Pontifícia Universidade Católica
do Rio de Janeiro

Nathalie Sanghikian

Matheuristics for Multi-Product Maritime Inventory Routing Problems

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre (opção profissional) em Engenharia de Produção.

Advisor: Prof. Rafael Martinelli Pinto

Rio de Janeiro
September 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Nathalie Sanghikian

Matheuristics for Multi-Product Maritime Inventory Routing Problems

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre (opção profissional) em Engenharia de Produção. Approved by the Examination Committee.

Prof. Rafael Martinelli Pinto
Advisor
Departamento de Engenharia Industrial – PUC-Rio

Prof. Pedro Munari
Departamento de Engenharia de Produção – Universidade Federal de São Carlos

Prof. Marcus Vinicius Poggi
Departamento de Informática – PUC-Rio

Rio de Janeiro, September 14, 2020

Nathalie Sanghikian

Bachelor in Chemical Engineering (2011) at Universidade Estadual de Campinas (UNICAMP). The author works as process engineer since 2013 at Petrobras and had the opportunity through the Professional Master in Logistics to start her studies in this area.

Bibliographic data

Sanghikian, Nathalie

Matheuristics for Multi-Product Maritime Inventory Routing Problems / Nathalie Sanghikian; advisor: Rafael Martinelli Pinto. - 2020.
59 f.: il. color. ; 30 cm

Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2020.
Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Roteamento Marítimo com Estoques. 3. Matheurística. 4. Programação Linear. I. Martinelli Pinto, Rafael. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. III. Título.

CDD: 658.5

To my parents, José Rubens and Lucimar, for their support, encouragement and love. To my brother Rubens for always being concerned with my well-being and happiness. To my husband and my love, Renan, for our love, your patience and for making life happier.

Acknowledgments

First of all, I thank God for life, for health, for always protecting me on the paths I have taken so far and for the people He has placed through my journey who contributed to my growth.

To my parents, who always supported me, encouraged me, and took care of me with all love, dedication. Thanks also for the scolding, ear tugging. I am who I am, thanks to you. From me, you have the greatest love in the world. Forever and always.

To my family, for all the support and affection.

To my husband and love of my life, Renan, for understanding the countless hours I spent in front of the computer or talking about the thesis. For the coffees brought in the middle of the afternoon, for the washed dishes, for the "I love you" or "meu mozinho inteligente", always helping and encouraging me.

To my advisor Rafael Martinelli, who, from the first day, was the best advisor I could ever have. Without him, this work would not be possible. Thanks for the knowledge, support and help, whatever the time or day (including holidays and weekends), for all the meetings, for the patience, for always answering me even when I disturbed a lot.

To Petrobras for investing in the development of its professionals and for all learning opportunities throughout my career.

To the coordinator Patrícia and the manager Antônio Carlos for the opportunity to take this professional master's course.

To my dear friends, Maurício, Flávio and Cláudio, who have been present almost every day in recent years, listening to my stories, teaching me and advising me.

To my dear friend Gabriel, my eternal karaoke duo, for all the chats, laughs and also presence in good and bad moments.

To Luiz Gustavo, always helpful.

To PUC-Rio professors.

To all the others who followed my trajectory and always encouraged me. If I were to quote all the names, the thanks would be larger than the thesis. But you have my huge "thank you".

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Sanghikian, Nathalie; Martinelli Pinto, Rafael (Advisor). **Matheuristics for Multi-Product Maritime Inventory Routing Problems**. Rio de Janeiro, 2020. 59p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

In the current scenario of the world economy, it is essential to increase the integration between the different players in the companies' supply chain, reducing operational costs, and improving efficiency. Ship routing is a substantial part of this integration regarding global maritime commerce, being the object of study by many authors. In this work, we present different methodologies to solve variants of the Maritime Inventory Routing Problem. This problem involves a large number of variables and is a computationally complex problem to solve. Our primary motivation is to solve a ship routing real case of a large company in the Oil & Gas sector, achieving high-quality solutions in plausible processing times and improving companies current results. All developed methodologies are based on a metaheuristic combination with a linear mathematical model. One of the main differences between the methodologies lies in the mathematical model to solve the inventory problem, where we tested discrete-time and continuous-time approaches. Other differences concern the number of evaluated products (single or multi-product) and the metaheuristic used (local search heuristics with a Simulated Annealing probability factor or Hybrid Variable Neighborhood Search). For the methodology using the discrete-time model, the results are satisfactory, with low and punctual inventory violations in an acceptable computational time. For the methodology using the continuous-time model, the results are better once, in reduced computational time, inventory violations remain low or non-existent, depending on the scenario evaluated and the metaheuristic used. The results obtained in this work are remarkable and allow its practical application for real cases.

Keywords

Maritime Inventory Routing; Matheuristics; Linear Programming;

Resumo

Sanghikian, Nathalie; Martinelli Pinto, Rafael. **Matheurísticas para Problemas de Roteamento Marítimo com Estoques e Múltiplos Produtos**. Rio de Janeiro, 2020. 59p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

No cenário atual da economia mundial, é essencial aumentar a integração entre os diferentes atores da cadeia de suprimentos das empresas, reduzindo custos operacionais e melhorando a eficiência. O roteamento de navios é parte imprescindível dessa integração no comércio marítimo global, sendo objeto de estudo de muitos autores. Neste trabalho, apresentamos diferentes metodologias para resolver variantes do Problema de Roteamento Marítimo com Estoques. Esse problema envolve um grande número de variáveis e é computacionalmente complexo de ser resolvido. Nossa principal motivação é resolver um caso real de roteamento de navios de uma grande empresa do setor de Óleo & Gás, obtendo soluções de alta qualidade em tempos computacionais plausíveis e melhorando os resultados atuais da empresa. Todas as metodologias desenvolvidas são baseadas em uma combinação de uma meta-heurística com um modelo matemático de programação linear.Uma das principais diferenças entre as metodologias está no modelo matemático para resolver o problema de estoque, onde testamos abordagens de tempo discreto e tempo contínuo. As outras diferenças dizem respeito ao número de produtos avaliados (único ou múltiplos produtos) e à meta-heurística usada (heurística de busca local com um fator de probabilidade de *Simulated Annealing* ou *Hybrid Variable Neighborhood Search*). Para a metodologia que utiliza um modelo de tempo discreto, os resultados são satisfatórios, com violações baixas e pontuais do estoque em um tempo computacional aceitável. Para a metodologia que utiliza um modelo de tempo contínuo, os resultados são ainda melhores, uma vez que, em reduzido tempo computacional, as violações de estoque permanecem baixas ou inexistentes, dependendo do cenário avaliado e da meta-heurística utilizada. Os resultados obtidos neste trabalho são notáveis e permitem sua aplicação prática em casos reais.

