5.15: Resulting $3-$lines GMS-problem after the second iteration.

Now, apply Algorithm 5.2 to solve this problem. The result will be:

$$\hat{\tau} = (a_1a_{11}a_{12}b_1b_2b_{22}b_{221}c_1c_{11}b_{21}b_{23}c_{12}c_{111}c_{112}),$$

which is the output of Algorithm 5.3. △

Theorem 5.13. *Let $(N, 0, E, \gamma, \alpha)$ be a tree GMS-problem, and let $\hat{\tau}$ be an order provided by Algorithm 5.3. Thus, $TC(\hat{\tau}) \leqslant TC(\tau)$ for all feasible order τ.*

Proof. This proof can be done by induction on v, following similar arguments to those seen in the proof of Theorem 5.9. □

This result proves the optimality of Algorithm 5.3. As we have already anticipated, the study of the GMS-problem comprises two approaches: the first one, from the optimization point of view, has been treated in this chapter; and the second one, from the allocation perspective, will be addressed in Chapter 6.

Chapter 6

An allocation rule for GMS-problems

The main objective of this chapter is to introduce a cost allocation rule for the tree GMS-problem described in Chapter 5 to distribute the cost of an optimal order among the players involved. The allocation mechanism we are going to present follows an algorithmic approach, and it will be based on the solution algorithms presented in Section 5.3. We will consistently and recursively use myopic reference orders to determine potential cost savings, which will then be appropriately allocated by our method. Interestingly, the transition process from the reference order to the optimal one will be smooth using the switching of blocks of agents based on the notion of fine-tuned merge segments.

The contents of this chapter are partially collected in Davila-Pena et al. (2022a).

Contents

6.1 Introduction

When building a water supply network, cost minimization is the first problem to be addressed. Knowing in which order to optimally connect certain agents to a source, a second task consists in sharing the costs of this network among all the agents involved. The allocation of these costs is usually accomplished by proposing general allocation rules, which is the purpose of the present chapter for the GMS-problem.

The following section will introduce the κ rule as a cost allocation rule for tree GMS-problems. The κ rule takes as a reference point a myopic order and is closely tied to the algorithmic approaches presented in Chapter 5. In particular, the merge segments that appear when solving a $2-$lines GMS-problem will be the foundation of the cost sharing that we will

discuss. Below, we illustrate the calculation of the κ rule for Example 5.2 without dwelling on the details.

Example 6.1. Consider the GMS-problem of Figure 5.2. The reference order is (321), with an associated cost of 28. However, the optimal order is (213), with cost 25. Going from the reference order to the optimal one leads to a gain of 3 that needs to be split among players 1, 2, and 3 in a certain way. The myopic reference costs for these agents are $(12, 11, 5)$, respectively, and the κ rule will determine the compensations for each one. The first step is to identify the merge segments obtained by applying Algorithm 5.1, which are $M_1 = \{2, 1\}$ and $M_2 = \{3\}$. Note that these merge segments are misplaced in the reference order with respect to the optimal one. Thus, by switching them, the optimal order is obtained. The saving of 3 is divided equally between both merge segments, and proportionally to the number of agents within each of these blocks. Therefore, a saving of 1.5 goes for the agents in M_1 (0.75 for player 1 and 0.75 for player 2), and a saving of 1.5 goes for player 3 (the only player in M_2). Hence, the κ rule leads to the allocation $(11.25, 10.25, 3.5)$, resulting from subtracting the above gains from the reference costs.

The above example illustrates how to calculate the κ rule for a simple GMS-problem. Our method takes a myopic starting point and involves determining compensations. As shown in Example 6.1, one aspect that plays a fundamental role in determining these compensations is the switching of merge segments.[1] This is similar to the ideas behind the EGS-rule (Curiel et al., 1989). There are other features of the κ rule that arise when the agents of the merge segments are not consecutive in the reference order or when there are several sub-sources.

Next, we explain the ideas behind the κ rule in more detail. The order that we will use as a reference is an endogenous and myopic order since it will depend on the particular problem we are considering, in the following way: at each step, the machine selects the agent that has a higher urgency, always taking into account the existing precedence relations. To compute the κ rule we will follow a recursive procedure like the one in Algorithm 5.3, starting with the $n-$lines GMS-problems of the highest level. Here, a methodology akin to that of Algorithm 5.2 will be used, also recursively solving the $2-$lines GMS-problems that comprise it. The approach taken to address each of these $2-$lines GMS-problems starts from a reference order and "repairs" it until the optimal order found by Algorithm 5.1 is reached. We will see that it is not possible to always obtain non-negative savings when going from this reference order to the optimal one by simply swapping pairs of consecutive agents. Instead, we will need to exchange blocks of agents chosen in a certain way, and these blocks will be

[1] This was one of our primary motivations to modify the algorithm of Sidney (1975).

related to the merge segments. The theoretical results of Section 5.3 and Algorithm 5.1, play a key role in this task.

The remainder of this chapter is structured as follows. Section 6.2 describes in detail the procedure of obtaining the κ rule for tree GMS-problems, drawing on results seen in Chapter 5. Section 6.3 illustrates through several examples how to compute the κ rule on $2-$lines, $n-$lines, and general tree GMS-problems. Finally, Section 6.4 summarizes the main conclusions of this work.

6.2 The κ rule

Let $(N, 0, E, \gamma, \alpha)$ be a tree GMS-problem. The starting point of our proposal will be a reference order, τ_0^N, which is an endogenous myopic order that depends on the problem considered: at each step, the player with higher urgency is selected. For the sake of simplicity, we will assume that $\frac{\alpha(i)}{\gamma(i)} \neq \frac{\alpha(j)}{\gamma(j)}$ for all $i, j \in N, i \neq j$, i.e., all players' urgencies are different. Thus, τ_0^N is unique[2]. As already seen, Algorithm 5.3 provides an optimal order $\hat{\tau}^N$ for the problem, thus, the total amount that will be saved is $g^N = TC(\tau_0^N) - TC(\hat{\tau}^N)$. Also, recall that $c(\tau_0^N)$ is defined as the individual cost vector with respect to the reference order. This vector is from which we need to subtract the quantity g^N in order to obtain a cost allocation rule of the optimal order. Therefore, we need to know how to fairly subtract g^N from $c(\tau_0^N)$; that is, how responsible each of the players involved is for the resulting savings. Our allocation procedure will try to find a vector f^N whose coordinates indicate, for all possible players, the proportion of g^N for which each one is responsible. We will attempt from now on to determine this vector, for which purpose we will take into account what happens locally at each possible sub-source of our problem.

For $s \in \mathcal{S}$, we define $N_s = F(s) \cup \{s\}$, where $F(s)$ is the set of followers of s in the graph $(N \cup \{0\}, E)$. For every sub-source $s \in \mathcal{S}$, one considers an induced $n-$lines GMS-problem on N_s, $(N_s, 0, E|_{N_s}, \gamma, \alpha)$, where all initial branches with respect to s in E have been recursively replaced by a line that corresponds to an optimal order with respect to this branch. This optimal order is provided by Algorithm 5.3. Naturally, if $\ell(s) = v$, then $(N_s, 0, E|_{N_s}, \gamma, \alpha)$ is already an $n-$lines GMS-problem and we call it a subproblem from the highest level. Also, given $s \in \mathcal{S}$, $\tau_0^{N_s}$ and $\hat{\tau}^{N_s}$ represent the reference order and the optimal order provided by Algorithm 5.2 for $(N_s, 0, E|_{N_s}, \gamma, \alpha)$, respectively. Considering this, we determine the stand-alone cost savings, that is, the gains that can be saved when we do not

[2]In case of ties, any possible reference order is considered with a certain probability. Section 6.3 will elaborate on this issue.

take into account unrelated sub-sources to s:

$$w(s) := TC(\tau_0^{N_s}) - TC(\hat{\tau}^{N_s}) \geqslant 0.$$

Thus, $w(s)$ could be seen as local savings made at sub-sources. It should be noted that the sum of them is not equal to the total savings we may have, that is, $\sum_{s\in\mathcal{S}} w(s) \neq g^N$. However, we will be able to use these numbers to infer the importance of the different sub-sources (and, mainly, of the players involved in the problems they induce) in the final savings obtained.

As already mentioned, an induced $n-$lines GMS-problem is tackled on N_s. To obtain our cost allocation rule we will follow a recursive procedure, by first solving the subproblems of the highest level. Once these problems have been solved, the highest level of the tree GMS-problem has been reduced by 1, and we repeat the process. Hence, we will always start from an $n-$lines GMS-problem, for which we will consider a branch-order, π^s, that will be determined by the reference order $\tau_0^{N_s}$ in the following way: the branches are selected in the same order as they appear in $\tau_0^{N_s}$. A branch-order simply indicates the numbering of each branch assuming we originally labeled them as $A^1, \ldots, A^n$. For instance, if $n = 3$ and $\pi^s = (2, 3, 1)$, then the first, second, and third branches are A^2, A^3, and A^1, respectively, for all applicable purposes. Let us assume we are in an $n-$lines GMS-problem $(N_s, 0, E|_{N_s}, \gamma, \alpha)$ such that $\ell(s) = v$. What we first do is to obtain the reference order, $\tau_0^{N_s}$. As already stated, this induces a branch order, π^s. We will denote the (ordered with respect to π^s) branches by $A^{1,\pi^s}, \ldots, A^{n,\pi^s}$. Thus, we start by selecting branches A^{1,π^s} and A^{2,π^s} and obtain the reference order as well as apply Algorithm 5.1 to get an optimal order of this particular $2-$lines GMS-problem. This results in local savings (for the considered sub-source), which we must distribute among the players involved. Once these savings have been allocated, we follow the same recursive approach as in Algorithm 5.2: convert branches A^{1,π^s} and A^{2,π^s} into one, A^{12,π^s}, and apply Algorithm 5.1 to the $2-$lines GMS-problem induced by A^{12,π^s} and A^{3,π^s}. Again, for this problem, we obtain the reference order as well as the optimal order provided by Algorithm 5.1 and allocate the gains achieved, and so on. To facilitate the understanding of the general cost allocation procedure, we will later elaborate on the method by which the resulting savings from solving a $2-$lines GMS-problem are obtained. Furthermore, to avoid making the notation cumbersome, we will henceforth omit to indicate the sub-source we are considering, but note that we have fixed one. Given π (former π^s), there will exist $\tau_0^{2,\pi}, \ldots, \tau_0^{n,\pi}$, the reference orders obtained when recursively considered $2-$lines GMS-problems in the order specified by π. For instance, if $n = 3$ and $\pi = (2, 3, 1)$, then $\tau_0^{2,\pi}$ is the reference order for the $2-$lines GMS-problem $(A^2 \cup A^3, 0, E|_{A^2\cup A^3}, \gamma, \alpha)$, and $\tau_0^{3,\pi}$ will be the reference order for the $2-$lines GMS-problem $(A^{23} \cup A^1, 0, E|_{A^{23}\cup A^1}, \gamma, \alpha)$. That

is, given $k \in \{2, \ldots, n\}$, $\tau_0^{k,\pi}$ represents the reference order of the $2-$lines GMS-problem induced by branches $A^{1\ldots k,\pi}$ and $A^{k,\pi}$. Also, $\hat{\tau}^{2,\pi}, \ldots, \hat{\tau}^{n,\pi}$ and $\hat{\sigma}^{2,\pi}, \ldots, \hat{\sigma}^{n,\pi}$ represent the optimal orders and their corresponding merge orders provided by Algorithm 5.1 for the aforementioned $2-$lines GMS-problems. Note that $\hat{\tau}^{n,\pi}$ coincides with $\hat{\tau}^{N_s}$. The stand-alone cost savings are defined by $g^{k,\pi} = TC(\tau_0^{k,\pi}) - TC(\hat{\tau}^{k,\pi})$ for each $k \in \{2, \ldots, n\}$. We also define $g^{\pi}(s) = \sum_{k=2}^{n} g^{k,\pi}$. Furthermore, $f^{2,\pi}, \ldots, f^{n,\pi}$ represent the *block splitting fraction vectors* that correspond to the normalized *block splitting rule* (BSR), complemented with 0's on those coordinates that refer to non-involved players. That is, for all $k \in \{2, \ldots, n\}$,

$$f^{k,\pi} = \frac{1}{g^{k,\pi}} \cdot \text{BSR}(\tau_0^{k,\pi}, \hat{\sigma}^{k,\pi}),$$

where $\text{BSR}(\tau_0^{k,\pi}, \hat{\sigma}^{k,\pi})$ represents the savings allocation vector obtained when considering the $2-$lines GMS-problem induced by branches $A^{1\ldots k,\pi}$ and $A^{k,\pi}$. Thus, it provides an allocation of $g^{k,\pi}$, i.e., the gains in going from the reference order $\tau_0^{k,\pi}$ to the optimal order $\hat{\sigma}^{k,\pi}$. This vector will be calculated consistently in the same way, and it will be explained at the end of the general procedure as a separate tool. Thus, we determine

$$f^{\pi}(s) = \sum_{k=2}^{n} \left[\frac{g^{k,\pi}}{g^{\pi}(s)} \right] \cdot f^{k,\pi},$$

the weighted average over all block splitting fraction vectors $f^{k,\pi}$. Therefore, we define the cost allocation rule, κ, as follows:

$$\kappa = c(\tau_0^N) - \left(TC(\tau_0^N) - TC(\hat{\tau}^N)\right) \cdot f^N,$$

where

$$f^N = \sum_{s \in \mathcal{S}} \left[\frac{w(s)}{\sum_{t \in \mathcal{S}} w(t)} \right] \cdot f^{\pi}(s).$$

Note that for an $n-$lines GMS-problem, $\mathcal{S} = \{0\}$, thus $f^N = f^{\pi}(0)$. Also, note that, although we have decided to omit the s in π, it is included in $g^{\pi}(s)$ and $f^{\pi}(s)$ for clarity, to emphasize that these are local savings and allocations, respectively, in the sub-source s.

Once the general allocation procedure has been presented, we will provide a $2-$lines example that will allow us to introduce how the allocation process is performed in this case.

Example 6.2. Let us consider the $2-$lines GMS-problem from Example 5.3. In order to compute the reference order, we start by comparing $\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{3.4}{2.2} = 1.55$ to $\frac{\gamma(b_1)}{\alpha(b_1)} = \frac{3.5}{2} = 1.75$. Since the urgency of player a_1 is higher, we select a_1 as the first player of τ_0^N. The second

step consists in comparing $\frac{\gamma(a_2)}{\alpha(a_2)} = \frac{5}{3} = 1.67$ to $\frac{\gamma(b_1)}{\alpha(b_1)} = \frac{3.5}{2} = 1.75$, leading to a_2 being the second player in τ_0^N. We repeat these comparisons until all players have been included in τ_0^N. The reference order is $\tau_0^N = (a_1a_2b_1b_2a_3a_4a_5b_3b_4)$, with $TC(\tau_0^N) = 387.29$. Note that, in this case, the branch-order induced by τ_0^N is $\pi = (1, 2)$. However, since there are only two branches and no ties, considering $\pi = (1, 2)$ or $\pi = (2, 1)$ would have led to the same result. We have seen in Example 5.3 that $\hat{\tau}^N = (a_1a_2a_3a_4b_1b_2a_5b_3b_4)$ is an optimal order, with $TC(\hat{\tau}^N) = 381.33$. Thus, τ_0^N is not optimal, and there is a saving of $g^N = 387.29 - 381.33 = 5.96$ from τ_0^N to $\hat{\tau}^N$. △

The above example raises the following questions: how can we allocate the savings of going from τ_0^N to $\hat{\tau}^N$? Which players should be compensated for such savings? Because of how τ_0^N is constructed, we know that we cannot switch b_2 with a_3 (the first point at which we have two consecutive players from different branches that are misplaced with respect to the optimal order) since this leads to a negative switch: in the reference order b_2 goes before a_3, this means that b_2 has a higher urgency than a_3. But we can switch some consecutive groups of players simultaneously, by blocks. In this particular case, we could switch b_1b_2 with a_3a_4, thus obtaining the optimal order. But, how do we choose these blocks? The determination of these blocks cannot be carried out simply by observing the orders τ_0^N and $\hat{\tau}^N$, but will be done iteratively. For this, we must take into account some of the theoretical results seen in Subsection 5.3.1. Also, note that given two consecutive misplaced blocks, X and Y, the gain resulted from switching them is:

$$\begin{aligned} g_{XY} &= \alpha[Y] \cdot \gamma[X] - \alpha[X] \cdot \gamma[Y] \\ &= \alpha[Y] \cdot \alpha[X] \cdot \frac{\gamma[X]}{\alpha[X]} - \alpha[X] \cdot \alpha[Y] \cdot \frac{\gamma[Y]}{\alpha[Y]} \\ &= \alpha[Y] \cdot \alpha[X] \cdot (CAT(X) - CAT(Y)) \,. \end{aligned}$$

Regarding a savings allocation rule, a possibility could be to allocate $\frac{1}{2}g_{XY}$ to players from X (equally among them) and $\frac{1}{2}g_{XY}$ to players from Y (equally among them), as long as it is guaranteed that g_{XY} is positive. As stated above, our objective will be to go from τ_0^N to $\hat{\tau}^N$ by having non-negative gains at each step. To do that, we will present in detail the aforementioned *block splitting rule* (BSR), which is constructed through an iterative procedure. The key point of this approach is to determine, at each step, which blocks are to be swapped. It is here where the merge segments play a fundamental role since by conveniently using their properties along with Algorithm 5.1 we will be able to guarantee non-negative savings at each iteration. Thus, this procedure consists of two main stages: firstly, we will repair those merge segments whose players are not consecutive, and secondly reorder them as in $\hat{\tau}^N$. To

this end, we will need to consider $\hat{\sigma}^N$, the corresponding merge order to $\hat{\tau}^N$. Algorithm 6.1 shows the scheme of this procedure.

Algorithm 6.1 Algorithm to allocate the gains of a $2-$lines GMS-problem

0. Obtain τ_0^N and apply Algorithm 5.1 to get $\hat{\sigma}^N = (M_1, M_2, \ldots, M_m)$. Initialize $k = 1$, $r = 1$, it $= 1$, and $\tau' = \tau_0^N$.

1. (a) If $|M_k/\tau'| = 1$, take $k = k + 1$.
   - If $k \leqslant m - 1$, go back to step 1.
   - If $k = m$, go to step 2.