Palavras-chave

Roteamento Marítimo com Estoques; Matheurística; Programação Linear;

Table of contents

List of figures

List of tables

1
Introduction

Reflecting the evolution of the world economy and commercial activity, international maritime trade lost strength in 2018: volumes grew 2.7% in 2018, down from 4.1% in 2017. The slowdown was wide-ranging and affected almost all sea cargo segments (UNCTAD 2019). In face of this situation, added to the intensification of market competition, it is essential to increase the integration between the different actors in the companies' supply chain, reducing operational costs and improving efficiency. In this scenario, the importance of an adequate routing of these ships is highlighted to guarantee the efficient use of the maritime fleet, meeting deadlines and competitive costs (Christiansen and Fagerholt 2009).

Even with the international maritime trade slowing down, between 2017 and 2018, oil and its derivatives represented 29.4% of the cargo loaded and 31.1% of the cargo unloaded in the world (UNCTAD 2019). In Brazil, according to the yearbook of the National Waterway Transport Agency (ANTAQ), in 2019 almost 65 thousand moorings were carried out and 1.104 billion tons were handled. Of this total, 224.7 million tons refer to the Oil and Derivatives sector, an increase of 11% over the previous year. Crude oil exports grew 37% in the 2018-2019 period and represent 61.2% of liquid bulk (ANTAQ 2019).

A large company in the gas oil sector is responsible for the production, refining and transportation of oil and oil products, such as LPG, naphtha, gasoline, kerosene, diesel, fuel oils. Thus, it is up to the company to contract ships and to design their routes, in order to meet the assumptions inherent to the process, such as, for example, guarantee of meeting demand, deadlines, inventory maintenance, aiming at the lowest possible cost. In this configuration, where the ship operator is also the inventory manager at the terminals, arises a problem known as Maritime Inventory Routing Problem (MIRP).

The MIRP is a combination of routing and scheduling problems of ships and inventory management (Christiansen and Fagerholt 2009). The routing must define which ports and in what sequence the ships will visit, respecting draft limitations and the capabilities of the ships. The time of each trip can vary from days to months. Ships do not have a mandatory port of origin. They

can start or end their journey anywhere. The product is produced and stored in cargo ports and is transported by sea to unloading ports. The goal is to find routes that minimize routing and inventory costs respecting the restrictions of the problem, such as upper and lower limits of inventory in ports, the capacities of ships, drafts and time horizon (Christiansen and Fagerholt 2009, Costa 2018). Furthermore, especially in this work, it is important to consider the multi-product variant, since the company refines and transports oil and oil products. Ports can be producers or consumers of one or more oil products and it is necessary to define which product will be loaded, transported and unloaded in ship's routing to meet a specifically demand. The MIRP is a variant of the Inventory Routing Problem (IRP). The main differences between the MIRP and the basic variant of the IRP are that in the IRP the vehicles usually start and end their route in a central depot and make their journey in a single day. In both problems the quantity loaded in each vehicle must respect the available capacity, and there may also be vehicles with different capacities.

The real case addressed in this work refers to a large Oil & Gas company, which, until recently, had its scheduling of ships' routes done empirically, based on the experience of the programmers. Seeking to improve the programming process, a ship routing system was developed by the company's information technology center in partnership with the logistics area. The system divides the model into two stages: inventory and routing. The first stage does not consider some relevant information, as ships initial inventory. The second one forces the programmer to fix the ships routes until they are empty. These issues cause, consequently, loss of relevant information, unfeasible scenarios and the need for the programmer to act on the results, making the system difficult to use and affecting the quality of the solutions. Despite these issues, this system is the current one used in the company. An alternative one-step model was developed by Costa (2018), considering all relevant variables to the system and eliminating the programmer's interference. However, using the branch and bound approach to solve an integer programming problem, it was found that the model resolution was not practical due to the high running times. Alternatively, the model has been extended to use *relax-and-fix* and *fix-and-optimize* heuristics to obtain good solutions in a shorter time. Despite presenting a better answer than the model, the search procedure was exhaustive and inefficient, taking a few hours to complete. Even this better solution is not applicable in the company, which requires faster and higher quality results (Costa 2018).

Metaheuristics is one of the main techniques for getting feasible solutions in which mathematical programming models are not able to find an optimal

solution. This approach does not guarantee optimal solutions, but possibly a feasible, high-quality solution. It is a method that provides a general framework and strategy guidelines for developing a specific heuristic method that is adjustable to a given problem. In the last decade, matheuristics or hybrid-metaheuristics are a growing field in operations research. They are optimization algorithms that result from the combination of metaheuristics and mathematical programming techniques (Papageorgiou et al. 2018). Hybrid-metaheuristics have been applied to several different routing problems (Archetti and Speranza 2014) and Papageorgiou et al. (2018) present an extensive computational study of hybrid-metaheuristics to find high-quality solutions for MIRP.

Thus, in this work, the objective is to develop a solution method for Multi-Product Maritime Inventory Routing Problem, which consists of a hybrid meta-heuristic, where a neighborhood-based heuristic is used for ship routing, assisted by a linear programming mathematical model which decides the inventory levels at ports, with the focus on minimum cost and acceptable computational time.

We developed different methodologies seeking high-quality solutions in a reduced computational time. This Master thesis is organized as follows: Chapter 2 presents a literature review of ship routing with inventory management and its solution methods. Then, Chapter 3 details the real case and a first comparison between the developed methodologies. Chapter 4 presents the solution approach using a discrete-time mathematical formulation. Next, Chapter 5 shows the solution approach using a continuous-time mathematical formulation and the two different metaheuristics developed, a Local Search with a Simulated Annealing probability factor and a Hybrid Variable Neighborhood Search. Chapter 6 presents the main results and its comparison. Finally, conclusions and future remarks are given in Chapter 7.

2
Literature Review

2.1
Inventory Routing Problems

Inventory Routing Problems (IRP) are challenging once they have to minimize costs by solving inventory and routing problems simultaneously, while avoiding stock-outs and respecting storage capacity limitations. The survey of Coelho et al. (2014) provides an introduction and an overview of Inventory Routing Problems. In a IRP basic version, each vehicle performs one route per time period, starting and ending at the supplier, and deliver products to a subset of customers. The quantity loaded in each vehicle must respect the available capacity, that may be different. The production and consumption rates are constant and deterministic (Bertazzi et al. 2008, Coelho et al. 2014).