   (b) If $|M_k/\tau'| > 1$, take $\tilde{k} \in \{2, \ldots, m_k\}$ such that $CAT(G_{\tilde{k}-1}) > CAT(G_{\tilde{k}})$ (we know there exists such a pair of components from Lemma 5.2). It is clear that between $G_{\tilde{k}-1}$ and $G_{\tilde{k}}$ there are only players from the other branch, that is

   $$\tau' = (\sim, G_{\tilde{k}-1}, Z, G_{\tilde{k}}, \sim),$$

   where Z is a segment from the opposite branch of M_k. From Proposition 5.1, we know that either $\tau_1 = (\sim, G_{\tilde{k}-1}, G_{\tilde{k}}, Z, \sim)$ or $\tau_2 = (\sim, Z, G_{\tilde{k}-1}, G_{\tilde{k}}, \sim)$ has a lower total cost than τ'. Take $\tau'' = \arg\min_\tau \{TC(\tau) \colon \tau \in \{\tau_1, \tau_2\}\}$.
   - If $\tau'' = \tau_1$, then the blocks that have been switched are Z and $G_{\tilde{k}}$. Players from Z should receive $\frac{1}{2|Z|} g_{ZG_{\tilde{k}}}$, while players from $G_{\tilde{k}}$ should receive $\frac{1}{2|G_{\tilde{k}}|} g_{ZG_{\tilde{k}}}$.
   - If $\tau'' = \tau_2$, then the blocks that have been switched are $G_{\tilde{k}-1}$ and Z. Players from $G_{\tilde{k}-1}$ should receive $\frac{1}{2|G_{\tilde{k}-1}|} g_{G_{\tilde{k}-1}Z}$, while players from Z should receive $\frac{1}{2|Z|} g_{G_{\tilde{k}-1}Z}$.

   Set $\tau' = \tau''$, and take it $=$ it $+ 1$. Go back to step 1.

2. (a) If $r \leqslant m$, consider the bijection

   $$\begin{aligned} \rho \colon \{1, 2, \ldots, m\} &\to \{1, 2, \ldots, m\} \\ i &\mapsto \rho(i) = j, \end{aligned}$$

   such that $\tau' = (M_{\rho(1)}, \ldots, M_{\rho(m)})$. We need to go from τ' to $\hat{\tau}$.

   i. If $\rho(r) = r$, take $r = r + 1$. Go back to step 2.

   ii. If $\rho(r) \neq r$, take $\tilde{r} \in \{r + 1, \ldots, m\}$ such that $\rho(\tilde{r}) = r$ (this means that M_r is on position $\tilde{r}$, i.e., $M_r = M_{\rho(\tilde{r})}$). By Algorithm 5.1, it holds that

   $$CAT(M_r) \leqslant AT\left(M_r^{\cup}\right),$$

   where $M_r^{\cup} = \bigcup_{l=r}^{\tilde{r}-1} M_{\rho(l)}$. Hence, the order

   $$\tau'' = (\sim, M_{\rho(\tilde{r})}, M_{\rho(r)}, \ldots, M_{\rho(\tilde{r}-1)}, M_{\rho(\tilde{r}+1)}, \sim)$$

   that consists in moving $M_{\rho(\tilde{r})}$ to the front of $M_{\rho(r)}$ (so that $M_{\rho(\tilde{r})} \equiv M_r$ is now on position r) has lower total cost than τ'. The blocks that have been switched are $M_r^{\cup}$ and M_r. Allocate $\frac{1}{2|M_r^{\cup}|} g_{M_r^{\cup} M_r}$ to the players in $M_r^{\cup}$ and $\frac{1}{2|M_r|} g_{M_r^{\cup} M_r}$ to the players in M_r. Set $\tau' = \tau''$, and take $r = r + 1$ and it $=$ it $+ 1$. Go back to step 2.

   (b) If $r > m$, then $\tau' = \hat{\tau}$ and $I =$ it $- 1$. The algorithm is finished. The outputs of the algorithm are the allocation and the misplaced blocks at each iteration it $\in \{1, \ldots, I\}$.

Let X^{it} and Y^{it} be the misplaced blocks switched at iteration it of Algorithm 6.1. W.l.o.g., let us assume that $X^{\text{it}} \subseteq A$ and $Y^{\text{it}} \subseteq B$. Therefore, the BSR can be formulated as:

$$\text{BSR} = \sum_{\text{it}=1}^{I} \frac{1}{2} g_{X^{\text{it}}Y^{\text{it}}} \left[\left(\frac{1}{|X^{\text{it}}|} e^{X^{\text{it}}}, \mathbf{0}_{\tilde{t}} \right) + \left(\mathbf{0}_{\tilde{s}}, \frac{1}{|Y^{\text{it}}|} e^{Y^{\text{it}}} \right) \right],$$

where $e^{X^{\text{it}}}$ and $e^{Y^{\text{it}}}$ are vectors of length $\tilde{s}$ and $\tilde{t}$ (the number of players in branches A and B, respectively) satisfying that $e^{X^{\text{it}}}_{k'} = 1$ if $k' \in \{1, \ldots, \tilde{s}\}$ such that $a_{k'} \in X^{\text{it}}$, and 0 otherwise; and $e^{Y^{\text{it}}}_{k} = 1$ if $k \in \{1, \ldots, \tilde{t}\}$ such that $b_k \in Y^{\text{it}}$, and 0 otherwise. Let BSR^{it} represent the allocation obtained at iteration it, hence $\text{BSR} = \sum_{\text{it}=1}^{I} \text{BSR}^{\text{it}}$.

As already mentioned, this procedure needs to be conducted every time we solve a $2-$lines GMS-problem in order to allocate the savings in going from the reference order to the optimal order provided by Algorithm 5.1. Therefore, and having into account the recursive methodology explained above for tree GMS-problems, this block splitting rule will be constantly computed and integrated throughout the calculation of the κ rule.

6.3 Examples

In this section, we will illustrate how to calculate the κ rule for each of the problems detailed in Chapter 5. To avoid overly tedious calculations and facilitate understanding of the methodology, we have assumed that all agents involved have an associated cost parameter equal to 1. In addition, at the end of this section, we will briefly explain how to proceed in case there are agents with the same urgency, an issue that was not covered when explaining the κ rule computation in the previous section. An example will be presented for illustrative purposes.

The following example shows how to obtain the cost allocation rule for a $2-$lines GMS-problem by applying the above procedure.

Example 6.3. Let us consider the $2-$lines GMS-problem from Figure 6.1. Note that $\alpha(i) = 1$ for all $i \in A \cup B$, so we decided not to incorporate them in the figure for clarity. The reference order would be

$$\tau_0^N = (b_1 b_2 b_3 b_4 a_1 a_2 b_5 a_3 a_4 a_5),$$

with an associated cost of $TC(\tau_0^N) = 992$. The induced branch-order is $\pi = (2, 1)$. However, given that there are no ties and only 2 branches, considering $\pi = (1, 2)$ or $\pi = (2, 1)$ would lead to the same result. The optimal solution provided by Algorithm 5.1 would be

$$\hat{\tau}^N = (b_1 a_1 a_2 a_3 a_4 b_2 b_3 b_4 a_5 b_5),$$

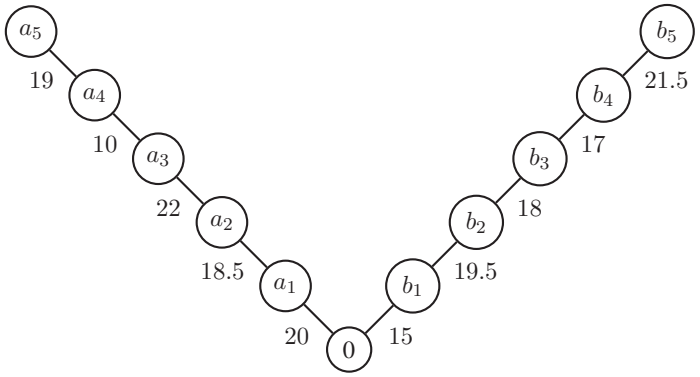


Figure 6.1: Example of a $2-$lines GMS-problem.

with an associated cost of $TC(\hat{\tau}^N) = 972$. The corresponding merge order to $\hat{\tau}^N$ is the following:

$$\hat{\sigma}^N = (\underbrace{b_1}_{M_1}\ \underbrace{a_1a_2a_3a_4}_{M_2}\ \underbrace{b_2b_3}_{M_3}\ \underbrace{b_4}_{M_4}\ \underbrace{a_5}_{M_5}\ \underbrace{b_5}_{M_6}).$$

The first merge segment that is divided in τ_0^N is M_2. In fact, $M_2/\tau_0^N = \{G_1, G_2\}$, where $G_1 = \{a_1, a_2\}$ and $G_2 = \{a_3, a_4\}$. These components have $Z = \{b_5\}$ in between. By Lemma 5.1, we know that $CAT(G_1) > CAT(G_2)$. By Proposition 5.1, we know that either moving G_2 to the front of Z or moving G_1 to the back of Z would lead to positive savings. Thus, we need to compare the following two orders:

$$\tau_1 = (b_1b_2b_3b_4a_1a_2\underbrace{a_3a_4}_{G_2}\overbrace{b_5}^{Z}a_5) \quad \text{and} \quad \tau_2 = (b_1b_2b_3b_4\overbrace{b_5}^{Z}\underbrace{a_1a_2}_{G_1}a_3a_4a_5).$$

Since $TC(\tau_1) = 981$ and $TC(\tau_2) = 996.5$, we choose $\tau' = \tau_1$. Hence, blocks Z and G_2 have been switched. The gain resulted from this swap is $g_{ZG_2} = |G_2| \cdot |Z| \cdot (CAT(Z) - CAT(G_2)) = 2 \cdot 1 \cdot \left(\frac{21.5}{1} - \frac{22+10}{1+1}\right) = 11$. Thus, we need to allocate $\frac{1}{2\cdot 2} \cdot 11 = 2.75$ for players a_3 and a_4, and $\frac{1}{1\cdot 1} \cdot 11 = 5.5$ for player b_5. For this first iteration, we have that:

$$\text{BSR}^1(\tau_0^N, \hat{\sigma}^N) = (0, 0, 2.75, 2.75, 0, 0, 0, 0, 0, 5.5).$$

Note that τ' can be expressed as follows:

$$\tau' = (\underbrace{b_1}_{M_1}\ \underbrace{b_2b_3}_{M_3}\ \underbrace{b_4}_{M_4}\ \underbrace{a_1a_2a_3a_4}_{M_2}\ \underbrace{b_5}_{M_6}\ \underbrace{a_5}_{M_5}).$$

Now that all the merge segments are together, we need to reorder them. The first difference in

τ' with respect to $\hat{\tau}^N$ occurs in M_3 since M_2 should go before. By Algorithm 5.1, we know that $CAT(M_2) \leqslant CAT(M_2^{\cup})$, where $M_2^{\cup} = M_3 \cup M_4$. Thus, we consider the following order, consisting in moving M_2 to the front of M_3, that is, switching $M_2^{\cup}$ and M_2:

$$\tau'' = (\underbrace{b_1}_{M_1}\ \underbrace{a_1a_2a_3a_4}_{M_2}\ \underbrace{b_2b_3}_{M_3}\ \underbrace{b_4}_{M_4}\ \underbrace{b_5}_{M_6}\ \underbrace{a_5}_{M_5}).$$

We need to allocate $g_{M_2^{\cup}M_2} = |M_2| \cdot |M_2^{\cup}| \cdot (CAT(M_2^{\cup}) - CAT(M_2)) = 4 \cdot 3 \cdot \left(\frac{19.5+18+17}{1+1+1} - \frac{20+18.5+22+10}{1+1+1+1}\right) = 6.5$ among the players in $M_2^{\cup}$ and M_2. In particular, we need to allocate $\frac{1}{2\cdot 3} \cdot 6.5 = 1.083$ for players b_2, b_3, and b_4; and $\frac{1}{2\cdot 4} \cdot 6.5 = 0.813$ for players a_1, a_2, a_3, and a_4. This leads to:

$$\text{BSR}^2(\tau_0^N, \hat{\sigma}^N) = (0.813, 0.813, 0.813, 0.813, 0, 0, 1.083, 1.083, 1.083, 0).$$

We still need to reorder M_5 and M_6. This leads to $\hat{\tau}$. We have thus to allocate $g_{M_6M_5} = |M_5| \cdot |M_6| \cdot (CAT(M_6) - CAT(M_5)) = 1 \cdot 1 \cdot \left(\frac{21.5}{1} - \frac{19}{1}\right) = 2.5$ among the players in M_5 and M_6. In particular, we allocate $\frac{1}{2\cdot 1} \cdot 2.5 = 1.25$ for a_5 and $\frac{1}{2\cdot 1} \cdot 2.5 = 1.25$ for b_5. The end of iteration 3 results in

$$\text{BSR}^3(\tau_0^N, \hat{\sigma}^N) = (0, 0, 0, 0, 1.25, 0, 0, 0, 0, 1.25).$$

Therefore, the final savings allocation rule provided by Algorithm 6.1 is the following:

$$\text{BSR}(\tau_0^N, \hat{\sigma}^N) = \textstyle\sum_{\text{it}=1}^{3} \text{BSR}^{\text{it}}(\tau_0^N, \hat{\sigma}^N) = (0.813, 0.813, 3.563, 3.563, 1.25, 0, 1.083, 1.083, 1.083, 6.75).$$

Since $n = 2$, we have that $\tau_0^{2,\pi} = \tau_0^N$, $\hat{\tau}^{2,\pi} = \hat{\tau}^N$, $\hat{\sigma}^{2,\pi} = \hat{\sigma}^N$, $g^{2,\pi} = 20$, $g^{\pi}(0) = 20$, $f^{2,\pi} = \frac{1}{20} \cdot \text{BSR}(\tau_0^{2,\pi}, \hat{\sigma}^{2,\pi})$, and $f^N = f^{\pi}(0) = f^{2,\pi}$. Hence,

$$\begin{aligned}
\kappa &= c(\tau_0^N) - \left(TC(\tau_0^N) - TC(\hat{\tau}^N)\right) \cdot f^N \\
&= (89.5, 108, 151.5, 161.5, 180.5, 15, 34.5, 52.5, 69.5, 129.5) \\
&\qquad - 20 \cdot \frac{1}{20} \cdot (0.813, 0.813, 3.563, 3.563, 1.25, 0, 1.083, 1.083, 1.083, 6.75) \\
&= (88.687, 107.187, 147.937, 157.937, 179.25, 15, 33.417, 51.417, 68.417, 122.75). \qquad \triangle
\end{aligned}$$

The following example gives a detailed explanation of how to obtain the cost allocation rule for an $n-$lines GMS-problem.

Example 6.4. Let us consider the $3-$lines GMS-problem from Figure 6.2. The reference

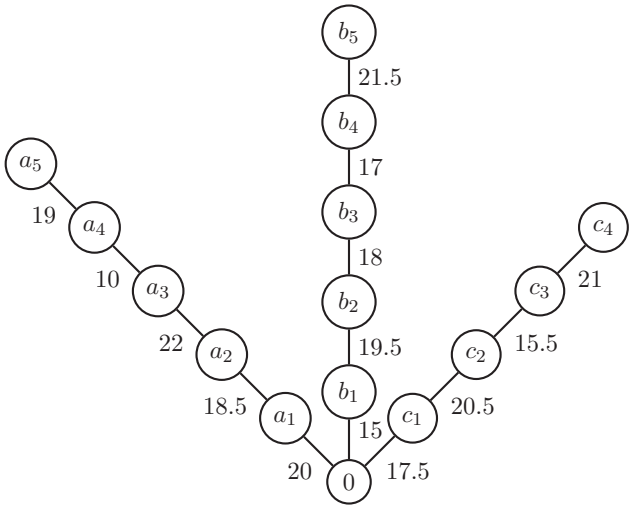


Figure 6.2: Example of a $3-$lines GMS-problem.

order for all players N is:

$$\tau_0^N = (b_1c_1b_2b_3b_4a_1a_2c_2c_3c_4b_5a_3a_4a_5),$$

with $TC(\tau_0^N) = 1900$. Also, we know that

$$c(\tau_0^N) = (107, 125.5, 226, 236, 255, 15, 52, 70, 87, 204, 32.5, 146, 161.5, 182.5).$$

This will be the reference individual cost vector from which we need to subtract the savings of having an optimal order. First of all, we need to determine the branch-order. By looking at τ_0^N, we know that $\pi = (2, 3, 1)$. We thus need to solve the $2-$lines GMS-problem induced by branches B and C. The reference order is given by:

$$\tau_0^{2,\pi} = (b_1c_1b_2b_3b_4c_2c_3c_4b_5),$$

with $TC(\tau_0^{2,\pi}) = 796.5$, while the resulting optimal order by applying Algorithm 5.1 is:

$$\hat{\tau}^{2,\pi} = (b_1c_1c_2c_3b_2b_3b_4c_4b_5).$$

The corresponding merge order to $\hat{\tau}^{2,\pi}$ is given by:

$$\hat{\sigma}^{2,\pi} = (\underbrace{b_1}_{M_1}\ \underbrace{c_1}_{M_2}\ \underbrace{c_2c_3}_{M_3}\ \underbrace{b_2}_{M_4}\ \underbrace{b_3}_{M_5}\ \underbrace{b_4}_{M_6}\ \underbrace{c_4}_{M_7}\ \underbrace{b_5}_{M_8}),$$

with $TC(\hat{\sigma}^{2,\pi}) = 795.5$. Let us compute $\text{BSR}(\tau_0^{2,\pi}, \hat{\sigma}^{2,\pi})$. Note that the elements of all merge segments are already together in $\tau_0^{2,\pi}$, so we just need to reorder them. It can be observed that M_4 is on position 3, while M_3 is on position 6. By Algorithm 5.1, we know that $CAT(M_3) \leqslant CAT(M_3^{\cup})$, where $M_3^{\cup} = M_4 \cup M_5 \cup M_6$. Thus, we move M_3 to the front of M_4, thus obtaining the optimal order. This leads to a saving of 1 that needs to be allocated among the players in $M_3^{\cup}$ and M_3. In particular, players b_2, b_3, and b_4 should receive $\frac{1}{2\cdot 3} \cdot 1 = 0.167$ each, and players c_2 and c_3 should receive $\frac{1}{2\cdot 2} \cdot 1 = 0.25$ each. The savings allocation rule results in:

$$\text{BSR}(\tau_0^{2,\pi}, \hat{\sigma}^{2,\pi}) = (\underbrace{0,0,0,0,0}_{A},\underbrace{0,0.167,0.167,0.167,0}_{B},\underbrace{0,0.25,0.25,0}_{C}).$$

Note that:

$$g^{2,\pi} = TC(\tau_0^{2,\pi}) - TC(\hat{\tau}^{2,\pi}) = 796.5 - 795.5 = 1.$$

Thus,

$$f^{2,\pi} = \frac{1}{g^{2,\pi}} \cdot \text{BSR}(\tau_0^{2,\pi}, \hat{\sigma}^{2,\pi}) = (0,0,0,0,0,0,0.167,0.167,0.167,0,0,0.25,0.25,0).$$

Following Algorithm 5.2, we can convert branches B and C into one branch induced by $\hat{\tau}^{2,\pi}$, which results in the $2-$lines GMS-problem of Figure 6.3. The reference order for this new

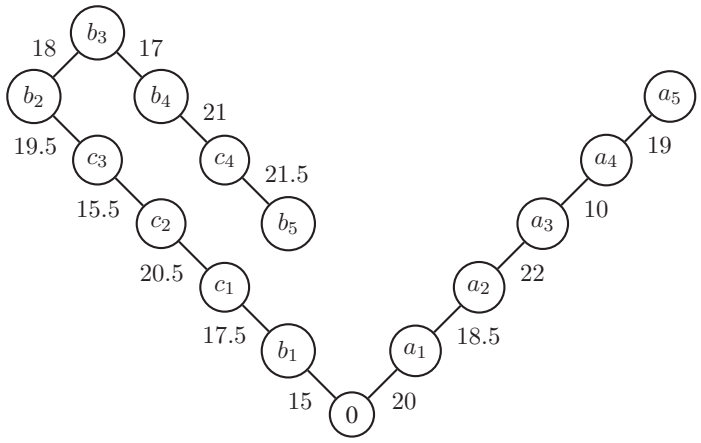


Figure 6.3: Resulting $2-$lines GMS-problem with π.

problem is given by:

$$\tau_0^{3,\pi} = (b_1c_1a_1a_2c_2c_3b_2b_3b_4c_4b_5a_3a_4a_5),$$

with an associated cost of $TC(\tau_0^{3,\pi}) = 1905.5$, while the resulting optimal order by applying Algorithm 5.1 would be:

$$\hat{\tau}^{3,\pi} = (b_1 c_1 a_1 a_2 a_3 a_4 c_2 c_3 b_2 b_3 b_4 a_5 c_4 b_5),$$

with $TC(\hat{\tau}^{3,\pi}) = 1859$. Note that this is also the optimal order provided by Algorithm 5.2 for all players N, $\hat{\tau}^N$. The corresponding merge order to $\hat{\tau}^{3,\pi}$ is

$$\hat{\sigma}^{3,\pi} = (\underbrace{b_1}_{M_1}\ \underbrace{c_1}_{M_2}\ \underbrace{a_1 a_2 a_3 a_4}_{M_3}\ \underbrace{c_2 c_3}_{M_4}\ \underbrace{b_2 b_3}_{M_5}\ \underbrace{b_4}_{M_6}\ \underbrace{a_5}_{M_7}\ \underbrace{c_4}_{M_8}\ \underbrace{b_5}_{M_9}).$$

We now need to calculate $\text{BSR}(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi})$. In order to allocate the savings in going from $\tau_0^{3,\pi}$ to $\hat{\sigma}^{3,\pi}$, we first need to put the merge segments together. First note that $|M_3/\tau_0^{3,\pi}| > 1$. By Lemma 5.1, we know that $CAT(\{a_1, a_2\}) > CAT(\{a_3, a_4\})$. Thus, by Proposition 5.1, one of the following two orders is better than $\tau_0^{3,\pi}$:

$$\tau_1 = (b_1 c_1 a_1 a_2\, \underbrace{a_3 a_4}\, \overbrace{c_2 c_3 b_2 b_3 b_4 c_4 b_5}\, a_5) \quad \text{or} \quad \tau_2 = (b_1 c_1\, \overbrace{c_2 c_3 b_2 b_3 b_4 c_4 b_5}\, \underbrace{a_1 a_2}\, a_3 a_4 a_5).$$

Since $TC(\tau_1) = 1863.5$ and $TC(\tau_2) = 1902$, we choose $\tau' = \tau_1$. The misplaced blocks are $\{c_2, c_3, b_2, b_3, b_4, c_4, b_5\}$ and $\{a_3, a_4\}$, which lead to a saving of 42. We need to allocate $\frac{1}{2\cdot 7} \cdot 42 = 3$ for players $c_2, c_3, b_2, b_3, b_4, c_4$, and b_5, and $\frac{1}{2\cdot 2} \cdot 42 = 10.5$ for a_3 and a_4. The savings allocation rule at this iteration is given by:

$$\text{BSR}^1(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi}) = (\underbrace{0, 0, 10.5, 10.5, 0}_{A}, \underbrace{0, 3, 3, 3, 3, 3}_{B}, \underbrace{0, 3, 3, 3}_{C}).$$

Note that τ' can be expressed as:

$$\tau' = (\underbrace{b_1}_{M_1}\ \underbrace{c_1}_{M_2}\ \underbrace{a_1 a_2 a_3 a_4}_{M_3}\ \underbrace{c_2 c_3}_{M_4}\ \underbrace{b_2 b_3}_{M_5}\ \underbrace{b_4}_{M_6}\ \underbrace{c_4}_{M_8}\ \underbrace{b_5}_{M_9}\ \underbrace{a_5}_{M_7}).$$

Now that all the merge segments are connected, we still need to reorder them. By looking at τ', it can be observed that M_8 is on position 7, while M_7 is on position 9. By Algorithm 5.1, we know that $CAT(M_7) \leqslant CAT(M_7^{\cup})$, where $M_7^{\cup} = M_8 \cup M_9$. Therefore, we move M_7 to the front of M_8, thus obtaining the optimal order. This leads to a saving of $g_{M_7^{\cup} M_7} = |M_7| \cdot |M_7^{\cup}| \cdot (CAT(M_7^{\cup}) - CAT(M_7)) = 1 \cdot 2 \cdot \left(\frac{21+21.5}{1+1} - \frac{19}{1}\right) = 4.5$ that needs to be allocated among the players in $M_7^{\cup}$ and M_7. In particular, players c_4 and b_5 should receive $\frac{1}{2\cdot 2} \cdot 4.5 = 1.125$ each, and player a_5 should receive $\frac{1}{2\cdot 1} \cdot 4.5 = 2.25$. The savings allocation

in this iteration results in:

$$\text{BSR}^2(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi}) = (\underbrace{0,0,0,0,2.25}_{A}, \underbrace{0,0,0,0,1.125}_{B}, \underbrace{0,0,0,1.125}_{C}).$$

The final savings allocation is:

$$\text{BSR}(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi}) = \sum_{\text{it}=1}^{2} \text{BSR}^{\text{it}}(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi}) = (\underbrace{0,0,10.5,10.5,2.25}_{A}, \underbrace{0,3,3,3,4.125}_{B}, \underbrace{0,3,3,4.125}_{C}).$$

Note that:

$$g^{3,\pi} = TC(\tau_0^{3,\pi}) - TC(\hat{\tau}^{3,\pi}) = TC(\tau_0^{3,\pi}) - TC(\hat{\tau}^{3,\pi}) = 1905.5 - 1859 = 46.5.$$

Thus,

$$f^{3,\pi} = \frac{1}{g^{3,\pi}} \cdot \text{BSR}(\tau_0^{3,\pi}, \hat{\sigma}^{3,\pi}) = \frac{1}{46.5} \cdot (0,0,10.5,10.5,2.25,0,3,3,3,4.125,0,3,3,4.125).$$

Also, we can now compute:

$$g^{\pi}(0) = g^{2,\pi} + g^{3,\pi} = 1 + 46.5 = 47.5,$$

and hence:

$$\begin{aligned} f^{\pi}(0) &= \frac{g^{2,\pi}}{g^{\pi}(0)} \cdot f^{2,\pi} + \frac{g^{3,\pi}}{g^{\pi}(0)} \cdot f^{3,\pi} = \frac{1}{47.5} \cdot f^{2,\pi} + \frac{46.5}{47.5} \cdot f^{3,\pi} \\ &= \frac{1}{47.5} \cdot (0,0,0,0,0,0,0.167,0.167,0.167,0,0,0.25,0.25,0) \\ &\quad + \frac{46.5}{47.5} \cdot \frac{1}{46.5} \cdot (0,0,10.5,10.5,2.25,0,3,3,3,4.125,0,3,3,4.125) \\ &= (0,0,0.221,0.221,0.047,0,0.067,0.067,0.067,0.087,0,0.068,0.068,0.087). \end{aligned}$$

Therefore, the κ rule leads to the allocation:

$$\begin{aligned}\kappa &= c(\tau_0^N) - \big(TC(\tau_0^N) - TC(\hat{\tau}^N)\big) \cdot f^{\pi}(0)\\ &= (107, 125.5, 226, 236, 255, 15, 52, 70, 87, 204, 32.5, 146, 161.5, 182.5)\\ &\quad - (1900 - 1859) \cdot (0, 0, 0.221, 0.221, 0.047, 0, 0.067, 0.067, 0.067, 0.087,\\ &\qquad\qquad 0, 0.068, 0.068, 0.087)\\ &= (107, 125.5, 226, 236, 255, 15, 52, 70, 87, 204, 32.5, 146, 161.5, 182.5)\\ &\quad - (0, 0, 9.061, 9.061, 1.927, 0, 2.747, 2.747, 2.747, 3.567, 0, 2.788, 2.788, 3.567)\\ &= (\underbrace{107, 125.5, 216.939, 226.939, 253.073}_{A}, \underbrace{15, 49.253, 67.253, 84.253, 200.433}_{B},\\ &\quad \underbrace{32.5, 143.212, 158.712, 178.933}_{C}).\end{aligned}$$ △

The following example illustrates how to achieve the cost allocation rule in a tree GMS-problem.

Example 6.5. Let us consider the tree GMS-problem from Figure 6.4. It can be observed that there are four sub-sources: the source 0, two sub-sources of level 2 (a_1 and b_1), and one sub-source of level 3 (a_{11}), the highest level of this problem. Thus, $\mathcal{S} = \{0, a_1, b_1, a_{11}\}$.

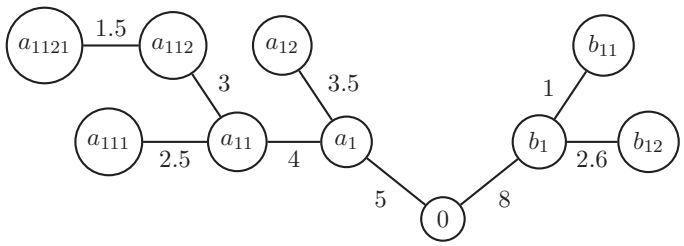


Figure 6.4: Example of a tree GMS-problem.

The reference order for all players N is:

$$\tau_0^N = (a_1 a_{12} a_{11} a_{111} a_{112} a_{1121} b_1 b_{11} b_{12}),$$

with $TC(\tau_0^N) = 165.6$. Also, we know that the individual cost vector is given by

$$c(\tau_0^N) = (\underbrace{12.5}_{a_{11}}, \underbrace{5}_{a_1}, \underbrace{27.5}_{b_1}, \underbrace{15}_{a_{111}}, \underbrace{18}_{a_{112}}, \underbrace{19.5}_{a_{1121}}, \underbrace{8.5}_{a_{12}}, \underbrace{28.5}_{b_{11}}, \underbrace{31.1}_{b_{12}}),$$

and from this vector is from which we need to subtract the quantity $TC(\tau_0^N) - TC(\hat{\tau}^N)$ to get the desired cost allocation vector.

We need to recursively solve the $2-$lines GMS-problems arising at the sub-sources. Note

that in this problem there are no ties since all the urgencies are different. Thus, considering the branch-order $\pi = (1, 2)$ or $\pi = (2, 1)$ at any of the sub-sources would lead to the same result. We have to start by applying Algorithm 5.1 to the $2-$lines GMS-problem arising from a_{11}, which is the sub-source from the highest level. This leads to $\hat{\tau}^{N_{a_{11}}} = (a_{112}a_{1121}a_{111})$, with an associated cost of $TC(\hat{\tau}^{N_{a_{11}}}) = 14.5$ and a corresponding merge order $\hat{\sigma}^{N_{a_{11}}} = (\underbrace{a_{112}a_{1121}}_{M_1}\underbrace{a_{111}}_{M_2})$. Note that for this subproblem, $\tau_0^{N_{a_{11}}} = (a_{111}a_{112}a_{1121})$, with $TC(\tau_0^{N_{a_{11}}}) = 15$. Hence,

$$w(a_{11}) = TC(\tau_0^{N_{a_{11}}}) - TC(\hat{\tau}^{N_{a_{11}}}) = 15 - 14.5 = 0.5.$$

To go from $\tau_0^{N_{a_{11}}}$ to $\hat{\sigma}^{N_{a_{11}}}$, we need to switch the merge segments M_2 and M_1. This leads to a saving of 0.5, which we need to allocate. We distribute $\frac{1}{2\cdot 1} \cdot 0.5 = 0.25$ to a_{111}; and $\frac{1}{2\cdot 2} \cdot 0.5 = 0.125$ to a_{112} and a_{1121}. Thus, we have that:

$$f(a_{11}) = \frac{1}{0.5} \cdot (0, 0, 0, 0.25, 0.125, 0.125, 0, 0, 0) = (0, 0, 0, 0.5, 0.25, 0.25, 0, 0, 0).$$

We convert these two branches into one, resulting in the tree GMS-problem of Figure 6.5, whose highest level has been reduced to 2.

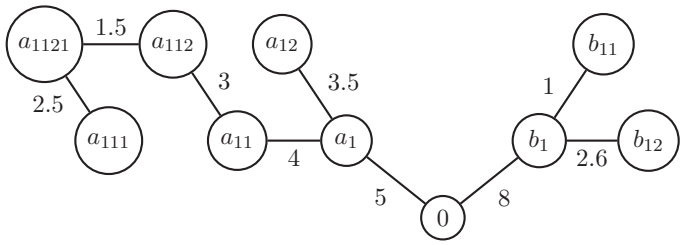


Figure 6.5: Resulting tree GMS-problem after the first iteration.

Let us now consider the $2-$lines GMS-problems arising from the sub-sources of level 2. Let us first focus on the subproblem starting at a_1. Note that $\tau_0^{N_{a_1}} = (a_{12}a_{11}a_{112}a_{1121}a_{111})$, with $TC(\tau_0^{N_{a_1}}) = 48$. By applying Algorithm 5.1, we get $\hat{\tau}^{N_{a_1}} = (a_{11}a_{112}a_{1121}a_{111}a_{12})$, with a total cost of $TC(\hat{\tau}^{N_{a_1}}) = 45$ and an associated merge order $\hat{\sigma}^{N_{a_1}} = (\underbrace{a_{11}a_{112}a_{1121}}_{M_1}\underbrace{a_{111}}_{M_2}\underbrace{a_{12}}_{M_3})$. Hence,

$$w(a_1) = TC(\tau_0^{N_{a_1}}) - TC(\hat{\tau}^{N_{a_1}}) = 48 - 45 = 3.$$

We need to allocate the savings in going from $\tau_0^{N_{a_1}}$ to $\hat{\sigma}^{N_{a_1}}$ among the players involved. In $\tau_0^{N_{a_1}}$, all the merge segments have their elements together. But we still need to reorder these merge segments. In particular, we first should allocate the savings in going from $\tau_0^{N_{a_1}}$

to $\tau'^{N_{a_1}} = (\underbrace{a_{11}a_{112}a_{1121}}_{M_1}\underbrace{a_{12}}_{M_3}\underbrace{a_{111}}_{M_2})$, with $TC(\tau'^{N_{a_1}}) = 46$. M_3 and M_1 have been switched, thus player a_{12} should receive $\frac{1}{2\cdot 1}\cdot 2 = 1$, and players a_{11}, a_{112}, and a_{1121} should obtain $\frac{1}{2\cdot 3}\cdot 2 = 0.333$. Now, we have to go from $\tau'^{N_{a_1}}$ to $\hat{\sigma}^{N_{a_1}}$ by switching M_3 and M_2, which results in a saving of 1. We have to allocate $\frac{1}{2\cdot 1}\cdot 1 = 0.5$ to a_{12}, and $\frac{1}{2\cdot 1}\cdot 1 = 0.5$ for a_{111}. Thus, we have that:

$$f(a_1) = \frac{1}{3}\cdot(0.333, 0, 0, 0.5, 0.333, 0.333, 1.5, 0, 0) = (0.111, 0, 0, 0.167, 0.111, 0.111, 0.5, 0, 0)\,.$$

Let us now consider the subproblem arising from b_1. In this case, the reference order, $\tau_0^{N_{b_1}} = (b_{11}b_{12})$, coincides with the order provided by Algorithm 5.1, so there is no positive saving to allocate. Thus,

$$\begin{aligned} w(b_1) &= 0,\\ f(b_1) &= (0, 0, 0, 0, 0, 0, 0, 0.5, 0.5)\,, \end{aligned}$$

where $f(b_1)$ is the assessment for the trivial case (consisting in allocating the same quantity to all the players involved, provided that the sum is 1).

We have already solved the subproblems of level 2, finishing the second iteration. We convert the subproblems into branches that follow the corresponding optimal orders provided by Algorithm 5.1. This results in the $2-$lines GMS-problem of Figure 6.6.

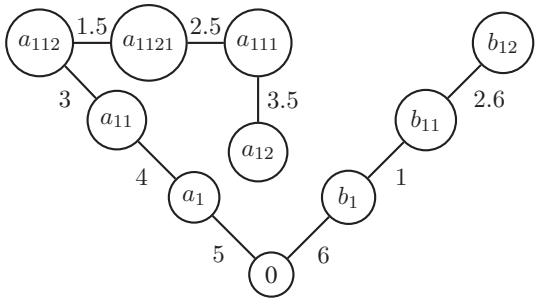


Figure 6.6: Resulting $2-$lines GMS-problem after the second iteration.