There are a significantly number of IRP extensions and in some versions of the IRP, several products are handled at once (multi-product variant). Popović et al. (2012) develop a Variable Neighborhood Search (VNS) metaheuristic for solving a multi-product multi-period IRP in fuel delivery with multi-compartment homogeneous vehicles, and deterministic consumption that varies over each petrol station and each fuel type. Moin et al. (2011) propose a hybrid genetic algorithm to solve a multi-period, multi-supplier and multi-product IRP. Mjirda et al. (2014) solve a multi-product IRP using a two-phase Variable Neighborhood Search (VNS) metaheuristic. Coelho and Laporte (2015) propose a branch-and-cut algorithm for a multi-product, multi-period routing problem in which vehicles are compartmentalized and the costumers have several tanks. Cordeau et al. (2015) present a decomposition-based heuristic for the multi-product IRP.

2.2
Ship Routing

It is common to distinguish between three main operation modes in maritime transport: liner, tramp and industrial shipping. Liner shipping is similar to a bus service, where fixed schedules and itineraries must be followed. Container transport belongs to this mode. In the tramp shipping, the maritime transport

operator follows the availability of cargo in the market, often carrying a mix of mandatory and optional cargo in order to maximize profit. In the case of industrial shipping, the operator is the owner of the cargo and controls the fleet, trying to minimize the cost of cargo transportation. When it comes to oil transportation, almost all maritime transportation follows these two last modes of operation (Hemmati et al. 2014). Based on this distinction, the current work tackles a MIRP which falls under the industrial shipping mode of operation.

Christiansen and Fagerholt (2009) describe a basic MIRP and some of its extensions. For a basic case, a single product is transported. Storage capacities are known in all ports (loading and unloading), as the production and demand rate (assumed constant during the planning horizon). Neither the number of services, nor the amount loaded or unloaded in a port during the planning horizon is predetermined. The ports inventory levels should be respected. The main goal is to develop routes and schedules for a fleet of ships, minimizing transportation and inventory costs, and meeting the demand within a given planning horizon. Real-life problems are more complex, with other aspects, such as consumer or central supplier, inventory constraints for distinct subsets of ports, variable production or demand rates, several products, charters, and other. Christiansen and Fagerholt (2009) make a brief survey of authors who addressed these variations in their work.

Costa (2018) compares MIRP studies presented by several authors, according to their dimensions, i.e., number of ports, ships, products and planning horizon. Several of those authors work only on the single-product variant and with a small fleet, but as the problems dimensions increase, the greater is its complexity. Therefore, due to its real-case extensions and dimensions, the MIRP is a difficult and challenging problem regarding to the development of solution methods. Among many solution methods, this work adopts a hybrid-metaheuristic approach. Despite not being the MIRP, Hemmati et al. (2014) and Homsi et al. (2020) achieved remarkable results in proposing metaheuristics for the resolution of maritime transport problems, emphasizing the applicability of this type of approach to solving complex problems.

2.3 Methods for solving MIRP

Several authors have applied heuristics and hybrid methods to solve different MIRPs in the literature, in theoretical and real-life contexts. Dauzère-Pérès et al. (2007) designed a decision support system, using a memetic algorithm, also known as local genetic search or hybrid genetic algorithm,

to deal with a problem related to a calcium carbonate paste supplier. It combines a local search heuristic with crossover operators. Christiansen et al. (2011) used a construction heuristic incorporated in a genetic algorithmic structure to solve a MIRP with multiple products for the cement industry. The constructive heuristic is deterministic, but it has parameters that can be varied to produce different plans. The genetic algorithm is used to search for parameters that produce good plans from the constructive heuristics. The constructive algorithm starts with an initial plan, which usually implies violations of constraints. With each iteration, violations are identified and attempts are made to postpone or eliminate them. The procedure is expected to lead to a plan with no capacity violations. However, the final plan may also contain violations if at any point during construction it is impossible to find a solution that would improve the critical violation. The algorithm evaluates the results according to a number of criteria and chooses the one that maximizes a weighted sum of the criteria scores. The weights used in this sum are the parameters that the genetic algorithm seeks to optimize.

Siswanto et al. (2011) addressed a ship routing and scheduling problem with many technical and physical constraints and non-dedicated compartments. To solve this problem, the authors developed a Mixed Integer Linear Programming (MILP), followed by a greedy one-step heuristic and, based on this heuristic, proposed a set of heuristics for each sub-problem (route selection, ship selection, loading and unloading). Song and Furman (2013) also used a hybrid approach combining mathematical formulation with heuristics to solve sub-problems and improve a set of given solutions for a MIRP. Uggen et al. (2013) have developed a heuristic approach based on relax-and-fix and fix-and-optimize for a MIRP. Hemmati et al. (2016) considered a short-range inventory routing problem for several products, in which a heterogeneous fleet of ships transports various products from production sites to consumption sites in a continuous time frame. A two-phase iterative hybrid metaheuristic called Hybrid Cargo Generating and Routing has been proposed. In the first phase, the inventory routing problem is converted into a ship routing and scheduling problem, and solved by a mathematical formulation. In the second phase, an adaptive large neighborhood search is applied to improve the solutions. Diz (2017) developed relax-and-fix and fix-and-optimize heuristics to deal with a real problem related to the Brazilian offshore oil industry. The relax-and-fix phase aims to find feasible solutions while the fix-and-optimize seeks to improve the solution obtained. Papageorgiou et al. (2018) presented an extensive computational study, comparing variants of rolling horizon heuristics, K-opt heuristics, local branching, solution polishing and hybrid approaches to solve

a MIRP. Munguía et al. (2019) developed a hybrid approach using MILP formulations to solve sub-problems iteratively. Bertazzi et al. (2019) formulated a MIRP as a MILP model and designed a three-phase matheuristic to solve the problem. The first phase (*clustering phase*) implements the idea of grouping customers into a set of *clusters*. In the second phase (*routing construction*), a set of routes is built for the *clusters* generated in the first phase. In the third phase (*optimization*), a binary linear programming model is optimally solved to obtain a feasible solution.

Costa (2018) handled a realistic multi-product MIRP from a Brazilian oil and gas company. The objective was to develop a decision support tool to automate the company's ship scheduling process. The author applied a combination of relax-and-fix and fix-and-optimize heuristics to solve the problem, improving the company's current solutions, with an average execution time of three hours. Finally, we refer the reader to the work of Papageorgiou et al. (2014) for a complete overview of the MIRP literature.