For this problem, $\tau_0^{N_0} = (a_1a_{11}a_{112}a_{1121}a_{111}a_{12}b_1b_{11}b_{12})$ with $TC(\tau_0^{N_0}) = 156.1$. However, by Algorithm 5.1, we have that $\hat{\tau}^{N_0} = (b_1b_{11}b_{12}a_1a_{11}a_{112}a_{1121}a_{111}a_{12})$ is an optimal order, with a cost of $TC(\hat{\tau}^{N_0}) = 155.2$. Note that $\hat{\tau}^{N_0}$ is the optimal order provided by Algorithm 5.3 for the tree GMS-problem of Figure 6.4, i.e., $\hat{\tau}^{N_0} = \hat{\tau}^N$. The associated merge order is given by:

$$\hat{\sigma}^{N_0} = (\underbrace{b_1b_{11}b_{12}}_{M_1}\underbrace{a_1}_{M_2}\underbrace{a_{11}}_{M_3}\underbrace{a_{112}}_{M_4}\underbrace{a_{1121}}_{M_5}\underbrace{a_{111}}_{M_6}\underbrace{a_{12}}_{M_7}).$$

Thus, we have that

$$w(0) = TC(\tau_0^{N_0}) - TC(\hat{\tau}^{N_0}) = 156.1 - 155.2 = 0.9.$$

Note that in $\tau_0^{N_0}$, all the merge segments are together, but they are not correctly placed with respect to $\hat{\sigma}^{N_0}$. Looking at $\tau_0^{N_0}$, it can be seen that M_1 is on position 7. Thus, we need to switch $M_1^{\cup} = \cup_{k=2}^{7} M_k$ and M_1, resulting in $\hat{\sigma}^{N_0}$. The total savings are 0.9, which should be distributed among the players involved. In particular, $\frac{1}{2\cdot 6} \cdot 0.9 = 0.075$ should be allocated to $a_1, a_{11}, a_{112}, a_{1121}, a_{111}$, and a_{12}, and $\frac{1}{2\cdot 3} \cdot 0.9 = 0.15$ should be allocated to b_1, b_{11}, and b_{12}. Hence, we have that:

$$\begin{aligned} f(0) &= \frac{1}{0.9} \cdot (0.075, 0.075, 0.15, 0.075, 0.075, 0.075, 0.075, 0.15, 0.15) \\ &= (0.083, 0.083, 0.167, 0.083, 0.083, 0.083, 0.083, 0.167, 0.167). \end{aligned}$$

Note that $\sum_{s\in\mathcal{S}} w(s)$ is given by:

$$\sum_{s\in\mathcal{S}} w(s) = w(a_{11}) + w(a_1) + w(b_1) + w(0) = 0.5 + 3 + 0 + 0.9 = 4.4.$$

In consequence, we have that:

$$\begin{aligned} f^N &= \frac{w(a_{11})}{\sum_{s\in\mathcal{S}} w(s)} \cdot f(a_{11}) + \frac{w(a_1)}{\sum_{s\in\mathcal{S}} w(s)} \cdot f(a_1) + \frac{w(b_1)}{\sum_{s\in\mathcal{S}} w(s)} \cdot f(b_1) + \frac{w(0)}{\sum_{s\in\mathcal{S}} w(s)} \cdot f(0) \\ &= \frac{0.5}{4.4} \cdot (0, 0, 0, 0.5, 0.25, 0.25, 0, 0, 0) + \frac{3}{4.4} \cdot (0.111, 0, 0, 0.167, 0.111, 0.111, 0.5, 0, 0) \\ &\quad + \frac{0}{4.4} \cdot (0, 0, 0, 0, 0, 0, 0, 0.5, 0.5) \\ &\quad + \frac{0.9}{4.4} \cdot (0.083, 0.083, 0.167, 0.083, 0.083, 0.083, 0.083, 0.167, 0.167) \\ &= (0.093, 0.017, 0.034, 0.188, 0.121, 0.121, 0.358, 0.034, 0.034). \end{aligned}$$

Therefore, the cost allocation rule can be computed as follows:

$$\begin{aligned} \kappa &= c(\tau_0^N) - \big(TC(\tau_0^N) - TC(\hat{\tau}^N)\big) \cdot f^N \\ &= (12.5, 5, 27.5, 15, 18, 19.5, 8.5, 28.5, 31.1) \\ &\quad - (165.6 - 155.2) \cdot (0.093, 0.017, 0.034, 0.188, 0.121, 0.121, 0.358, 0.034, 0.034) \\ &= (11.533, 4.823, 27.146, 13.045, 16.742, 18.242, 4.777, 28.146, 30.746). \end{aligned}$$

△

The procedure presented in the previous section for calculating the κ rule is in principle limited to the case where all agents have different urgencies and, consequently, there is a single reference order (both for the whole problem and locally at each sub-source). In the following, we will give some general indications on how to proceed when there are ties, and we will briefly show an example as an illustration.

First, we consider all possible reference orders. The machine, faced with a tie, chooses with equal probability which job to process. Thus, given a set of reference orders, we will also have a probability distribution. Now, fixed a reference order, we proceed in the same way as explained in Section 6.2, obtaining the κ rule (which will now depend on that reference order). The only difference is that, when solving the $n-$lines GMS-problems associated with each sub-source, the (local) reference order in these subproblems will not be recalculated according to possible ties that may exist. Instead, we will take the restriction of the (general) reference order that we are considering to the agents involved in that subproblem. Once we have the κ rule for each reference order, we will obtain the final cost allocation rule as a weighted average of all the κ rules obtained, where the weights are the probabilities of each possible reference order. Let us see with the following example how to obtain such a rule when we have ties.

Example 6.6. Let us slightly modify the $3-$lines GMS-problem from Example 6.4, as shown in Figure 6.7.

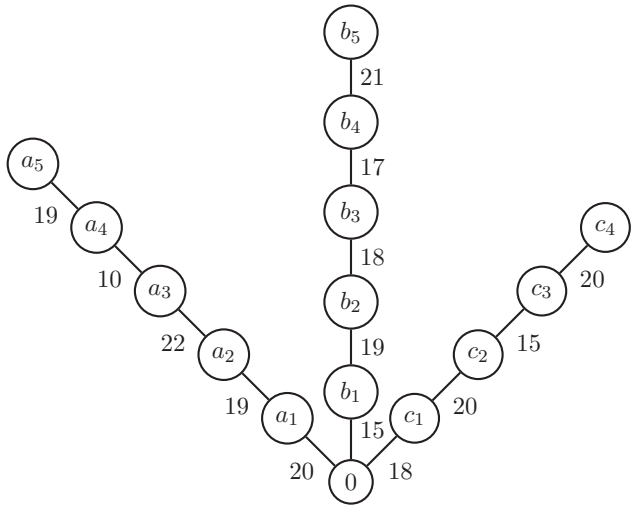


Figure 6.7: Example of a $3-$lines GMS-problem with ties.

When constructing the reference order, it is clear that the machine starts by selecting agents b_1, c_1, b_2, b_3 and b_4. At that point, the machine needs to decide between processing job

a_1 or job c_2 since both have an urgency of $\frac{1}{20}$. It chooses each of them with probability $\frac{1}{2}$. If job a_1 is selected, the remaining jobs are processed further without encountering any ties. If job c_2 is chosen, the machine is again faced with a decision: choose a_1 or c_4. Again, it picks each with probability $\frac{1}{2}$. Thus, there are three possible reference orders for all players N that can occur with probabilities $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{4}$, respectively:

$$\begin{aligned}
\tau^N_{0,1} &= (b_1c_1b_2b_3b_4a_1a_2c_2c_3c_4b_5a_3a_4a_5), \text{ or}\\
\tau^N_{0,2} &= (b_1c_1b_2b_3b_4c_2c_3c_4a_1a_2b_5a_3a_4a_5), \text{ or}\\
\tau^N_{0,3} &= (b_1c_1b_2b_3b_4c_2c_3a_1a_2c_4b_5a_3a_4a_5),
\end{aligned}$$

with $TC(\tau^N_{0,1}) = 1891$, $TC(\tau^N_{0,2}) = 1884$, and $TC(\tau^N_{0,3}) = 1883$. Also, we know that

$$\begin{aligned}
c(\tau^N_{0,1}) &= (107, 126, 224, 234, 253, 15, 52, 70, 87, 202, 33, 146, 161, 181),\\
c(\tau^N_{0,2}) &= (162, 181, 224, 234, 253, 15, 52, 70, 87, 202, 33, 107, 122, 142),\\
c(\tau^N_{0,3}) &= (142, 161, 224, 234, 253, 15, 52, 70, 87, 202, 33, 107, 122, 181).
\end{aligned}$$

These will be the reference vectors from which we need to subtract the savings of having the corresponding optimal order to each initial one. In this problem, the branch-order associated with each reference order is the same: $\pi = (3, 2, 1)$. Once this branch-order is fixed, we proceed analogously for the three reference orders. We will only show the intermediate elements obtained for $\tau^N_{0,1}$, as an illustration, to calculate $\kappa(\tau^N_{0,1})$. We begin by first solving the $2-$lines GMS-problem induced by branches B and C, thus leading to:

$$\begin{aligned}
\tau^{2,\pi}_{0,1} &= \tau^N_{0,1}|_{C\cup B} = (b_1c_1b_2b_3b_4c_2c_3c_4b_5);\\
\hat{\tau}^{2,\pi}_1 &= (b_1c_1c_2c_3b_2b_3b_4c_4b_5);\\
\hat{\sigma}^{2,\pi}_1 &= (\underbrace{b_1}_{M_1}\ \underbrace{c_1}_{M_2}\ \underbrace{c_2c_3}_{M_3}\ \underbrace{b_2}_{M_4}\ \underbrace{b_3}_{M_5}\ \underbrace{b_4}_{M_6}\ \underbrace{c_4}_{M_7}\ \underbrace{b_5}_{M_8});\\
g^{2,\pi}_1 &= TC(\tau^{2,\pi}_{0,1}) - TC(\hat{\tau}^{2,\pi}_1) = 791 - 788 = 3;\\
f^{2,\pi}_1 &= \frac{1}{g^{2,\pi}_1} \cdot \text{BSR}(\tau^{2,\pi}_{0,1}, \hat{\sigma}^{2,\pi}_1) = \frac{1}{3} \cdot (0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0.5, 0, 0, 0.75, 0.75, 0).
\end{aligned}$$

Following Algorithm 5.2, we can convert branches B and C into one branch induced by $\hat{\tau}^{2,\pi}_1$, which results in the $2-$lines GMS-problem of Figure 6.8. At this point, if we were to consider the urgencies of the jobs, we would obtain three possible reference orders, with different probabilities. However, as already mentioned, we take the restriction of the general reference order, $\tau^N_{0,1}$, to the agents involved, which for this problem are all of them. Thus, we

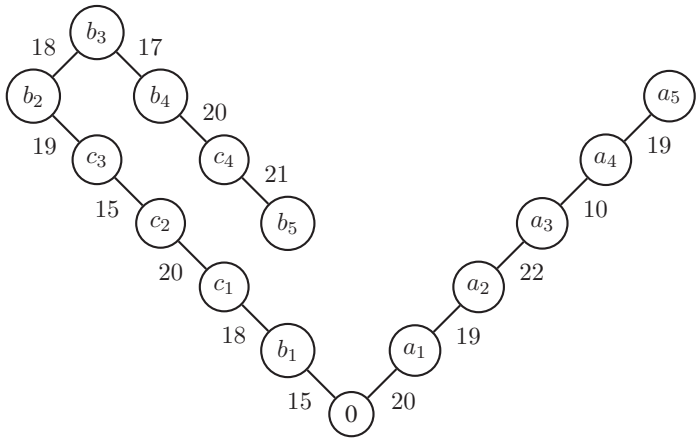


Figure 6.8: Resulting $2-$lines GMS-problem.

will have that:

$$\tau_{0,1}^{3,\pi} = \tau_{0,1}^{N} = (b_1c_1b_2b_3b_4a_1a_2c_2c_3c_4b_5a_3a_4a_5);$$
$$\hat{\tau}_1^{3,\pi} = \hat{\tau}_1 = (b_1c_1c_2c_3a_1a_2a_3a_4b_2b_3b_4a_5c_4b_5);$$
$$\hat{\sigma}_1^{3,\pi} = \hat{\sigma}_1 = (\underbrace{b_1}_{M_1}\ \underbrace{c_1c_2c_3}_{M_2}\ \underbrace{a_1a_2a_3a_4}_{M_3}\ \underbrace{b_2}_{M_4}\ \underbrace{b_3}_{M_5}\ \underbrace{b_4}_{M_6}\ \underbrace{a_5}_{M_7}\ \underbrace{c_4}_{M_8}\ \underbrace{b_5}_{M_9});$$
$$g_1^{3,\pi} = TC(\tau_{0,1}^{3,\pi}) - TC(\hat{\tau}_1^{3,\pi}) = 1891 - 1856 = 35;$$
$$f_1^{3,\pi} = \frac{1}{g_1^{3,\pi}} \cdot \text{BSR}(\tau_{0,1}^{3,\pi}, \hat{\sigma}_1^{3,\pi})$$
$$= \frac{1}{35} \cdot (1.475, 1.475, 4.875, 4.875, 1.5, 0, 1.6, 1.6, 1.6, 5.25, 0, 2.75, 2.75, 5.25).$$

Consequently:

$$g_1^{\pi}(0) = g_1^{2,\pi} + g_1^{3,\pi} = 3 + 35 = 38;$$
$$f_1^{\pi}(0) = \frac{g_1^{2,\pi}}{g_1^{\pi}(0)} \cdot f_1^{2,\pi} + \frac{g_1^{3,\pi}}{g_1^{\pi}(0)} \cdot f_1^{3,\pi}$$
$$= (0.039, 0.039, 0.128, 0.128, 0.039, 0, 0.055, 0.055, 0.055, 0.138, 0, 0.092, 0.092, 0.138).$$

Hence,

$$\kappa(\tau_{0,1}^{N}) = c(\tau_{0,1}^{N}) - \left(TC(\tau_{0,1}^{N}) - TC(\hat{\tau}_1^{N})\right) \cdot f_1^{\pi}(0)$$
$$= (\underbrace{105.635, 124.635, 219.520, 229.520, 251.635}_{A}, \underbrace{15, 50.075, 68.075, 85.075, 197.170}_{B},$$
$$\underbrace{33, 142.780, 157.780, 176.170}_{C}).$$

Proceeding analogously with $\tau^N_{0,2}$ and $\tau^N_{0,3}$, we obtain:

$$\kappa(\tau^N_{0,2}) = (\underbrace{160.645, 179.645, 220.387, 230.387, 251.645}_{A}, \underbrace{15, 49.742, 67.742, 84.742, 196.806}_{B},$$
$$\underbrace{33, 105.645, 120.645, 139.968}_{C});$$
$$\kappa(\tau^N_{0,3}) = (\underbrace{141.662, 160.662, 219.613, 229.613, 251.650}_{A}, \underbrace{15, 50.650, 68.650, 85.650, 197.275}_{B},$$
$$\underbrace{33, 105.650, 120.650, 176.275}_{C}).$$

Therefore, we calculate the final cost allocation rule as follows:

$$\kappa = \sum_{k=1}^{3} p(\tau^N_{0,k}) \cdot \kappa(\tau^N_{0,k}) = \frac{1}{2} \cdot \kappa(\tau^N_{0,1}) + \frac{1}{4} \cdot \kappa(\tau^N_{0,2}) + \frac{1}{4} \cdot \kappa(\tau^N_{0,3})$$
$$= (\underbrace{128.398, 147.398, 219.755, 229.755, 251.633}_{A}, \underbrace{15, 50.131, 68.131, 85.131, 197.103}_{B},$$
$$\underbrace{33, 124.212, 139.212, 167.143}_{C}). \qquad \triangle$$

6.4 Conclusions and further research

The GMS-problem was formally presented in Chapter 5. Due to the complexity of solving the general model, we focused on tree GMS-problems. The proposed solution algorithms are reformulations of previous work, but including more elaborate procedures and additional ingredients that were essential for cost allocation. In particular, in this chapter, we have provided an allocation rule, the κ rule, to distribute the cost of an optimal order. It follows an algorithmic procedure similar to the one used in the optimization methods described in Chapter 5. More specifically, the methodology employed is based on recursion, where the network structures consisting of 2 lines arising from the source, play a fundamental role. For the calculation of this rule, a myopic reference order is used, which will depend on the problem in question. Using the theoretical results of the previous chapter, we see that it is possible to go from the reference order to the optimal order by non-negative savings when we exchange consecutive blocks of agents chosen in a particular manner.

As future work, it would be of special interest to find properties that satisfies κ, as well as to propose an axiomatic characterization of such rule. Another open direction of research is to study the GMS-problem from the perspective of game theory. To this respect, works such as Estévez-Fernández et al. (2008) and Musegaas et al. (2018), may provide some ideas.

Chapter 7

Conclusions

The focus of this dissertation was the introduction and study of OR and game theoretical problems that could be applied directly or indirectly in the field of logistics. New models and methods have been proposed, motivated by situations that arose in diverse areas. In Section 7.1, we summarize the work that has been conducted in this thesis, which is structured in three parts, and discuss the results achieved in each chapter. In addition, open questions requiring future research are also addressed in Section 7.2.

Contents

7.1 Results and discussion

Part I of this thesis concentrates on a rich VRP variant, the MC-TTRP, which is a combination of the TTRP and MC-VRP. This problem arises motivated by the requirements of an agricultural cooperative that needs to distribute several types of feed to farms geographically scattered throughout Galicia (Spain), while minimizing the distance traveled. In the MC-TTRP, the vehicle fleet consists of trucks and trailers in such a way that when the complete vehicle (truck with trailer attached) cannot access a customer, the trailer is uncoupled and the delivery is carried out only with the truck. A mathematical formulation of the MC-TTRP as an MILP problem is presented in Chapter 2, detailing each of its constraints. Caramia and Guerriero (2010b) studies a similar problem motivated by the collection of milk on farms in Italy, but does not provide a complete modeling of it. Therefore, to the best of our knowledge, our mathematical formulation is the only existing one for the MC-TTRP. Subsequently, we implemented the optimization model in `AMPL` software and solved via `Gurobi` some small instances taken from the real cooperative case. We observed that the computational time becomes prohibitively high when the number of customers increases. One of our long-term goals is that the results of this thesis can be put into practice by the cooperative that motivated the study. This type of company seeks to obtain efficient route planning in a timely

manner, as they can be surprised by higher or lower demand at short notice. For this reason, we turn to approximate methods to solve the problem. In particular, Chapter 3 centers on proposing different two-phase heuristics. The first stage, common to all proposals, is a constructive heuristic based on the Clarke and Wright savings algorithm and the second stage is a metaheuristic that improves the solution obtained in the first stage. Thus, the developed metaheuristics (ITS, ALNS, ALNS-TS, and PALNS-TS) all start from the same initial solution, which is the solution obtained at the end of the first phase. They were implemented in R software and an extensive comparative study was carried out to analyze their performance. As no benchmark problems existed for the MC-TTRP, the first step consisted in designing a set of 21 new instances by adapting previous ones from the TTRP literature. The 21 problems were solved by the four methods and it was observed that the ITS obtained better results, so a more detailed study of it was conducted. First, the ITS implementation was tailored to the TTRP and the results achieved by this method were compared with other previous algorithms. Even though our proposal was not specifically created for the TTRP, competitive results were attained with the existing ones, showing the good performance of our ITS. Afterward, the solutions obtained by this metaheuristic in the new 21 instances were further analyzed, from which it was deduced that the ITS is capable of considerably improving the initial solutions with low computational cost. Finally, the results obtained by the ITS and the exact model in real instances of the agricultural cooperative were compared, showing the usefulness of our heuristic proposal, since it obtains solutions close to the optimal one in a much shorter computational time.