2.4 Other Maritime Routing Studies

The company, in which this work is based on, looking for achieve new methods/approaches aiming to achieve high quality-solutions for its maritime routing problems. Some partnerships studies has been developed in the last years. Rodrigues et al. (2016) study a ship routing problem with pickup and delivery and time windows for maritime oil transportation presenting an optimization approach based on a mixed integer programming (MIP) model and an application of two tailor-made MIP heuristics, based on relax-and-fix and time decomposition procedures. Stanzani et al. (2018) address a real-life routing and scheduling problem with inventory constraints. Small sized instances are solved by a mathematical programming software and, given the difficulties of solving larger examples, they propose a multistart heuristic method that includes a metaheuristic GRASP and improvement procedures, and also a rolling horizon heuristic.

3
Problem Description

Next, we describe the problem and its main features. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote an undirected graph where $\mathcal{N}$ is the set of ports, and $\mathcal{E}$ the set of edges connecting them. To meet a given demand, the company has a heterogeneous fleet of ships, defined in the set $\mathcal{V}$, transporting different oil products, defined in the set $\mathcal{P}$, between ports within a given planning horizon H, which is the number of days evaluated. Each port $i \in \mathcal{N}$ has a product handling rate R_i (daily amount that can be loaded/unloaded in port i), and a known production or demand PD_{ip} for product $p \in \mathcal{P}_i$ (positive values represent production, while negative ones represent demands), where $\mathcal{P}_i \subseteq \mathcal{P}$ is the subset of products that are produced or demanded by port i. Moreover, inventory levels, at each port $i \in \mathcal{N}$ for each product $p \in \mathcal{P}_i$, must be within a given interval $[S_{ip}^{MN}, S_{ip}^{MX}]$ during the planning horizon. The time taken by a ship v to traverse an edge $(i, j) \in \mathcal{E}$ is given by T_{ijv}. Each ship $v \in \mathcal{V}$ has a capacity K_v and it is only able to visit ports in the subset $\mathcal{N}_v \subseteq \mathcal{N}$. Draft constraints are also considered in the problem, taking into account the cargo on the ships and the characteristics of the ports. To accomplish the draft constraints, the total loading on-board each ship v when visiting port i must be within a given interval $[L_{iv}^{MN}, L_{iv}^{MX}]$. Ports and ships may have initial inventory levels, indicated by parameters S_{0ip} and L_{0vp}, respectively.

As mentioned above, during a planning horizon each port has either a demand or a production for a specific product $p \in \mathcal{P}_i$. However, some demands can be met by different products, as long as they have the necessary quality. Thus, we define $\mathcal{P}_p$ as the subsets of products, allowed to meet a specific demand for product p. This transformation allows the model the possibility to choose which product to use to meet a demand with flexible quality, aiming the lowest cost. The transformation of one product into another is only allowed during a loading/unloading operation in a call of a ship in a port.

The objective is to define routes for the available ships, deciding which ports to visit, and the amount of products to be loaded and unloaded at each visit, minimizing routing and operational costs. Each port can be visited several times by the same ship during the planning horizon. Route's costs relate to the total distance that each ship must travel in a given solution, where C_{ij}^{T} is

the traveling distance associated to edge $(i, j) \in \mathcal{E}$. Operational costs relate to product handling and inventory holding costs at the ports. C^H_{iv} is the handling cost of loading or unloading one product unit by ship v at port i, while C^S_i is the inventory holding cost at port i. If the ships have to wait to load/unload, this cost is also take into account as C^W_v. Products delivered to ports might generate earnings that reduce operational costs. E^U_{ip} is the amount that the company earns for each product $p \in \mathcal{P}$ unit delivered at port $i \in \mathcal{N}$.

Comparing this work to Christiansen et al. (2011), there are some differences. In their work, products cannot be mixed (compartments on ships, different silos), production rate/demand varies over time. Besides, there are peak periods where the ports are ranked in a priority line to decide which demand meet. Also, the external locations are raw material supplier and they define an action (loading or unloading) for each visit in a port. On the other hand, there are some similarities as: a heterogeneous fleet, multi-product, upper and lower inventory limits, draft restrictions, different vessel capacity, variable load/unload, ships can start in port or at sea.

3.1 Real Case and Developed Methodologies

In this work, ports are split into national (located on the Brazilian coast) or international ones. National ports are producers and consumers, while international ports are exclusively consumers. The demand for international ports is not mandatory but generate earnings for the company. Production and demand rates are not necessarily constant, and may vary over the horizon. The operating and waiting costs for ships in ports are different and known. The fleet of ships is heterogeneous with restrictions on minimum and maximum draft, as well as capacity. Their initial positions are known, and may be at sea or in a port. Initial ship load is known and partial loading and unloading is allowed.

A critical aspect of real cases is that, in general, it is hard to find feasible solutions. In many cases, inventory violations occur and the company needs to find alternative ways (spot charters, for example) to dispose the excess production or meet incomplete demands, increasing its operating costs. Thus, this work aims high-quality solutions but we are aware of the infeasibilities that may occur.

It is challenging to consider all the above aspects in a single approach, providing high performance results, concerning costs and computational time. Yet, we tried to cover the highest number possible of them, developing different

methodologies during this work.

Table 3.1 presents the main differences between the three methodologies. Initially, we use a discrete-time linear programming mathematical model to optimize the inventory level (methodology 1). We adopt this approach to be able to compare with Costa (2018), which also used discrete-time linear programming mathematical model. In this first stage, the designed approach uses a Simulated Annealing metaheuristic responsible for ships' routing and scheduling. Despite the solutions quality, the computational time is not low enough for practical use and the model does not allow the inclusion of the multi-product feature straightforwardly. To overcome these issues, a continuous-time model is then developed without the need to discretize the time horizon, reducing the number of variables considerably. Lastly, two different metaheuristics, a local search heuristic with a Simulated Annealing probability factor and a Hybrid Variable Neighborhood Search, are evaluated in order to check the solution quality vs computational time (methodologies 2 and 3, respectively). The methodology 2 presents the single and multi-product variants, while the methodology 3 focuses on the multi-product variant.