Part II of this dissertation introduces cooperative game theory in the field of machine learning, more precisely in classification problems. Starting from a set of categorized observations, a topic of great interest is knowing how responsible each of the characteristics of those observations has been in a certain categorization. Some classifiers have the advantage of providing scores for the explanatory variables in classification problems. This ranking will be dependent on the classifier under consideration. Works such as Štrumbelj and Kononenko (2010) or Datta et al. (2015) propose model-agnostic methods based on cooperative game theory solutions. The approach of Štrumbelj and Kononenko (2010) serves to evaluate the influence that the attributes of a certain individual have had on their assigned class. Datta et al. (2015) is not focused at an individual-level, i.e., it is not locally oriented, but considers a sample and bases its method theoretically only on the binary scenario. In Chapter 4, we have generalized these procedures by designing a globally oriented classifier-independent influence measure that allows us to assess how important each feature value is in the classification of a set of individuals. Thus, the work that has been carried out is framed within the context

of explainability and interpretability in classification problems. As far as we are aware, this specific approach had not been adopted before. The Shapley value was used to define our influence measure, for which an axiomatic characterization is provided through efficiency and balanced contribution properties. Subsequently, different computational experiments are designed and conducted to corroborate the proper performance of the measure, and we compare the results of our method with those obtained using the proposal of Datta et al. (2015). Both influence measures are implemented in R software. Finally, our methodology is applied to a database of patients infected with COVID-19 in the first wave of the pandemic in Galicia. Some demographic and risk factors of the patients (sex, age, and cardiovascular, respiratory, metabolic, and urinary diseases) are taken as features, and whether the individuals have deceased, have been admitted to the ICU, or have been hospitalized is considered as a response. The results achieved are in line with situations that actually occurred during the pandemic in the time period and geographic region under study. For example, it is observed that older patients are more likely to be hospitalized, but not to be sent to the ICU, probably because many of them passed away prior to their admission.

Part III of this thesis focuses on a new cooperative OR problem that arises by combining ideas from the MCSTPs and sequencing problems: the GMS-problem. It can be motivated by a situation of water distribution from a central reservoir to different locations in an urban area, which correspond to agents. The objective pursued comprises two stages: first, to design a network for supplying all the agents at minimum cost, and second, to distribute the cost of the optimal network among all the agents involved. Chapter 5 concentrates on the first goal. It is initially shown that solving the general GMS-problem is complex, so we restrict ourselves to the situation in which the initial structure is a tree. The first goal consists thus in finding an optimal order in which to provision the agents, such that the total costs are minimized, while considering the precedence relations imposed by the tree. A reformulation of earlier results is performed and more sophisticated solving algorithms are proposed for allocation purposes, for which optimality is proved. We illustrate through several examples how to apply these results to address a variety of problems. In Chapter 6, the GMS-problem is studied from the allocation point of view. We provide a cost allocation rule, the κ rule, whose computation involves an algorithmic procedure. Essentially, a reference order is taken as a starting point and is calculated using a myopic process. The savings corresponding to going from the reference order to the optimal one are divided among the agents involved, compensating each of them appropriately. Our method determines how these compensations are obtained, for which it is essential to draw on the theoretical results and algorithms of Chapter 5. Numerous examples are also given to illustrate the calculation of the κ rule.

7.2 Further research

In light of the above, some further research lines, which would allow to broaden the studies carried out in this dissertation, have been identified.

As for Part I, it would be of special interest to extend the proposed model to new real situations. This would enable us to solve similar problems that may arise in other logistics companies. The research of Chapter 3 would also benefit from a deeper comparison of the designed metaheuristics. It would be worthwhile to apply the ISA methodology (Smith-Miles et al., 2014) to identify more varied benchmark problems, and thus be able to determine the best performing regions for each of the methods. Finally, a work in progress is the development of an `R` package that integrates all the algorithms created in this part of the thesis.

Regarding Part II, the most immediate line of future work is to apply the influence measure to other databases. In addition, the possibility of comparing the results obtained by our method with other statistical techniques could be of interest. It would therefore be of particular benefit to create an `R` package that includes our methodology.

With respect to Part III, an open line of research is to conduct an in-depth study of the κ rule, by suggesting and analyzing properties that this rule could satisfy. We also highlight the important challenge of applying game theoretical approaches to the GMS-problem.

Finally, it is worth mentioning that work has been initiated in the context of cooperative routing problems, where game theory techniques are applied to VRPs, thus combining two of the main topics of this dissertation. In particular, a new transportation problem with cooperation has been modeled, for which the design of a solution algorithm is being studied. Short-term plans are to continue with this line of research.

Resumen en castellano

Los contenidos de esta tesis, titulada *Optimización y cooperación con aplicaciones logísticas*, son el resultado del trabajo realizado por Laura Davila Pena durante los estudios correspondientes al Programa de Doctorado en Estadística e Investigación Operativa de la Universidade de Santiago de Compostela. Este trabajo fue realizado en colaboración con sus directores de tesis Ignacio M. García Jurado (Universidade da Coruña) y Balbina V. Casas Méndez (Universidade de Santiago de Compostela), así como con los profesores José Fernando Oliveira y Maria Antónia Carravilla (Universidade do Porto) y Peter Borm (Tilburg University), responsables de las estancias de investigación preceptivas para optar a la mención internacional, y con los investigadores David Rodríguez Penas (Instituto de Investigacións Mariñas, IIM-CSIC) y Jop Schouten (Tilburg University).

La Investigación Operativa es una disciplina en la que se desarrollan y utilizan métodos analíticos avanzados para resolver problemas y ayudar en la toma de decisiones. Existen una gran variedad de campos en los que se puede aplicar esta disciplina, uno de los cuales es la logística. Esta área es la responsable de gestionar el flujo de bienes, personas o recursos entre diferentes localizaciones, atendiendo a ciertos requerimientos. También se extiende a la gestión y planificación de personal o recursos en distintos dominios.

El principal objetivo de esta tesis es estudiar diferentes problemas de Investigación Operativa que puedan tener o estar relacionados con aplicaciones en el ámbito logístico. Está estructurada en tres partes independientes que detallaremos un poco más abajo, y que ponen de manifiesto la pluralidad del presente trabajo.

El problema de rutas de vehículos (VRP) es uno de los problemas logísticos más conocidos y estudiados. Su naturaleza diversa lleva a la incorporación de restricciones más o menos complejas, resultando en una gran variedad de modelos y objetivos a perseguir. La primera parte de esta tesis se centra en una variante del VRP motivada por una situación real acontecida en una cooperativa agrícola gallega. Se introduce el modelo a considerar (Capítulo 2) y se proponen diferentes algoritmos de resolución (Capítulo 3).

Además del diseño óptimo de las rutas, los problemas logísticos también son tratados desde el punto de vista cooperativo. Una vez creadas las rutas, surge la pregunta de cómo

distribuir los costes derivados entre los agentes involucrados. En la tercera parte de esta tesis se estudia un nuevo problema de Investigación Operativa en el que los costes de un cierto sistema deben ser divididos entre los clientes que lo componen. Esto aparece motivado por una situación logística de distribución de agua en áreas urbanas, que será tratada tanto desde la perspectiva de la optimización (Capítulo 5) como de la asignación de costes (Capítulo 6).

El aspecto cooperativo en los problemas logísticos también se puede estudiar haciendo uso de la teoría de juegos. Esta disciplina puede aplicarse en una gran cantidad de dominios, destacando en los últimos años el aprendizaje automático. En la segunda parte de esta tesis combinamos estos dos campos. Motivados por la situación de pandemia derivada de la COVID-19, se propone un método para evaluar la influencia que ciertas características de pacientes infectados tuvieron en su evolución (Capítulo 4). Aunque este problema no trata directamente una aplicación logística, sí sirve como una herramienta de soporte a la toma de decisiones por parte de los profesionales sanitarios, encuadrándose dentro de la gestión de recursos.

A continuación presentamos los contenidos de la presente tesis más en profundidad, indicando sus principales contribuciones.

Parte I. Sobre los valiosos problemas de rutas de vehículos

El VRP es un problema de optimización combinatoria que busca encontrar una configuración óptima de rutas recorridas por múltiples vehículos para servir a un conjunto de clientes. Esta parte de la tesis se centra en una variante específica del VRP, llamada problema de rutas con camiones y remolques compartimentados (MC-TTRP). Está compuesta por dos capítulos que presentan el modelo y algoritmos de resolución. En lo que sigue, detallamos los contenidos de cada uno de ellos.

Capítulo 2. Problema de rutas con camiones y remolques compartimentados (MC-TTRP)

En los últimos años, la logística del transporte jugó un rol fundamental en la industria. Muchas compañías, públicas y privadas, están interesadas en desarrollar herramientas computacionales para diseñar sus rutas, con objetivos tales como la minimización de los costes y/o la maximización de los productos servidos. Los VRPs son problemas a través de los cuales se pueden modelizar rutas de transporte para distintos vehículos. La resolución de estos problemas de optimización matemática es un desafío para los investigadores operacionales. Cuando se incorporan nuevas características de problemas reales, tales como vehículos con capacidad

limitada, entrega de mercancía en unos periodos de tiempo determinados, o comportamientos estocásticos, aparecen nuevas variantes del VRP original, dando lugar a la necesidad de desarrollar nuevos modelos y técnicas de resolución.

Una modificación prometedora del VRP es el problema de rutas con camiones y remolques (TTRP), que surge a raíz de restricciones de accesibilidad (Chao, 2002). En esta variante, la flota está compuesta por camiones y remolques que visitan a un conjunto de clientes, donde algunos de ellos (clientes de vehículo, VCs) pueden ser atendidos por el vehículo completo (es decir, el camión con el remolque enganchado), mientras que otros solo pueden ser servidos por el camión (clientes de camión, TCs). Ejemplos de TCs son clientes en zonas del interior de la ciudad, en regiones montañosas, o en lugares donde la maniobra o acceso con un remolque no es posible. Para resolver este problema, distinguimos tres tipos de rutas: las rutas de camión puras (PTRs), que pueden ser recorridas solamente por camiones, las rutas de vehículo puras (PVRs), que se realizan enteramente con el vehículo completo y las rutas de vehículo mixtas (MVRs), que consisten en una ruta principal recorrida por el vehículo completo y una o más subrutas recorridas solo por el camión. La Figura RC.1 ilustra una posible solución al TTRP.

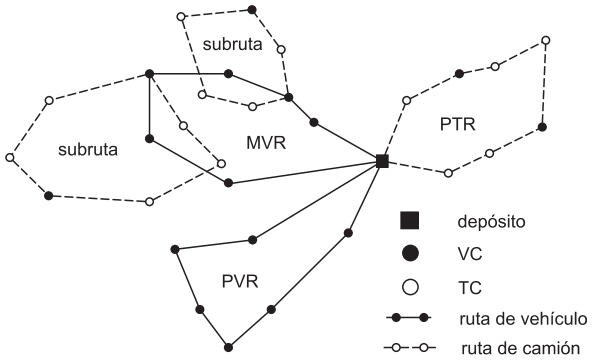


Figura RC.1: Posible solución al TTRP.

Otra variante interesante del VRP es el caso multi-compartimento, en el cual se deben dividir productos distintos en diferentes almacenajes durante el transporte. Si bien es cierto que la inclusión de compartimentos añade una complejidad adicional, puede ser un requerimiento en algunas aplicaciones logísticas reales.

En base a lo anteriormente expuesto, este capítulo se centra en un problema de rutas de vehículos que combina el TTRP con la compartimentación de los productos, al cual llamamos problema de rutas con camiones y remolques compartimentados (MC-TTRP). En particular, hemos propuesto una formulación matemática del MC-TTRP como un modelo de programación lineal y entera mixta (MILP) que abarca todo el problema, algo que no había sido

realizado anteriormente. La combinación de estas dos características está motivada por las necesidades de una cooperativa agrícola gallega que produce y distribuye pienso para el ganado (Guitián de Frutos y Casas-Méndez, 2019). En este capítulo, presentamos también el estudio de caso y lo usamos para probar, ilustrar y aplicar nuestra formulación. Utilizamos `AMPL` para la implementación del modelo y el solucionador `Gurobi` para resolverlo.

Los contenidos de este capítulo están recogidos en Davila-Pena et al. (2023).

Capítulo 3. Heurísticas para el problema de rutas con camiones y remolques compartimentados

En el Capítulo 2, empleando los datos de la cooperativa agrícola, observamos que obtener la solución exacta de un problema con más de 10 clientes es computacionalmente muy costoso. Esto deriva en la necesidad de recurrir a métodos aproximados de resolución, tales como heurísticas. En particular, en el presente capítulo, se introducen e implementan cuatro algoritmos heurísticos en dos fases.

La efectividad del algoritmo de ahorros de Clarke–Wright (CW) al construir soluciones para distintos VRPs, así como el requerimiento de resolver MC-TTRPs que involucran un número relativamente elevado de clientes, ha servido de motivación para adaptar este método al caso del MC-TTRP. Así, la primera fase de nuestras heurísticas coincide para las cuatro propuestas: una solución inicial es generada mediante un algoritmo heurístico constructivo basado en el CW.

En cuanto a la segunda fase, proponemos distintas metaheurísticas para mejorar la solución inicial obtenida por el CW. Específicamente, diseñamos cuatro algoritmos para esta segunda fase: dos de ellos son completamente independientes, la búsqueda tabú iterativa (ITS) y la búsqueda adaptativa de grandes vecindarios (ALNS); mientras que los otros dos combinan ideas de ambos, el ALNS híbrido con búsqueda tabú (ALNS-TS) y el ALNS-TS penalizado (PANLS-TS).

Tanto las fases constructiva como de mejora están integradas en unos algoritmos novedosos en dos fases para resolver el MC-TTRP. La implementación de estos métodos se realizó en el programa informático `R`. Se llevó a cabo un exhaustivo estudio computacional sobre diferentes conjuntos de datos. En primer lugar, adaptamos adecuadamente los 21 problemas de referencia del TTRP (Chao, 2002) para considerar compartimentos, dando lugar a 21 nuevas instancias. Nuestras heurísticas obtuvieron soluciones factibles para todos estos problemas, y los resultados alcanzados hacen evidente el efectivo y eficiente comportamiento de estos métodos al resolver el MC-TTRP. Realizamos un estudio comparativo de estas cuatro metaheurísticas y observamos que el ITS era la que llegaba a mejores soluciones. De acuerdo con

esto, se llevó a cabo un análisis más profundo de este algoritmo. Primeramente, y con el fin de obtener una validación inicial del mismo, decidimos adaptar nuestro método al caso del TTRP. De esta forma, es posible comparar los resultados con técnicas de resolución ya existentes en la literatura para el TTRP, y observamos que los resultados alcanzados por el ITS eran competitivos con los métodos previos, llegando incluso a obtener mejores soluciones para algunos de los problemas. También se realizó un estudio más detallado en las instancias generadas. Además, este algoritmo se aplicó al caso real de la cooperativa agrícola, con el fin de comparar estos resultados con la solución óptima proporcionada por nuestra formulación exacta. Se obtuvieron soluciones de buena calidad en un tiempo computacional mucho menor. Con todo esto, consideramos al ITS como un método de resolución prometedor para el MC-TTRP.

Los contenidos de este capítulo están recogidos en Davila-Pena et al. (2023) y Davila-Pena et al. (2022c).

Parte II. Sobre la teoría de juegos en los problemas de clasificación

Existen muchas situaciones en las que necesitamos anticipar un cierto resultado, en base a unas entradas específicas. Este tipo de problemas aparecen en una gran cantidad de campos tales como ciencia, economía o sanidad. Por ejemplo, podemos estar interesados en predecir los beneficios de una cierta compañía o en saber si un determinado paciente sufre o no una enfermedad. Este último caso, donde la respuesta es categórica, se encuadra dentro de los problemas de clasificación.

Así, un problema de clasificación implica predecir la categoría a la que pertenece un cierto individuo, en función de sus atributos. Estas predicciones están basadas en el conocimiento obtenido a través de una muestra de individuos para los que se conocen los valores de los atributos y de la respuesta. En este tipo de problemas, además de clasificar, es importante conocer cómo de influyentes son cada uno de estos atributos en la respuesta. Esta segunda parte de la tesis está compuesta por un único capítulo que presenta una medida para evaluar dicha influencia basándose en ideas de la teoría de juegos.

Capítulo 4. Una medida de influencia para los problemas de clasificación

Uno de los elementos fundamentales al tratar problemas de clasificación es disponer de una muestra de entrenamiento, es decir, un conjunto de observaciones para las que se conoce su

categoría y los valores de los atributos. En este capítulo se hace uso de técnicas de clasificación importadas del aprendizaje automático que permitirán desarrollar una herramienta metodológica para el análisis exploratorio de una muestra de entrenamiento como la que acabamos de mencionar. Algunos clasificadores tienen la ventaja de proporcionar puntuaciones para las variables explicativas en los problemas de clasificación, es decir, de evaluar su importancia. Estas puntuaciones dependerán del clasificador en cuestión. Trabajos como Štrumbelj y Kononenko (2010) o Datta et al. (2015) proponen métodos agnósticos al modelo basados en soluciones de la teoría de juegos cooperativa. El enfoque de Štrumbelj y Kononenko (2010) sirve para evaluar la influencia que los atributos de un cierto individuo han tenido en su clase asignada. Datta et al. (2015) no se centra en un único individuo, es decir, su propuesta no está localmente orientada, sino que considera una muestra de individuos y fundamenta teóricamente su método en el escenario binario. En nuestro caso, generalizamos estos procedimientos por medio del diseño de una medida de influencia que está orientada globalmente y es independiente del clasificador utilizado, que nos permitirá evaluar cómo de importantes son los distintos valores de los atributos en la clasificación de un conjunto de individuos. Hasta donde conocemos, este enfoque específico no había sido adoptado previamente en la literatura.

Para construir nuestra medida de influencia, comenzamos definiendo un juego cooperativo (cuyos jugadores son los atributos considerados) y luego aplicamos una solución de la teoría de juegos, conocida como el valor de Shapley. Se realiza también una caracterización axiomática de la medida propuesta por medio de propiedades adaptadas de aquellas que verifica el valor de Shapley en el contexto general de la teoría de juegos, y que son altamente deseables desde el punto de vista del análisis exploratorio de datos: eficiencia y contribuciones equilibradas. Para probar el alcance y adecuación de nuestra medida de influencia, se lleva a cabo un experimento de control que da lugar a unos resultados satisfactorios. Además, se efectúa también una comparativa entre nuestra propuesta y la de Datta et al. (2015), dando lugar nuestro método a resultados más adecuados desde la perspectiva de la interpretabilidad. Ambas herramientas han sido implementadas en el programa informático `R`.

Finalmente, aplicamos nuestra medida de influencia a un problema real en el que nos vimos inmersos recientemente. Durante la primera ola de la pandemia de la COVID-19, tuvimos acceso a una base de datos de $10\,454$ pacientes de Galicia infectados entre el 6 de marzo y el 7 de mayo de 2020. Conocer las características que hayan podido incrementar significativamente su probabilidad de necesitar acceso a ciertos servicios sanitarios es altamente útil para que las autoridades sanitarias tomen las decisiones más acertadas. De este modo, empleamos estos datos para determinar cuáles eran los valores de las características de los

pacientes que más afectaban al empeoramiento de su condición, y que podían dar lugar a su fallecimiento, a su ingreso en la unidad de cuidados intensivos (ICU), o a su hospitalización. Los atributos considerados fueron factores demográficos (edad y sexo) y ciertas patologías previas (enfermedades cardíacas, respiratorias, metabólicas y urinarias), discretizados en distintos niveles. Los resultados obtenidos son coherentes con las situaciones que realmente se produjeron durante la pandemia en el periodo de tiempo y región geográfica estudiados. Por ejemplo, se observa que los pacientes de más edad son más propensos a ser hospitalizados, pero no a ser enviados a la ICU, probablemente porque muchos de ellos fallecieron antes de su ingreso.

En base a lo anterior, nuestro método presenta un prometedor futuro como instrumento de apoyo a la toma de decisiones. Sirve, en particular, para alertar a los profesionales sanitarios de la importancia de tener en cuenta ciertas características de los pacientes, frente a la menor importancia o influencia de otras. Tales características pueden representar un potencial riesgo para pacientes con una cierta enfermedad, que en consecuencia deben ser tomados en cuenta tanto con respecto a los cuidados que deben recibir como en la planificación de los recursos destinados para ellos.

Los contenidos de este capítulo están recogidos en Davila-Pena et al. (2022b).

Parte III. Sobre los problemas cooperativos de Investigación Operativa

Los problemas de árboles de expansión de mínimo coste (MCSTPs) y los problemas de secuenciación son problemas multiagente de Investigación Operativa que persiguen dos objetivos: por un lado, minimizar los costes relativos a la construcción o diseño de una red o sistema; y por otro, distribuir esos costes entre los agentes involucrados. Una combinación de ideas de estos dos problemas da lugar al problema de planificación de máquinas en grafos (problema GMS), donde el objetivo perseguido consta también de dos fases. La tercera parte de esta tesis se centra en este problema y está formada por dos capítulos que presentan y estudian el modelo desde la perspectiva de la optimización y de la asignación de costes.

Capítulo 5. El problema de planificación de máquinas en grafos (problema GMS)

Tal y como se presenta en Bergantiños et al. (2014), existen numerosos problemas reales que requieren la construcción de infraestructuras para conectar un conjunto de agentes a una

fuente, bien directa o indirectamente. Uno de ellos es el abastecimiento urbano de agua desde un depósito central a determinados puntos de interés (agentes), que involucra la construcción de tuberías por toda una ciudad. La instalación de estas tuberías entre dos puntos requiere un cierto tiempo. El primer problema que surge en este tipo de situaciones es la cuestión de dónde colocar cada una de estas tuberías. El objetivo sería, así, conectar todos los agentes a la red de tal forma que se minimice el tiempo total de instalación o construcción. Para tratar este problema, el MCSTP ha sido ampliamente utilizado.

Sin embargo, es a menudo esencial que los agentes dispongan de suministro de agua en todo momento (por ejemplo, un hospital), por lo que deben contratar a una compañía externa que ofrezca este servicio mientras no puedan ser abastecidos desde el depósito central. De este modo, cada agente tiene un coeficiente asociado que indica el coste por unidad de tiempo en el sistema, es decir, por unidad de tiempo en la que los conductos que lo conectan a la fuente todavía no están instalados. Por lo tanto, el coste de un agente vendrá determinado por su coeficiente por unidad de tiempo multiplicado por el tiempo total requerido para conectar ese agente específico. Este tiempo total dependerá del momento en el que este agente se conecte a la fuente: por ejemplo, si el agente 2 se conecta a la fuente por medio del agente 1, entonces se debe construir primero la tubería que conecta el agente 1 a la fuente, seguido por la tubería que conecta el agente 1 al agente 2. Así, el tiempo total requerido para conectar el agente 2 a la fuente será la suma de los tiempos de construcción de las tuberías que conectan la fuente al agente 1, y el agente 1 al agente 2. El objetivo será, por tanto, minimizar los costes totales en lugar del tiempo de construcción total de la red.

Situaciones como la descrita arriba resultan en un nuevo tipo de problema, al que denominamos problema GMS. Un aspecto que nos gustaría subrayar de nuevo es la proximidad de nuestro problema a otros dos problemas estudiados en la literatura. Primero, el problema GMS está estrechamente ligado al MCSTP. No obstante, en el problema GMS los costes se calculan de un modo completamente diferente. En particular, el orden en el que se construyen las tuberías tiene un efecto sustancial en el coste en nuestro problema, mientras que este orden es irrelevante al calcular el coste en un MCSTP. Segundo, el problema GMS está profundamente relacionado con los problemas de secuenciación con restricciones de precedencia. Aunque los problemas de secuenciación con relaciones de precedencia han sido tratados en la literatura (Hamers et al., 2005), el enfoque bajo el que estudiaremos aquí nuestro problema nunca había sido adoptado, hasta donde conocemos.

Ahora bien, en este capítulo nos centramos en el estudio del problema GMS desde el punto de vista de la optimización. Comenzamos mostrando que resolver de forma óptima un problema GMS es en general complejo, por lo que consideramos situaciones en las que la

estructura inicial es un árbol. Así, el objetivo consiste en encontrar un orden óptimo en el que abastecer a los agentes, de tal forma que se minimicen los costes totales teniendo en cuenta las relaciones de precedencia impuestas por dicha estructura. Se lleva a cabo la reformulación de resultados previos de la literatura y se proponen algoritmos de resolución más sofisticados que los existentes, para los cuales se prueba su optimalidad. La introducción de nuevos ingredientes es fundamental para el estudio del problema GMS desde la perspectiva de la asignación de costes, que será tratado en el siguiente capítulo. Además, se ilustra mediante numerosos ejemplos cómo aplicar estos resultados a una variedad de problemas.

Los contenidos de este capítulo están recogidos en Davila-Pena et al. (2022a).

Capítulo 6. Una regla de asignación para los problemas GMS

Al construir un sistema de abastecimiento de agua, la minimización de costes es el primer problema a abordar. Después de conocer en qué orden se deben conectar de forma óptima unos ciertos agentes a la fuente, una segunda tarea consiste en distribuir los costes del sistema entre los agentes involucrados. La asignación de estos costes se lleva a cabo normalmente mediante la propuesta de reglas de asignación, que es el objetivo del presente capítulo para el problema GMS.

En particular, se introduce la regla κ, una regla de asignación de costes para los problemas GMS en árboles. Este mecanismo de asignación sigue un procedimiento algorítmico, y está basado en los algoritmos de resolución del capítulo anterior. Nuestro método toma como punto de partida un orden endógeno y miope, e involucra la determinación de compensaciones, que surgen al pasar del orden de referencia al orden óptimo. Un aspecto que juega un rol fundamental a la hora de determinar estas compensaciones es el intercambio de bloques de agentes escogidos de una cierta forma. Para la elección de estos bloques, es necesario recurrir a los resultados del Capítulo 5. Específicamente, los ingredientes adicionales introducidos en la reformulación de los resultados previos serán la base de la distribución de costes que presentamos en este capítulo. Ilustramos también cómo obtener la regla κ para distintas situaciones por medio de diversos ejemplos.

Los contenidos de este capítulo están recogidos en Davila-Pena et al. (2022a).

Resumo en galego

Os contidos desta tese, titulada *Optimización e cooperación con aplicacións loxísticas*, son o resultado do traballo realizado por Laura Davila Pena durante os estudos correspondentes ao Programa de Doutoramento en Estatística e Investigación Operativa da Universidade de Santiago de Compostela. Este traballo foi realizado en colaboración cos seus directores de tese Ignacio M. García Jurado (Universidade da Coruña) e Balbina V. Casas Méndez (Universidade de Santiago de Compostela), así como cos profesores José Fernando Oliveira e Maria Antónia Carravilla (Universidade do Porto) e Peter Borm (Tilburg University), responsables das estadías de investigación preceptivas para optar á mención internacional, e cos investigadores David Rodríguez Penas (Instituto de Investigacións Mariñas, IIM-CSIC) e Jop Schouten (Tilburg University).

A Investigación Operativa é unha disciplina na que se desenvolven e utilizan métodos analíticos avanzados para resolver problemas e axudar na toma de decisións. Existen unha gran variedade de campos nos que se pode aplicar esta disciplina, un dos cales é a loxística. Esta área é a responsable de xestionar o fluxo de bens, persoas ou recursos entre diferentes localizacións, atendendo a certos requirimentos. Tamén se estende á xestión e planificación de persoal ou recursos en distintos dominios.

O principal obxectivo desta tese é estudar diferentes problemas de Investigación Operativa que poidan ter ou estar relacionados con aplicacións no ámbito loxístico. Está estruturada en tres partes independentes que detallaremos un pouco máis abaixo, e que poñen de manifesto a pluralidade do presente traballo.

O problema de rutas de vehículos (VRP) é un dos problemas loxísticos máis coñecidos e estudados. A súa natureza diversa leva á incorporación de restricións máis ou menos complexas, resultando nunha gran variedade de modelos e obxectivos a perseguir. A primeira parte desta tese céntrase nunha variante do VRP motivada por unha situación real acontecida nunha cooperativa agrícola galega. Introdúcese o modelo a considerar (Capítulo 2) e propóñense diferentes algoritmos de resolución (Capítulo 3).

Ademais do deseño óptimo das rutas, os problemas loxísticos tamén son tratados dende o punto de vista cooperativo. Unha vez creadas as rutas, xorde a pregunta de como distribuír os

custos derivados entre os axentes involucrados. Na terceira parte desta tese estúdase un novo problema de Investigación Operativa no que os custos dun certo sistema deben ser divididos entre os clientes que o compoñen. Isto aparece motivado por unha situación loxística de distribución de auga en áreas urbanas, que será tratada tanto dende a perspectiva da optimización (Capítulo 5) como da asignación de custos (Capítulo 6).

O aspecto cooperativo nos problemas loxísticos tamén se pode estudar facendo uso da teoría de xogos. Esta disciplina pode aplicarse nunha gran cantidade de dominios, destacando nos últimos anos a aprendizaxe automática. Na segunda parte desta tese combinamos estes dous campos. Motivados pola situación de pandemia derivada da COVID-19, proponse un método para avaliar a influencia que certas características de pacientes infectados tiveron na súa evolución (Capítulo 4). Aínda que este problema non trata directamente unha aplicación loxística, si serve como unha ferramenta de soporte á toma de decisións por parte dos profesionais sanitarios, encadrándose dentro da xestión de recursos.

A continuación presentamos os contidos da presente tese máis en profundidade, indicando as súas principais contribucións.

Parte I. Sobre os valiosos problemas de rutas de vehículos

O VRP é un problema de optimización combinatoria que busca atopar unha configuración óptima de rutas percorridas por múltiples vehículos para servir a un conxunto de clientes. Esta parte da tese céntrase nunha variante específica do VRP, chamada problema de rutas con camións e remolques compartimentados (MC-TTRP). Está composta por dous capítulos que presentan o modelo e algoritmos de resolución. No que segue, detallamos os contidos de cada un deles.

Capítulo 2. Problema de rutas con camións e remolques compartimentados (MC-TTRP)

Nos últimos anos, a loxística do transporte xogou un rol fundamental na industria. Moitas compañías, públicas e privadas, están interesadas en desenvolver ferramentas computacionais para deseñar as súas rutas, con obxectivos tales como a minimización dos custos e/ou a maximización dos produtos servidos. Os VRPs son problemas a través dos cales se poden modelar rutas de transporte para distintos vehículos. A resolución destes problemas de optimización matemática é un desafío para os investigadores operacionais. Cando se incorporan novas características de problemas reais, tales como vehículos con capacidade limitada, entrega de mercancía nuns períodos de tempo determinados, ou comportamentos estocásticos,

aparecen novas variantes do VRP orixinal, dando lugar á necesidade de desenvolver novos modelos e técnicas de resolución.

Unha modificación prometedora do VRP é o problema de rutas con camións e remolques (TTRP), que xorde a raíz de restricións de accesibilidade (Chao, 2002). Nesta variante, a frota está composta por camións e remolques que visitan un conxunto de clientes, onde algúns deles (clientes de vehículo, VCs) poden ser atendidos polo vehículo completo (é dicir, o camión co remolque enganchado), mentres que outros só poden ser servidos polo camión (clientes de camión, TCs). Exemplos de TCs son clientes en zonas do interior da cidade, en rexión montañosas, ou en lugares onde a manobra ou acceso cun remolque non é posible. Para resolver este problema, distinguimos tres tipos de rutas: as rutas de camión puras (PTRs), que poden ser percorridas soamente por camións, as rutas de vehículo puras (PVRs), que se realizan enteiramente co vehículo completo e as rutas de vehículo mixtas (MVRs), que consisten nunha ruta principal percorrida polo vehículo completo e unha ou máis subrutas percorridas só polo camión. A Figura RG.1 ilustra unha posible solución ao TTRP.

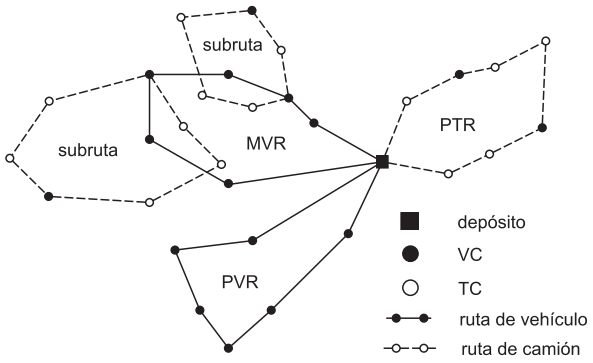


Figura RG.1: Posible solución ao TTRP.

Outra variante interesante do VRP é o caso multi-compartimento, no cal se deben dividir produtos distintos en diferentes almacenaxes durante o transporte. Se ben é certo que a inclusión de compartimentos engade unha complexidade adicional, pode ser un requirimento nalgunhas aplicacións loxísticas reais.

En base ao anteriormente exposto, este capítulo céntrase nun problema de rutas de vehículos que combina o TTRP coa compartimentación dos produtos, ao cal chamamos problema de rutas con camións e remolques compartimentados (MC-TTRP). En particular, propuxemos unha formulación matemática do MC-TTRP como un modelo de programación lineal e enteira mixta (MILP) que abarca todo o problema, algo que non fora realizado anteriormente. A combinación destas dúas características está motivada polas necesidades dunha cooperativa agrícola galega que produce e distribúe penso para o gando (Guitián de Frutos

e Casas-Méndez, 2019). Neste capítulo, presentamos tamén o estudo de caso e usámolo para probar, ilustrar e aplicar a nosa formulación. Utilizamos `AMPL` para a implementación do modelo e o solucionador `Gurobi` para resolvelo.

Os contidos deste capítulo están recollidos en Davila-Pena et al. (2023).

Capítulo 3. Heurísticas para o problema de rutas con camións e remolques compartimentados

No Capítulo 2, empregando os datos da cooperativa agrícola, observamos que obter a solución exacta dun problema con máis de 10 clientes é computacionalmente moi custoso. Isto deriva na necesidade de recorrer a métodos aproximados de resolución, tales como heurísticas. En particular, no presente capítulo, introdúcense e impleméntanse catro algoritmos heurísticos en dúas fases.

A efectividade do algoritmo de aforros de Clarke–Wright (CW) ao construír solucións para distintos VRPs, así como o requirimento de resolver MC-TTRPs que involucran un número relativamente elevado de clientes, serviu de motivación para adaptar este método ao caso do MC-TTRP. Así, a primeira fase das nosas heurísticas coincide para as catro propostas: unha solución inicial é xerada mediante un algoritmo heurístico construtivo baseado no CW.

En canto á segunda fase, propoñemos distintas metaheurísticas para mellorar a solución inicial obtida polo CW. Especificamente, deseñamos catro algoritmos para esta segunda fase: dous deles son completamente independentes, a busca tabu iterativa (ITS) e a busca adaptativa de grandes veciñanzas (ALNS); mentres que os outros dous combinan ideas de ambos, o ALNS híbrido con busca tabú (ALNS-TS) e o ALNS-TS penalizado (PANLS-TS).

Tanto as fases construtiva como de mellora están integradas nuns algoritmos novidosos en dúas fases para resolver o MC-TTRP. A implementación destes métodos realizouse no programa informático `R`. Levouse a cabo un exhaustivo estudo computacional sobre diferentes conxuntos de datos. En primeiro lugar, adaptamos adecuadamente os 21 problemas de referencia do TTRP (Chao, 2002) para considerar compartimentos, dando lugar a 21 novas instancias. As nosas heurísticas obtiveron solucións factibles para todos estes problemas, e os resultados acadados fan evidente o efectivo e eficiente comportamento destes métodos ao resolver o MC-TTRP. Realizamos un estudo comparativo destas catro metaheurísticas e observamos que o ITS era a que chegaba a mellores solucións. De acordo con isto, levouse a cabo unha análise máis profunda deste algoritmo. Primeiramente, e co fin de obter unha validación inicial do mesmo, decidimos adaptar o noso método ao caso do TTRP. Desta forma, é posible comparar os resultados con técnicas de resolución xa existentes na literatura para o TTRP, e observamos que os resultados acadados polo ITS eran competitivos cos métodos

previos, chegando incluso a obter mellores solucións para algúns dos problemas. Tamén se realizou un estudo máis detallado nas instancias xeradas. Ademais, este algoritmo aplicouse ao caso real da cooperativa agrícola, co fin de comparar estes resultados coa solución óptima proporcionada pola nosa formulación exacta. Obtivéronse solucións de boa calidade nun tempo computacional moito menor. Con todo isto, consideramos o ITS como un método de resolución prometedor para o MC-TTRP.