	METHODOLOGY		
	1	**2**	**3**
Discrete Time	X		
Continuous Time		X	X
Single-Product	X	X	
Multi-Product		X	X
Production and Demand Rates	Variable	Constant	Constant
Simulated Annealing	X	X	
Hybrid VNS			X

Table 3.1: Differences between methodologies

As Table 3.1 shows, the production/demand variable rates are not considered in the continuous-time model, since its implementation is not simple. Despite this issue, the impact of constant production/demand rates assumption in the final results is small, once the variations in production/demand are small for the evaluated instances.

In the following chapters, detailed explanations of all stages will be presented.

4
Discrete-Time Approach

In the discrete-time approach, the planning horizon is discretized in a daily base and not allowing a vessel to stay for less than one day in each port. It is an initial approach, based on the Costa (2018) model, in an attempt to improve its results. However, this attempt results in a limited approach, which prevents the development of the multi-product variant, once the number of variables and, consequently, the computational time are already high. Also, the number of berths is not defined, allowing two or more ships to be in the same port at the same time. It does not reflect the reality, once the number of berths in the ports is limited.

In the methodology presented in this chapter, the routing and scheduling problems are solved by a Simulated Annealing metaheuristic (Kirkpatrick et al. 1983), while the inventory problem is solved by a discrete-time model. Both work together to optimize the routing and inventory costs.

The model is responsible for optimize inventory costs, respecting the problem restrictions as draft and inventory limits. It is the metaheuristic responsibility to search for neighbor solutions that improve the current solution through perturbation moves in ships routes and schedules. Each neighbor solution is defined by a single change of the current solution. In every perturbation move, the inventory cost for the neighbor solution is obtained from the mathematical model and provided to the metaheuristic. The metaheuristic then evaluates the neighbor solution total cost, compares this cost to the current solution total cost, keeping the best solution obtained so far.

4.1
Solution Representation and Neighborhoods

Firstly, the solution representation is presented. For each ship $v \in \mathcal{V}$, the solution keeps a list of pairs (i, s), $i \in \mathcal{N}_v$ and $s \in \mathbb{Z}$. This list represents the complete ship voyage during the planning horizon in order, and each element gives the port visited and its stay length in days.

To illustrate, Figure 4.1 shows an example of a solution following the proposed representation. Two ships and four ports are considered in the case

given, with the total length of stay in each port ranging between one and four days.

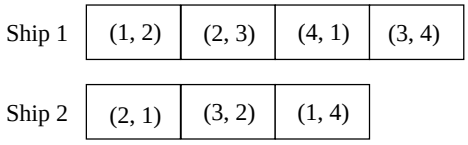

Figure 4.1: Solution representation (port, length of stay).

Note that Ship 1 visits the ports in the following order: 1, 2, 4, and 3 and stays in each port 2, 3, 1 and 4 days, respectively. This time is the total time (operational and waiting time) spent by the ship 1 in each port. Ship 2 visits port 2, 3 and 1, and stays in each port 1, 2 and 4 days, respectively.

The navigation time T_{ijv} between the ports i and j is also considered in the solution evaluation. It is calculated by taking the distance between the ports and dividing by the ship's speed. Additionally, considering fuel costs and the ship consumption, the routing cost C_{ij}^{T} is calculated. So, for each ship v its total routing cost and time can be obtained by summing all these elements.

The cost evaluation proceeds by providing the solution to the model. The linear programming mathematical model calculates the inventory (C_i^S), waiting (C_v^W) and handling (C_{vi}^H) costs and returns this value to the metaheuristic.

Some neighborhoods are responsible for searching for solutions that may improve the current solution through perturbation moves in ships routes and schedules. For moves in ships routes, i.e., a change in the order of the ports, considering an element as a pair (port, stay), the set of possible neighborhoods is:

- `Swap`: exchanges two elements in the solution;
- `Relocate`: removes an element in the solution, inserting in another position;
- `Insert`: inserts a new element in a route;
- `Delete`: deletes an element from a route.

The first two neighborhoods, *Swap* and *Relocate*, can be performed in its intra-route (same route) and inter-route (between two different routes) versions. For moves in the ship's schedules, the changes occur only in the *stay*. The possible neighborhoods are: exchange two *stays* in the same route, or increase / decrease a day of stay of a ship in a specific port, i.e. *stay* $+1$ or *stay* -1.

The Simulated Annealing metaheuristic is presented in Algorithm 1. It needs one parameter to run: the total number of iterations (η).

Algorithm 1: Simulated Annealing metaheuristic - Discrete-Time Approach (η)

```
1  L ← {1, .. , ℓ}
2  s ← Construct()
3  s* ← s
4  for η iterations do
5  |   k ← Random(L)
6  |   s' ← Perturb (s, k)
7  |   s' ← Formulation(s')
8  |   if Accept(s', s) then
9  |   |   s ← s'
10 |   |   if f(s) < f(s*) then
11 |   |   |   s* ← s
12 |   |   end
13 |   end
14 end
15 return s*
```

Initially, a set of neighborhoods $\mathcal{L}$ is defined (Line 1). They can be randomly selected, and they change either the position of the ports for each ship, or the length of stay. A constructive heuristic sets an initial solution and considers it as the best solution s^* (Lines 2-3). Its motivation is that starting from a solution in that the ships stay long enough to meet ports production/demand, the search for high-quality solutions will be faster. Initially, the number of necessary days to unload the entire inventory is calculated for all ports ($dmax_i$), as can be seen in Equation (4-1).

$$dmax_i = (S_i^{MX} - S_i^{MN})/R_i \qquad \forall i \in N \qquad (4\text{-}1)$$

Starting from the original position of all ships, assuming all stopped, the model is solved and total inventory violations are obtained for the entire time horizon for all ports. These violations are ordered from the largest to the smallest and associated with its respective ports and number of days ($dmax_i$). From this, ports are randomly inserted in the routes, respecting the time horizon and the inventory model. In order to keep the constructive heuristic simple, avoiding infeasibility, a port can only be inserted once on each route, but it can be on different initial routes.

The main loop is executed η times (Lines 4-14), and consists of a perturbation procedure that moves the solution s to a random neighbor solution s' in neighborhood k (Lines 5-6). This perturbation procedure selects randomly two routes from the solution s. Depending on whether the routes

are the same or not and their lengths, a subset of possible neighborhoods among all presented is available. The metaheuristic chooses randomly which neighborhood k will be evaluated. For each random neighbor solution s', its cost evaluation is carried out (Line 7). This evaluation considers routing, scheduling and inventory costs. The routing costs are calculated by the neighborhood itself, whereas for the calculation of inventory, handling and waiting costs, the linear programming model must be executed. As previously mentioned, it is necessary that the metaheuristic provide the information of which ship services which port on which day so that the model can be executed. The algorithm checks if the new solution s' will be accepted by a pre-established criterion (Lines 8-13).