Os contidos deste capítulo están recollidos en Davila-Pena et al. (2023) e Davila-Pena et al. (2022c).

Parte II. Sobre a teoría de xogos nos problemas de clasificación

Existen moitas situacións nas que necesitamos anticipar un certo resultado, en base a unhas entradas específicas. Este tipo de problemas aparecen nunha gran cantidade de campos tales como ciencia, economía ou sanidade. Por exemplo, podemos estar interesados en predicir os beneficios dunha certa compañía ou en saber se un determinado paciente sofre ou non unha enfermidade. Este último caso, onde a resposta é categórica, encádrase dentro dos problemas de clasificación.

Así, un problema de clasificación implica predicir a categoría á que pertence un certo individuo, en función dos seus atributos. Estas predicións están baseadas no coñecemento obtido a través dunha mostra de individuos para os que se coñecen os valores dos atributos e da resposta. Neste tipo de problemas, ademais de clasificar, é importante coñecer como de influentes son cada un destes atributos na resposta. Esta segunda parte da tese está composta por un único capítulo que presenta unha medida para avaliar dita influencia baseándose en ideas da teoría de xogos.

Capítulo 4. Unha medida de influencia para os problemas de clasificación

Un dos elementos fundamentais ao tratar problemas de clasificación é dispoñer dunha mostra de adestramento, é dicir, un conxunto de observacións para as que se coñece a súa categoría e os valores dos atributos. Neste capítulo faise uso de técnicas de clasificación importadas da aprendizaxe automática que permitirán desenvolver unha ferramenta metodolóxica para a análise exploratoria dunha mostra de adestramento como a que acabamos de mencionar. Algúns clasificadores teñen a vantaxe de proporcionar puntuacións para as variables explicativas nos problemas de clasificación, é dicir, de avaliar a súa importancia. Estas puntuacións

dependerán do clasificador en cuestión. Traballos como Štrumbelj e Kononenko (2010) ou Datta et al. (2015) propoñen métodos agnósticos ao modelo baseados en solucións da teoría de xogos cooperativa. O enfoque de Štrumbelj e Kononenko (2010) serve para avaliar a influencia que os atributos dun certo individuo tiveron na súa clase asignada. Datta et al. (2015) non se centra nun único individuo, é dicir, a súa proposta non está localmente orientada, senón que considera unha mostra de individuos e fundamenta teoricamente o seu método no escenario binario. No noso caso, xeneralizamos estes procedementos por medio do deseño dunha medida de influencia que está orientada globalmente e é independente do clasificador utilizado, que nos permitirá avaliar como de importantes son os distintos valores dos atributos na clasificación dun conxunto de individuos. Ata onde coñecemos, este enfoque específico non fora adoptado previamente na literatura.

Para construír a nosa medida de influencia, comezamos definindo un xogo cooperativo (cuxos xogadores son os atributos considerados) e logo aplicamos unha solución da teoría de xogos, coñecida coma o valor de Shapley. Realízase tamén unha caracterización axiomática da medida proposta por medio de propiedades adaptadas daquelas que verifica o valor de Shapley no contexto xeral da teoría de xogos, e que son altamente desexables dende o punto de vista da análise exploratoria de datos: eficiencia e contribucións equilibradas. Para probar o alcance e adecuación da nosa medida de influencia, lévase a cabo un experimento de control que dá lugar a uns resultados satisfactorios. Ademais, efectúase tamén unha comparativa entre a nosa proposta e a de Datta et al. (2015), dando lugar o noso método a resultados máis adecuados dende a perspectiva da interpretabilidade. Ambas ferramentas foron implementadas no programa informático `R`.

Finalmente, aplicamos a nosa medida de influencia a un problema real no que nos vimos inmersos recentemente. Durante a primeira ola da pandemia da COVID-19, tivemos acceso a unha base de datos de $10\,454$ pacientes de Galicia infectados entre o 6 de marzo e o 7 de maio de 2020. Coñecer as características que poden incrementar significativamente a súa probabilidade de necesitar acceso a certos servizos sanitarios é altamente útil para que as autoridades sanitarias tomen as decisións máis acertadas. Deste modo, empregamos estes datos para atopar cales eran os valores das características dos pacientes que máis afectaban ao empeoramento da súa condición, e que podían dar lugar ao seu falecemento, ao seu ingreso na unidade de coidados intensivos (ICU), ou á súa hospitalización. Os atributos considerados foron factores demográficos (idade e sexo) e certas patoloxías previas (enfermidades cardíacas, respiratorias, metabólicas e urinarias), discretizados en distintos niveis. Os resultados obtidos son coherentes coas situacións que realmente se produciron durante a pandemia no período de tempo e rexión xeográfica estudados. Por exemplo, obsérvase que os pacientes de máis

idade son máis propensos a ser hospitalizados, pero non a ser enviados á ICU, probablemente porque moitos deles faleceron antes do seu ingreso.

En base ao anterior, o noso método presenta un prometedor futuro como instrumento de apoio á toma de decisións. Serve, en particular, para alertar aos profesionais sanitarios da importancia de ter en conta certas características dos pacientes, fronte á menor importancia ou influencia doutras. Tales características poden representar un potencial risco para pacientes cunha certa enfermidade, que en consecuencia deben ser tomados en conta tanto con respecto aos coidados que deben recibir como na planificación dos recursos destinados para eles.

Os contidos deste capítulo están recollidos en Davila-Pena et al. (2022b).

Parte III. Sobre os problemas cooperativos de Investigación Operativa

Os problemas de árbores de expansión de mínimo custo (MCSTPs) e os problemas de secuenciación son problemas multiaxente de Investigación Operativa que perseguen dous obxectivos: por un lado, minimizar os custos relativos á construción ou deseño dunha rede ou sistema; e por outro, distribuír eses custos entre os axentes involucrados. Unha combinación de ideas destes dous problemas dá lugar ao problema de planificación de máquinas en grafos (problema GMS), onde o obxectivo perseguido consta tamén de dúas fases. A terceira parte desta tese céntrase neste problema e está formada por dous capítulos que presentan e estudan o modelo dende a perspectiva da optimización e da asignación de custos.

Capítulo 5. O problema de planificación de máquinas en grafos (problema GMS)

Tal e como se presenta en Bergantiños et al. (2014), existen numerosos problemas reais que requiren a construción de infraestruturas para conectar un conxunto de axentes a unha fonte, ben directa ou indirectamente. Un deles é o abastecemento urbano de auga dende un depósito central a determinados puntos de interese (axentes), que involucra a construción de condutos por toda unha cidade. A instalación destes condutos entre dous puntos require un certo tempo. O primeiro problema que xorde neste tipo de situacións é a cuestión de onde colocar cada un destes condutos. O obxectivo sería, así, conectar todos os axentes á rede de tal forma que se minimice o tempo total de instalación ou construción. Para tratar este problema, o MCSTP foi amplamente utilizado.

Non obstante, é a miúdo esencial que os axentes dispoñan de subministración de auga en

todo momento (por exemplo, un hospital), polo que deben contratar unha compañía externa que ofreza este servizo mentres non poidan ser abastecidos dende o depósito central. Deste modo, cada axente ten un coeficiente asociado que indica o custo por unidade do tempo no sistema, é dicir, por unidade de tempo na que os condutos que o conectan á fonte aínda non están instalados. Polo tanto, o custo dun axente virá determinado polo seu coeficiente por unidade de tempo multiplicado polo tempo total requirido para conectar ese axente específico. Este tempo total dependerá do momento no que este axente se conecte á fonte: por exemplo, se o axente 2 se conecta á fonte por medio do axente 1, entón débese construír primeiro o conduto que conecta o axente 1 á fonte, seguido polo conduto que conecta o axente 1 ao axente 2. Así, o tempo total requirido para conectar o axente 2 á fonte será a suma dos tempos de construción dos condutos que conectan a fonte ao axente 1, e o axente 1 ao axente 2. O obxectivo será, por tanto, minimizar os custos totais en lugar do tempo de construción total da rede.

Situacións como a descrita arriba resultan nun novo tipo de problema, ao que denominamos problema GMS. Un aspecto que nos gustaría subliñar de novo é a proximidade do noso problema a outros dous problemas estudados na literatura. Primeiro, o problema GMS está estreitamente ligado ao MCSTP. Non obstante, no problema GMS os custos calcúlanse dun xeito completamente diferente. En particular, a orde na que se constrúen os condutos ten un efecto substancial no custo no noso problema, mentres que esta orde é irrelevante ao calcular o custo nun MCSTP. Segundo, o problema GMS está profundamente relacionado cos problemas de secuenciación con restricións de precedencia. Aínda que os problemas de secuenciación con relacións de precedencia foron tratadas na literatura (Hamers et al., 2005), o enfoque baixo o que estudaremos aquí o noso problema nunca fora adoptado, ata onde coñecemos.

Agora ben, neste capítulo centrámonos no estudo do problema GMS dende o punto de vista da optimización. Comezamos mostrando que resolver de forma óptima un problema GMS é en xeral complexo, polo que consideramos situacións nas que a estrutura inicial é unha árbore. Así, o obxectivo consiste en atopar unha orde óptima na que abastecer aos axentes, de tal forma que se minimicen os custos totais tendo en conta as relacións de precedencia impostas por dita estrutura. Lévase a cabo a reformulación de resultados previos da literatura e propóñense algoritmos de resolución máis sofisticados cá os existentes, para os cales se proba a súa optimalidade. A introdución de novos ingredientes é fundamental para o estudo do problema GMS dende a perspectiva da asignación de custos, que será tratado no seguinte capítulo. Ademais, ilústrase mediante numerosos exemplos como aplicar estes resultados a unha variedade de problemas.

Os contidos deste capítulo están recollidos en Davila-Pena et al. (2022a).

Capítulo 6. Unha regra de asignación para os problemas GMS

Ao construír un sistema de abastecemento de auga, a minimización de custos é o primeiro problema a abordar. Despois de coñecer en que orde se deben conectar de xeito óptimo uns certos axentes á fonte, unha segunda tarefa consiste en distribuír os custos do sistema entre os axentes involucrados. A asignación destes custos lévase a cabo normalmente mediante a proposta de regras de asignación, que é o obxectivo do presente capítulo para o problema GMS.

En particular, introdúcese a regra κ, unha regra de asignación de custos para os problemas GMS en árbores. Este mecanismo de asignación segue un procedemento algorítmico, e está baseado nos algoritmos de resolución do capítulo anterior. O noso método toma como punto de partida unha orde endóxena e miope, e involucra a determinación de compensacións, que xorden ao pasar da orde de referencia á orde óptima. Un aspecto que xoga un rol fundamental á hora de determinar estas compensacións é o intercambio de bloques de axentes escollidos dunha certa forma. Para a elección destes bloques, é necesario recorrer aos resultados do Capítulo 5. Especificamente, os ingredientes adicionais introducidos na reformulación dos resultados previos serán a base da distribución de custos que presentamos neste capítulo. Ilustramos tamén como obter a regra κ para distintas situacións por medio de diversos exemplos.

Os contidos deste capítulo están recollidos en Davila-Pena et al. (2022a).

References

Accorsi, L. and Vigo, D. (2020). A hybrid metaheuristic for single truck and trailer routing problems. *Transportation Science*, 54(5):1351–1371.

Alatartsev, S., Stellmacher, S., and Ortmeier, F. (2015). Robotic task sequencing problem: A survey. *Journal of Intelligent & Robotic Systems*, 80(2):279–298.

Algaba, E., Fragnelli, V., Llorca, N., and Sánchez-Soriano, J. (2019a). Horizontal cooperation in a multimodal public transport system: The profit allocation problem. *European Journal of Operational Research*, 275(2):659–665.

Algaba, E., Fragnelli, V., and Sánchez-Soriano, J. (2019b). *Handbook of the Shapley Value*. CRC Press, Taylor & Francis Group, Boca Raton, FL, 1 edition.

Alidaee, B. and Ahmadian, A. (1993). Two parallel machine sequencing problems involving controllable job processing times. *European Journal of Operational Research*, 70(3):335–341.

Alinaghian, M. and Shokouhi, N. (2018). Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, 76:85–99.

Alvarez, A., Munari, P., and Morabito, R. (2018). Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, 25(6):1785–1809.

Amorim, P., Parragh, S. N., Sperandio, F., and Almada-Lobo, B. (2014). A rich vehicle routing problem dealing with perishable food: a case study. *Top*, 22(2):489–508.

Amorosi, L., Puerto, J., and Valverde, C. (2021). Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs. *Computers & Operations Research*, 136:105445.

Asawarungsaengkul, K., Rattanamanee, T., and Wuttipornpun, T. (2013). A multi-size compartment vehicle routing problem for multi-product distribution: Models and solution procedures. *International Journal of Artificial Intelligence*, 11(A13):237–256.

Baker, K. R. (1971). *Single machine sequencing with weighting factors and precedence constraints*. University of Michigan. Department of Industrial Engineering. *Unpublished work*.

Baker, K. R. and Su, Z.-S. (1974). Sequencing with due-dates and early start times to minimize maximum tardiness. *Naval Research Logistics Quarterly*, 21(1):171–176.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Bergantiños, G. and Gómez-Rúa, M. (2010). Minimum cost spanning tree problems with groups. *Economic Theory*, 43(2):227–262.

Bergantiños, G. and Gómez-Rúa, M. (2015). An axiomatic approach in minimum cost spanning tree problems with groups. *Annals of Operations Research*, 225(1):45–63.

Bergantiños, G., Gómez-Rúa, M., Llorca, N., Pulido, M., and Sánchez-Soriano, J. (2014). A new rule for source connection problems. *European Journal of Operational Research*, 234(3):780–788.

Bergantiños, G. and Kar, A. (2010). On obligation rules for minimum cost spanning tree problems. *Games and Economic Behavior*, 69(2):224–237.

Bergantiños, G. and Lorenzo, L. (2004). A non-cooperative approach to the cost spanning tree problem. *Mathematical Methods of Operations Research*, 59(3):393–403.

Bergantiños, G., Lorenzo, L., and Lorenzo-Freire, S. (2010). The family of cost monotonic and cost additive rules in minimum cost spanning tree problems. *Social Choice and Welfare*, 34(4):695–710.

Bergantiños, G., Lorenzo, L., and Lorenzo-Freire, S. (2011). A generalization of obligation rules for minimum cost spanning tree problems. *European Journal of Operational Research*, 211(1):122–129.

Bergantiños, G. and Lorenzo-Freire, S. (2008a). A characterization of optimistic weighted Shapley rules in minimum cost spanning tree problems. *Economic Theory*, 35(3):523–538.

Bergantiños, G. and Lorenzo-Freire, S. (2008b). "Optimistic" weighted Shapley rules in minimum cost spanning tree problems. *European Journal of Operational Research*, 185(1):289–298.

Bergantiños, G. and Vidal-Puga, J. (2007a). A fair rule in minimum cost spanning tree problems. *Journal of Economic Theory*, 137(1):326–352.

Bergantiños, G. and Vidal-Puga, J. (2007b). The optimistic TU game in minimum cost spanning tree problems. *International Journal of Game Theory*, 36(2):223–239.

Bergantiños, G. and Vidal-Puga, J. (2011). The folk solution and Boruvka's algorithm in minimum cost spanning tree problems. *Discrete Applied Mathematics*, 159(12):1279–1283.

Bergantiños, G. and Vidal-Puga, J. (2015). Characterization of monotonic rules in minimum cost spanning tree problems. *International Journal of Game Theory*, 44:835–868.

Bergantiños, G. and Vidal-Puga, J. (2021). A review of cooperative rules and their associated algorithms for minimum-cost spanning tree problems. *SERIEs*, 12(1):73–100.

Bird, C. G. (1976). On cost allocation for a spanning tree: A game theoretic approach. *Networks*, 6(4):335–350.

Borm, P., Fiestras-Janeiro, G., Hamers, H., Sánchez, E., and Voorneveld, M. (2002). On the convexity of games corresponding to sequencing situations with due dates. *European Journal of Operational Research*, 136(3):616–634.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Brown, G. G., Ellis, C. J., Graves, G. W., and Ronen, D. (1987). Real-time, wide area dispatch of mobil tank trucks. *Interfaces*, 17(1):107–120.

Brown, G. G. and Graves, G. W. (1981). Real-time dispatch of petroleum tank trucks. *Management Science*, 27(1):19–32.

Burkart, N. and Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317.

Cabo, M., Possani, E., Potts, C. N., and Song, X. (2015). Split–merge: Using exponential neighborhood search for scheduling a batching machine. *Computers & Operations Research*, 63:125–135.

Calleja, P., Borm, P., Hamers, H., Klijn, F., and Slikker, M. (2002). On a new class of parallel sequencing situations and related games. *Annals of Operations Research*, 109(1):265–277.

Calleja, P., Estévez-Fernández, A., Borm, P., and Hamers, H. (2006). Job scheduling, cooperation, and control. *Operations Research Letters*, 34(1):22–28.

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

Caramia, M. and Guerriero, F. (2010b). A milk collection problem with incompatibility constraints. *Interfaces*, 40(2):130–143.

Carpente, L., Casas-Méndez, B., Jácome, C., and Puerto, J. (2010). A model and two heuristic approaches for a forage harvester planning problem: a case study. *Top*, 18(1):122–139.

Carrizosa, E., Molero-Río, C., and Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. *Top*, 29:5–33.

Casalicchio, G., Molnar, C., and Bischl, B. (2019). Visualizing the feature importance for black box models. In Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., and Ifrim, G., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 655–670. Springer, Cham.

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

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1998). A new algorithm for the site-dependent vehicle routing problem. In Woodruff, D. L., editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristics Search: Interfaces in Computer Science and Operations Research*, pages 301–312. Springer, Boston, MA.

Chen, H., Covert, I. C., Lundberg, S. M., and Lee, S.-I. (2022). Algorithms to estimate Shapley value feature attributions. *arXiv e-prints*, page arXiv:2207.07605.

Cheng, T. C. E. and Gupta, M. C. (1989). Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, 38(2):156–166.

Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In Christofides, N., Mingozzi, A., Toth, P., and Sandi, C., editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester.