This criterion is fulfilled if the cost of the new solution s' is better than the cost of s. Also, it is possible to accept a non-improving solution by a probability factor (Kirkpatrick et al. 1983). Once accepted, s is updated, and then it is verified if s is better than the best solution obtained so far s^* (Lines 10-12). The complete procedure is repeated until reaching the limit of the Simulated Annealing method. The method returns the final solution s^* in Line 15. It is worth mentioning that if, with the random movements made by the metaheuristic, the route exceeds the time horizon, the excess days are discarded, considering only the evaluated horizon. The final route, within the time horizon, has its total routing cost (C^T) and the model, for the last time, provides the inventory (C_i^S), waiting (C_v^W) and handling (C_{vi}^H) cost.

4.2 Mathematical Formulation

As described in the previous section, at each move of the metaheuristic, the mathematical model is executed to optimize inventory costs, respecting the restrictions established by the problem. This model follows the single-product and time-index variables approaches.

The routes of a set of $\mathcal{V}$ ships are evaluated. The ships can visit a set of $\mathcal{N}$ ports, in a $\mathcal{T}$ horizon of days. Each ship $v \in \mathcal{V}$ has capacity K_v and can load q_{vit}^L or unload q_{vit}^U an amount of a single-product on a port $i \in \mathcal{N}$ on day $t \in \mathcal{T}$. In the discrete-time approach, we do not define the number of berths, due to its additional complexity to the time-index model. It means that, two different ships can be in the same port i on the same day t. Also, the ship $v \in \mathcal{V}$ can deliver the product in port $i \in \mathcal{N}$ and profit from it (E_i^U).

The ports $i \in \mathcal{N}$ have a production/demand PD_i^t on day $t \in \mathcal{T}$ and a product handling rate R_i. As the metaheuristic does not difference handling and waiting time, since it would be complex to evaluate the neighborhoods

moves, the model considers that the handling costs C^H_{vi} are related to the necessary time to load/unload the established amount and the waiting cost C^W_v is automatically added as the cost of one day of waiting by ship v.

The initial inventory levels s^1_i on port $i \in \mathcal{N}$ and the initial load l^1_v on-board ship $v \in \mathcal{V}$ are given. For $t \in \mathcal{T}, t \neq 1$, the inventory level s^t_i on port $i \in \mathcal{N}$ and the total load l^t_v on-board ship $v \in \mathcal{V}$ are evaluated and C^S_i represents the inventory holding cost at port $i \in \mathcal{N}$. Maximum and minimum limits for the inventory level (S^{MX}_{it} and S^{MN}_{it}) and total load on-board (L^{MX}_{vi} and L^{MN}_{vi}) are defined.

In real applications, it is challenging to find solutions for the problem that meets the limits of cargo on-board ships, and draft and inventory limits at the ports. Also, it is important to have an indication of how far the solution is from feasibility, in order to guide the method towards feasible solutions. Based on this, we relax the draft and inventory limits constraints, penalizing each violation with a parameter μ. Thus, if the inventory levels on port $i \in \mathcal{N}$ on day $t \in \mathcal{T}$ (s^{slack}_{it}) or the total load on-board by ship $v \in \mathcal{V}$, on port $i \in \mathcal{N}$ on day $t \in \mathcal{T}$ (l^{slack}_{vit}) stay outside the limit ranges, the objective function is penalized by μ for each unity. The proposed method searches for solutions with minimum penalties, aiming to achieve feasible solutions if possible.

Once the inventory sub-problem is defined, it is important to explain how the metaheuristic and the model are connected. The constant y^t_{vi} is the link between the metaheuristic and the model. A procedure assesses whether a ship v will be in a particular port i on a given day t . If so, the value of y^t_{vi} is equal to 1, otherwise equal to 0. This information allows the model to know the routes of each ship defined by the metaheuristic and to evaluate the inventory C^S_i, handling C^H_{vi} and waiting C^W_v costs.

Next, the discrete-time model, based on the one described by Costa (2018), is presented with its objective function and constraints.

$$
\begin{aligned}
\min & \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} C^S_i s^t_i + && \text{(inventory costs)} \\
& \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} \left(C^W_v\, y^t_{vi} + C^H_{vi}(q^L_{vit} + q^U_{vit})\right) - && \text{(waiting and handling costs)} \\
& \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} E^U_i\, q^U_{vit} + && \text{(delivery earnings)} \\
& \mu\left(\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} s^{slack}_{it} + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} l^{slack}_{vit}\right) && \text{(infeasibility penalization)}
\end{aligned}
\tag{4-2}
$$

subject to

$$s_i^t = s_i^{t-1} + PD_i^t + \sum_{v \in V} \left(q_{vit}^U - q_{vit}^L \right) \qquad \forall i \in N, t \in T \backslash \{1\} \quad (4\text{-}3)$$

$$s_i^{t-1} + \sum_{v \in V} q_{vit}^U + PD_i^t \leq S_{it}^{MX} + s_{it}^{slack} + \sum_{v \in V} q_{vit}^L \qquad \forall i \in N, t \in T \backslash \{1\} \quad (4\text{-}4)$$

$$s_i^{t-1} + \sum_{v \in V} q_{vit}^U + PD_i^t \geq S_{it}^{MN} - s_{it}^{slack} + \sum_{v \in V} q_{vit}^L \qquad \forall i \in N, t \in T \backslash \{1\} \quad (4\text{-}5)$$

$$s_i^t \geq S_{it}^{MN} - s_{it}^{slack} \qquad \forall i \in N, t \in T \quad (4\text{-}6)$$

$$s_i^t \leq S_{it}^{MX} + s_{it}^{slack} \qquad \forall i \in N, t \in T \quad (4\text{-}7)$$

$$l_v^t - l_v^{t-1} = \sum_{i \in N} \left(q_{vit}^L - q_{vit}^U \right) y_{vi}^t \qquad \forall v \in V, t \in T \backslash \{1\} \quad (4\text{-}8)$$

$$l_v^t \leq K_v \qquad \forall v \in V, t \in T \quad (4\text{-}9)$$

$$l_v^t y_{vi}^t \leq L_{vi}^{MX} y_{vi}^t \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}10)$$

$$l_v^t y_{vi}^t \geq L_{vi}^{MN} y_{vi}^t - l_{vit}^{slack} \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}11)$$

$$q_{vit}^L \leq K_v y_{vi}^t \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}12)$$

$$q_{vit}^U \leq K_v y_{vi}^t \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}13)$$

$$q_{vit}^U + q_{vit}^L \leq R_i \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}14)$$

$$q_{vit}^L, q_{vit}^U, l_{vit}^{slack} \geq 0 \qquad \forall i \in N, v \in V, t \in T \quad (4\text{-}15)$$

$$s_{it}^{slack} \geq 0 \qquad \forall i \in N, t \in T \quad (4\text{-}16)$$

$$l_v^t \geq 0 \qquad \forall v \in V, t \in T \quad (4\text{-}17)$$

As mentioned before, the objective function (4-2) minimizes the operational costs, considering holding, waiting and inventory costs at the ports, unloading earnings and infeasibility penalties.

Constraints (4-3) guarantee the daily inventory flow in the ports, contemplating its production/demand such as loading and unloading operations. Constraints (4-4) and (4-5) force the inventory level to respect limits $[S_{it}^{MN}, S_{it}^{MX}]$, using the slack variable s_{it}^{slack} to relax the constraints, allowing violations. Constraints (4-6) and (4-7) force that, when the inventory is outside its lower and upper limits, the slack variable s_{it}^{slack} must represent the violated amount.

Constraints (4-8) keep the daily variation of the ship's load due to the loading and unloading operations performed. Constraints (4-9) express that the ship's capacity must be respected. Constraints (4-10) and (4-11) refer to the maximum and minimum drafts of each port, respectively. For the minimum draft restrictions, a relaxation is performed, incurring in a penalty if the drafts are not respected.

Constraints (4-12) and (4-13) indicate that the loads and unloads that each ship carries out in each port in a period must be less than the capacity of

each ship. Constraints (4-14) refers to the average maximum daily flow from the port, which the ship's loading and unloading must respect. In addition, Constraints (4-15)-(4-17) specify the variables' domains.

To exemplify, a single iteration from the developed hybrid-metaheuristic is presented. The current solution s, presented in Figure 4.2A, has a total cost of C^A. Randomly, the metaheuristic selects the ships 1 and 2 and, also randomly, the *Relocate* neighborhood is chosen. The perturbation consists of removing the element (4,1) from the ship's 1 route and inserting on ship's 2 route between elements (3,2) and (1,4). Then, the neighbor solution s' becomes as shown in Figure 4.2B. The metaheuristic calculates the routing cost and constants y^t_{vi} for ships 1 and 2 are updated. Constants y^t_{vi} represent if a ship v will be in a port i on a given day t. The model receives the changes through y and evaluates the inventory, handling and waiting cost. Those costs are provided to the metaheuristic, that calculates a total cost C^B. The metaheuristic decides that solution s' will be accepted. Once accepted, s is updated and it is verified if s is better then the best solution obtained so far s^*. If it is, the method keeps this solution.

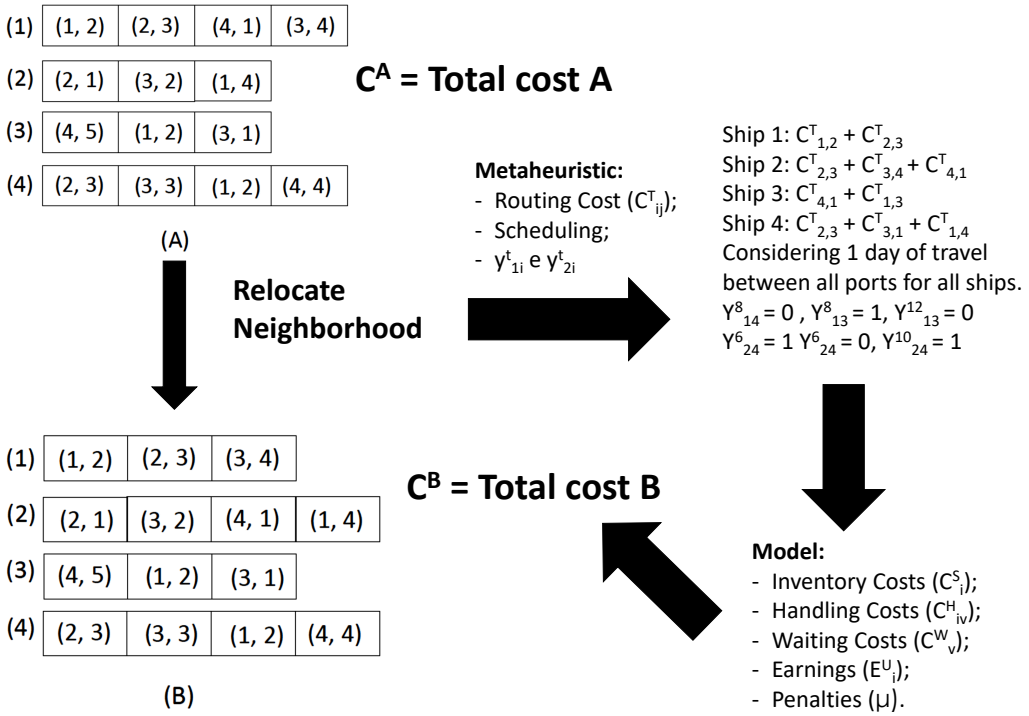

Figure 4.2: Example - Discrete-Time Approach

5
Continuous-Time Approaches

As mentioned in Chapter 4, the discrete-time approach does not contemplate some important MIRP features, as multi-product, product transformation, single berth availability and separated handling and waiting time, especially due to its high computational time. To contemplate all these features, aiming to obtain better results in a smaller computational time, a new mathematical formulation is introduced. It is based on the one presented by Christiansen and Fagerholt (2009), using continuous instead of discrete-time, allowing a vessel to stay for less than one day in each port. In this approach the time index on the variables disappears and consequently the number of variables is considerably reduced. Also, the model is responsible for inventory and scheduling sub-problems. On the other hand, the variable production/demand rate is no longer considered since its implementation in the continuous-time approach would not be simple, here assuming constant values for the planning horizon.

Besides the model, the metaheuristic also has to change and, in this approach, it is responsible only for the routing sub-problem. Also, due to the faster model resolution, it is possible to implement an additional local search procedure. Two different metaheuristics are developed for the continuous-time approach, aiming to achieve high-quality solutions. In the first metaheuristic, a local search with a Simulated Annealing probability factor (SA), a solution is constructed, followed by a local search procedure. After that, a perturbation is performed and the solution is accepted or not by a pre-established criterion. Finally, we run the local search. In the second metaheuristic, a Hybrid Variable Neighborhood Search (VNS), the first three steps are the same but after the perturbation a new local search takes place before the solution is accepted. Figure 5.1 presents the main differences between both approaches.