Çiftçi, B., Borm, P., Hamers, H., and Slikker, M. (2013). Batch sequencing and cooperation. *Journal of Scheduling*, 16(4):405–415.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Claus, A. and Kleitman, D. J. (1973). Cost allocation for a spanning tree. *Networks*, 3(4):289–304.

Coelho, L. C. and Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research*, 242(3):854–864.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S. (2005). New heuristics for the vehicle routing problem. In Langevin, A. and Riopel, D., editors, *Logistics Systems: Design and Optimization*, pages 279–297. Springer, Boston, MA.

Cordeau, J.-F. and Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050.

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

Curiel, I. (1997). Minimum cost spanning tree games. In *Cooperative Game Theory and Applications*, pages 129–148. Springer, Boston, MA.

Curiel, I. (2015). Compensation rules for multi-stage sequencing games. *Annals of Operations Research*, 225(1):65–82.

Curiel, I., Pederzoli, G., and Tijs, S. (1989). Sequencing games. *European Journal of Operational Research*, 40(3):344–351.

Dantzig, G. B., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Datta, A., Datta, A., Procaccia, A. D., and Zick, Y. (2015). Influence in classification via cooperative game theory. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 511–517.

Davila-Pena, L. (2019). Modelos y algoritmos en una clase de problemas de rutas de vehículos. Master's thesis, University of Santiago de Compostela.

Davila-Pena, L., Borm, P., García-Jurado, I., and Schouten, J. (2022a). An allocation rule for the graph machine scheduling problem. *Manuscript under preparation*.

Davila-Pena, L., García-Jurado, I., and Casas-Méndez, B. (2022b). Assessment of the influence of features on a classification problem: An application to COVID-19 patients. *European Journal of Operational Research*, 299(2):631–641.

Davila-Pena, L., R. Penas, D., and Casas-Méndez, B. (2023). A new two-phase heuristic for a problem of food distribution with compartmentalized trucks and trailers. *International Transactions in Operational Research*, 30(2):1031–1064.

Davila-Pena, L., R. Penas, D., Casas-Méndez, B., Carravilla, M. A., and Oliveira, J. F. (2022c). An adaptive large neighborhood search for the multi-compartment truck and trailer routing problem. *Manuscript under preparation*.

de Morais, C. S., Ramos Jorge, D. R., Aguiar, A. R., Barbosa-Póvoa, A. P., Antunes, A. P., and Ramos, T. R. P. (2022). A solution methodology for a Smart Waste Collection Routing Problem with workload concerns: computational and managerial insights from a real case study. *International Journal of Systems Science: Operations & Logistics*, pages 1–31.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., and Vogel, U. (2011). Vehicle routing with compartments: applications, modelling and heuristics. *OR Spectrum*, 33(4):885–914.

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

Drexl, M. (2021). On the one-to-one pickup-and-delivery problem with time windows and trailers. *Central European Journal of Operations Research*, 29(3):1115–1162.

Dutta, B. and Kar, A. (2004). Cost monotonicity, consistency and minimum cost spanning tree games. *Games and Economic Behavior*, 48(2):223–248.

El Fallahi, A., Prins, C., and Calvo, R. W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 35(5):1725–1741.

Estañ, T., Llorca, N., Martínez, R., and Sánchez-Soriano, J. (2021). On how to allocate the fixed cost of transport systems. *Annals of Operations Research*, 301(1):81–105.

Estévez-Fernández, A., Borm, P., Calleja, P., and Hamers, H. (2008). Sequencing games with repeated players. *Annals of Operations Research*, 158(1):189–203.

Estévez-Fernández, A. and Reijnierse, H. (2014). On the core of cost-revenue games: Minimum cost spanning tree games with revenues. *European Journal of Operational Research*, 237(2):606–616.

Feltkamp, V., Tijs, S., and Muto, S. (1994). On the irreducible core and the equal remaining obligations rule of minimum cost spanning extension problems. *Center for Economic Research*, 106. Tilburg University.

Fernández, F. R., Hinojosa, M. A., and Puerto, J. (2004). Multi-criteria minimum cost spanning tree games. *European Journal of Operational Research*, 158(2):399–408.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181.

Gayialis, S. P., Kechagias, E. P., and Konstantakopoulos, G. D. (2022). A city logistics system for freight transportation: Integrating information technology and operational research. *Operational Research*, 22(5):5953–5982.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

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

Ghaddar, B. and Naoum-Sawaya, J. (2018). High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research*, 265(3):993–1004.

Giménez-Palacios, I., Parreño, F., Álvarez-Valdés, R., Paquay, C., Oliveira, B. B., Carravilla, M. A., and Oliveira, J. F. (2022). First-mile logistics parcel pickup: Vehicle routing with packing constraints under disruption. *Transportation Research Part E: Logistics and Transportation Review*, 164:102812.

Glover, F. and Laguna, M. (1998). Tabu search. In Du, D.-Z. and Pardalos, P. M., editors, *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer, Boston, MA.

Gómez-Rúa, M. and Vidal-Puga, J. (2017). A monotonic and merge-proof rule in minimum cost spanning tree situations. *Economic Theory*, 63(3):813–826.

González-Díaz, J., García-Jurado, I., and Fiestras-Janeiro, M. G. (2010). *An Introductory Course on Mathematical Game Theory*. American Mathematical Society, Providence, RI, 1 edition.

Granot, D. and Huberman, G. (1981). Minimum cost spanning tree games. *Mathematical Programming*, 21(1):1–18.

Granot, D. and Huberman, G. (1984). On the core and nucleolus of the minimum cost spanning tree games. *Mathematical Programming*, 29(3):323–347.

Guitián de Frutos, R. M. and Casas-Méndez, B. (2019). Routing problems in agricultural cooperatives: a model for optimization of transport vehicle logistics. *IMA Journal of Management Mathematics*, 30(4):387–412.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

Hachicha, H. K., Rebah, S. B., and Layeb, S. B. (2019). A new MILP-based decision support system for the fuel distribution. In *2019 7th International conference on ICT & Accessibility (ICTA)*, pages 1–6.

Hamers, H., Klijn, F., and van Velzen, B. (2005). On the convexity of precedence sequencing games. *Annals of Operations Research*, 137(1):161–175.

Hamers, H., Suijs, J., Tijs, S., and Borm, P. (1996). The split core for sequencing games. *Games and Economic Behavior*, 15(2):165–176.

Henke, T., Speranza, M. G., and Wäscher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, 246(3):730–743.

Henke, T., Speranza, M. G., and Wäscher, G. (2019). A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. *Annals of Operations Research*, 275(2):321–338.

Jothi, N., Husain, W., and Rashid, N. A. (2021). Predicting generalized anxiety disorder among women using Shapley value. *Journal of Infection and Public Health*, 14(1):103–108.

Kar, A. (2002). Axiomatization of the Shapley value on minimum cost spanning tree games. *Games and Economic Behavior*, 38(2):265–277.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Lee, C.-Y. (1991). Parallel machines scheduling with nonsimultaneous machine available time. *Discrete Applied Mathematics*, 30(1):53–61.

Lespay, H. and Suchan, K. (2021). A case study of consistent vehicle routing problem with time windows. *International Transactions in Operational Research*, 28(3):1135–1163.

Li, T. and Chen, J. (2020). Alliance formation in assembly systems with quality-improvement incentives. *European Journal of Operational Research*, 285(3):931–940.

Lin, S.-W., Yu, V. F., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692.

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

Liu, D., Ji, X., Tang, J., and Li, H. (2020). A fuzzy cooperative game theoretic approach for multinational water resource spatiotemporal allocation. *European Journal of Operational Research*, 282(3):1025–1037.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc., Red Hook, NY.

Méndez-Fernández, I., Lorenzo-Freire, S., García-Jurado, I., Costa, J., and Carpente, L. (2020). A heuristic approach to the task planning problem in a home care business. *Health Care Management Science*, 23(4):556–570.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37(11):1886–1898.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2011). Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science*, 45(3):346–363.

Merrick, L. and Taly, A. (2020). The explanation game: Explaining machine learning models using Shapley values. In Holzinger, A., Kieseberg, P., Tjoa, A. M., and Weippl, E., editors, *Machine Learning and Knowledge Extraction*, pages 17–38. Springer, Cham.

Moretti, S., Branzei, R., Norde, H., and Tijs, S. (2004). The P-value for cost sharing in minimum cost spanning tree situations. *Theory and Decision*, 56(1):47–61.

Musegaas, M., Borm, P., and Quant, M. (2018). On the convexity of step out–step in sequencing games. *Top*, 26(1):68–109.

Muyldermans, L. and Pang, G. (2010). On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research*, 206(1):93–103.

Myerson, R. B. (1980). Conference structures and fair allocation rules. *International Journal of Game Theory*, 9(3):169–182.

Neves-Moreira, F., Amorim-Lopes, M., and Amorim, P. (2020). The multi-period vehicle routing problem with refueling decisions: Traveling further to decrease fuel cost? *Transportation Research Part E: Logistics and Transportation Review*, 133:101817.

Norde, H., Moretti, S., and Tijs, S. (2004). Minimum cost spanning tree games and population monotonic allocation schemes. *European Journal of Operational Research*, 154(1):84–97.

Ostermeier, M., Henke, T., Hübner, A., and Wäscher, G. (2020). Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions. *European Journal of Operational Research*, 292(3):799–817.

Ostermeier, M. and Hübner, A. (2018). Vehicle selection for a multi-compartment vehicle routing problem. *European Journal of Operational Research*, 269(2):682–694.

Ostermeier, M., Martins, S., Amorim, P., and Hübner, A. (2018). Loading constraints for a multi-compartment vehicle routing problem. *OR Spectrum*, 40(4):997–1027.

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

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401.

Rais, A. and Viana, A. (2011). Operations Research in Healthcare: a survey. *International Transactions in Operational Research*, 18(1):1–31.

Renaud, J., Boctor, F. F., and Laporte, G. (1996). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8(2):134–143.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 52(5):1174–1190.

Saavedra-Nieves, A. and Saavedra-Nieves, P. (2020). On systems of quotas from bankruptcy perspective: the sampling estimation of the random arrival rule. *European Journal of Operational Research*, 285(2):655–669.

Saavedra-Nieves, A., Schouten, J., and Borm, P. (2020). On interactive sequencing situations with exponential cost functions. *European Journal of Operational Research*, 280(1):78–89.

Santos, M. J., Amorim, P., Marques, A., Carvalho, A., and Póvoa, A. (2020). The vehicle routing problem with backhauls towards a sustainability perspective: a review. *Top*, 28(2):358–401.

Scheuerer, S. (2004). *Neue Tabusuche-Heuristiken für die logistische Tourenplanung bei restringierendem Anhängereinsatz, mehreren Depots und Planungsperioden*. PhD thesis, University of Regensburg.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1–15.

Schouten, J., Saavedra-Nieves, A., and Fiestras-Janeiro, M. G. (2021). Sequencing situations and games with non-linear cost functions under optimal order consistency. *European Journal of Operational Research*, 294(2):734–745.

Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, 61(1):45–65.

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

Serafini, P. (1996). Scheduling jobs on several machines with the job splitting property. *Operations Research*, 44(4):617–628.

Shapley, L. S. (1953). A value for n-person games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games (AM-28)*, volume II, pages 307–318. Princeton University Press, Princeton, NJ.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming — CP98*, pages 417–431. Springer, Berlin, Heidelberg.

Shen, L., Wang, D., and Wang, X.-Y. (2013). Parallel-machine scheduling with non-simultaneous machine available time. *Applied Mathematical Modelling*, 37(7):5227–5232.

Sidney, J. B. (1975). Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23(2):283–298.

Silvestrin, P. V. and Ritt, M. (2017). An iterated tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 81:192–202.

Slikker, M. (2005). Balancedness of sequencing games with multiple parallel machines. *Annals of Operations Research*, 137(1):177–189.

Smith, M. and Alvarez, F. (2021). Identifying mortality factors from Machine Learning using Shapley values – a case of COVID19. *Expert Systems with Applications*, 176:114832.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24.

Smith-Miles, K., Muñoz, M. A., and Neelofar (2019). MATILDA: Melbourne Algorithm Test Instance Library with Data Analytics. https://matilda.unimelb.edu.au. Accessed: 2022-11-09.

Speranza, M. G. (2018). Trends in transportation and logistics. *European Journal of Operational Research*, 264(3):830–836.

Štrumbelj, E. and Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18.

Štrumbelj, E. and Kononenko, I. (2011). A general method for visualizing and explaining black-box regression models. In Dobnikar, A., Lotrič, U., and Šter, B., editors, *Adaptive and Natural Computing Algorithms*, pages 21–30. Springer, Berlin, Heidelberg.

Štrumbelj, E., Kononenko, I., and Šikonja, M. R. (2009). Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68(10):886–904.

Tijs, S., Moretti, S., Branzei, R., and Norde, H. (2006). The Bird core for minimum cost spanning tree problems revisited: monotonicity and additivity aspects. In Seeger, A., editor, *Recent Advances in Optimization*, pages 305–322. Springer, Berlin, Heidelberg.

Tijs, S., Parthasarathy, T., Potters, J., and Prasad, V. R. (1984). Permutation games: Another class of totally balanced games. *Operations-Research-Spektrum*, 6(2):119–123.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1 edition.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. Mathematical Optimization Society and Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition.

Further information

In compliance with the rules of doctoral studies at Universidade de Santiago de Compostela collected in *Regulamento dos estudos de doutoramento na USC, DOG de 16 de setembro de 2020*, we provide some information regarding the articles that support the contributions of this thesis, as well as information about the journals in which such works are published. In particular, we give the authors' names, journals' titles, publishing years, ISSN (or E-ISSN), publishers, DOI-type links, Journal Impact Factor and quartile from the Journal Citation Reports, CiteScore rating and quartile from Scopus, in addition to the specific contributions of the PhD candidate. Furthermore, some relevant information regarding copyright and use of the published articles is given. The links have been checked on November 10, 2022.

Articles and journals

Information about the works on which this dissertation is based is given in this section. First, details about the journals in which some of the works are published are provided. A list of unpublished works is given afterward.

Published works

The following works, corresponding to parts of Chapters 2–4, have been published. Information about the articles, journals, and copyright permissions is provided hereafter.

Chapters 2 and 3: Davila-Pena et al. (2023)

TITLE: A new two-phase heuristic for a problem of food distribution with compartmentalized trucks and trailers.

AUTHORS: L. Davila-Pena[1,2], D. R. Penas[1,2], and B. Casas-Méndez[1,2].
AFFILIATIONS:

[1] CITMAga.

[2] Departamento de Estatística, Análise Matemática e Optimización, Universidade de Santiago de Compostela.

JOURNAL: International Transactions in Operational Research.

YEAR: 2023.

E-ISSN: 1475-3995.

PUBLISHER: John Wiley and Sons.

LINK: https://doi.org/10.1111/itor.13071.

JCR IMPACT FACTOR: The data from 2023 is still not available. The data from 2021: 3.610 [Q2 in Operations Research & Management Science (28/87)].

CITESCORE: The data from 2023 is still not available. The data from 2021: 7.4 [Q1 in Management Science and Operations Research (25/184)].

CONTRIBUTIONS OF THE PHD CANDIDATE: The candidate contributed to the conceptualization of the work, methodology, software implementation, formal analysis of the results, and to the manuscript preparation.

Chapter 4: Davila-Pena et al. (2022b)

TITLE: Assessment of the influence of features on a classification problem: An application to COVID-19 patients.

AUTHORS: L. Davila-Pena[1,2], I. García-Jurado[1,3], and B. Casas-Méndez[1,2].

AFFILIATIONS:

[1] CITMAga.

[2] Departamento de Estatística, Análise Matemática e Optimización, Universidade de Santiago de Compostela.

[3] Departamento de Matemáticas, Universidade da Coruña.

JOURNAL: European Journal of Operational Research.

YEAR: 2022.

ISSN: 0377-2217.

PUBLISHER: Elsevier.

LINK: https://doi.org/10.1016/j.ejor.2021.09.027.

JCR IMPACT FACTOR: The data from 2022 is still not available. The data from 2021: 6.363 [Q1 in Operations Research & Management Science (17/87)].

CITESCORE: The data from 2022 is still not available. The data from 2021: 10.5 [Q1 in Management Science and Operations Research (11/184)]

CONTRIBUTIONS OF THE PHD CANDIDATE: The candidate contributed to the conceptualization of the work, methodology, proofs of the theoretical results, software implementation, formal analysis of the results, and to the manuscript preparation.

Unpublished works

The following works, corresponding to parts of Chapters 3, 5, and 6 of the thesis, are not published but are being considered for publication by JCR journals.

Chapter 3: Davila-Pena et al. (2022c)

TITLE: An adaptive large neighborhood search for the multi-compartment truck and trailer routing problem.

AUTHORS: L. Davila-Pena[1,2], D. R. Penas[1,2,3], B. Casas-Méndez[1,2], M. A. Carravilla[4], and J. F. Oliveira[4].

AFFILIATIONS:

[1] CITMAga.

[2] Departamento de Estatística, Análise Matemática e Optimización, Universidade de Santiago de Compostela.

[3] Computational Biology Lab, MBG-CSIC.

[4] INESC TEC, Faculdade de Engenharia, Universidade do Porto.

CONTRIBUTIONS OF THE PHD CANDIDATE: The candidate contributed to the conceptualization of the work, methodology, software implementation, formal analysis of the results, and to the manuscript preparation.

Chapters 5 and 6: Davila-Pena et al. (2022a)

TITLE: An allocation rule for the graph machine scheduling problem.

AUTHORS: L. Davila-Pena[1,2], P. Borm[3], I. García-Jurado[1,4], and J. Schouten[3].

AFFILIATIONS:

[1] CITMAga.

[2] Departamento de Estatística, Análise Matemática e Optimización, Universidade de Santiago de Compostela.

[3] CentER, Department of Econometrics and Operations Research, Tilburg University.

[4] Departamento de Matemáticas, Universidade da Coruña.

CONTRIBUTIONS OF THE PHD CANDIDATE: The candidate contributed to the conceptualization of the work, methodology, proofs of the theoretical results, formal analysis of the results, and to the manuscripts preparation.

Operations Research (OR) is a discipline in which advanced analytical methods are developed and used to solve problems and aid in decision-making. There are a wide variety of fields where OR can be applied, one of which is logistics. Optimizing and scheduling a certain network is a fundamental task, accomplished by proposing models and solving algorithms for each particular situation. Logistic problems are also often treated by the practitioners of OR from a cooperative standpoint, which can be approached through game theory. This thesis addresses several OR and game theoretical problems that have applications in the logistics domain. The performance of the proposed methodologies is analyzed and illustrated with real data applications.