5.1
Solution Representation and Neighborhoods

The first change is on the metaheuristic. In the continuous-time version, the metaheuristic is responsible for the routing sub-problem, while the scheduling and inventory sub-problems are solved by the mathematical model.

Local Search with Simulated Annealing Probabilty Factor (SA)	Hybrid Variable Neighborhood Search (VNS)
1. Construct; 2. Local Search; 3. Perturb; 4. Accept or not; 5. Local Search.	1. Construct; 2. Local Search; 3. Perturb; 4. Local Search; 5. Accept or not.

Figure 5.1: Metaheuristics Differences

In this approach, the solution representation changes and it is represented by $(port, call)$ instead of $(port, stay)$. Thus, for each ship $v \in \mathcal{V}$, the solution keeps a list of pairs (i, m), $i \in \mathcal{N}_v$ and $m \in \mathbb{Z}$, that represents the complete ship voyage during the planning horizon in order, and each element gives the port visited and its call. Assuming a visit of a ship v in a port i. If it is the first time port i is visited by any ship $v \in \mathcal{V}$, its call is marked as 1 and is represented by the pair $(i, 1)$. If it is the second visit that occurs in port i, regardless of which ship is, the port's call is marked as 2 and represented by $(i, 2)$.

Each ship $v \in \mathcal{V}$ is only able to visit ports in the subset $\mathcal{N}_v \subseteq \mathcal{N}$. We define a set $m \in \mathcal{M}_i$ of possible calls to sequence the visits in the ports. Thus, a specific solution for the MIRP, can be defined as a permutation of pairs $(i, m) \in \mathcal{N}_v \times \mathcal{M}_i$, indicating the sequence of ports that each ship visits along with its respective call.

Figure 5.2 shows an example of a solution following the proposed representation, with each element on the solution represented as a pair indicating a port and its call. Two ships and four ports are considered in the given case, with up to two calls performed at each port. Note that Ship 1 and Ship 2 visit the ports in the following order: 1, 2, 4, 3 and 2, 3, 1, respectively.

Regarding Port 1, its first call is performed by Ship 2, followed by a visit from Ship 1. Note that when solving the inventory part of the problem, the visit $(1, 2)$ of Ship 1 will only be performed after the completion of Ship 2 route, due to visit $(1, 1)$. Moreover, this solution representation solves the berth problem presented for the discrete-time approach, in which the number of ships in a port could not be defined. As shown, Ship 1 has to wait for a berth in Port 1, assuming Port 1 has only a single-berth. For this work, it is adopted a single-berth per port, but if needed, it can be adjusted by replicating the port by a set of ports.

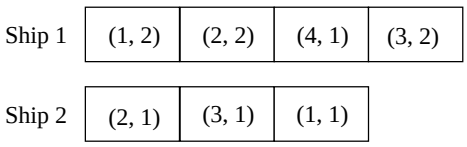

Figure 5.2: Solution representation (port, call).

As mentioned before, in the continuous-time approach, the neighborhoods are responsible for moves in ship's routes only, unlike the neighborhoods used in the discrete-time approach, detailed in Chapter 4, which are also responsible for scheduling moves. But despite the neighborhoods being not the same anymore, they are very similar and consider $(port, call)$ instead of $(port, stay)$.

For ship routing, i.e., a change in the order of the ports, we develop neighborhoods of *Swap*, *Relocate*, *Delete* and *Insert*. The first two neighborhoods are considered in their intra-route (on the same route) and inter-route (between two different routes) versions. This set of neighborhoods $\mathcal{L}'$ is used on both developed metaheuristics, a Local Search with a Simulated Annealing probability factor and a Hybrid Variable Neighborhood Search.

5.2 Local Search with a Simulated Annealing probability factor

The first metaheuristic is a local search heuristic with a Simulated Annealing probability factor, similar to the one presented in Algorithm 1. The main difference is that a local search procedure is added, aiming for higher-quality solutions. It is possible, in this approach, due to the smaller computational time spent in the continuous-time model execution. Also, the second difference lies in the constructive procedure, which is no longer the same. Algorithm 2 starts by building the set of neighborhoods $\mathcal{L}'$ to be used during the local search step (Line 1). Then, it calls a procedure that builds an initial solution s considering only each ship's initial position, calling the local search before returning the solution. Thus, initial routes for the ships are not defined, allowing the local search to build the solution iteratively while optimizing inventory levels (Line 2).

The next steps in the algorithm are the same as presented in section 4.1 (Lines 3 - 6), except the set of neighborhoods, whose change was previously explained. For the solution total cost evaluation (Line 7), the routing costs are calculated by the neighborhood itself. In this approach, the metaheuristic provides the routing information to the model through sets, but it is the model responsibility to obtain scheduling and inventory costs. Then, the algorithm

checks if the new solution s' will be accepted by a pre-established criterion, same as applied to the discrete-time approach (Lines 8-23). Once the solution is accepted, a Randomized Variable Neighborhood Descent (RVND) local search is applied on s' (Lines 10-19). The RVND procedure starts by shuffling the set of neighborhoods $\mathcal{L}'$ (Line 10), and running the local search following the defined random order of neighborhoods (Lines 11-19). For each move s'' of neighborhood ℓ, the mathematical formulation is executed (Line 13) and the algorithm checks whether the new solution s'' is better than s, updating s accordingly, moving to the next neighborhood if true. After the RVND, the algorithm verifies if s is better then the best solution obtained so far s^* (Lines 20-22). The complete procedure is repeated until reaching the maximum number of iterations η. The method returns the final solution s^* in Line 25.

Algorithm 2: Local search heuristic with a Simulated Annealing probability factor - Continuous-Time Approach (η)

```
1  L' ← {1, .. , ℓ}
2  s ← Construct()
3  s* ← s
4  for η iterations do
5      k ← Random(L')
6      s' ← Perturb (s, k)
7      s' ← Formulation(s')
8      if Accept(s', s) then
9          s ← s'
10         shuffle(L')
11         for ℓ ∈ L' do
12             for s'' ∈ N_ℓ(s') do
13                 s'' ← Formulation(s'')
14                 if f(s'') < f(s) then
15                     s ← s''
16                     break
17                 end
18             end
19         end
20         if f(s) < f(s*) then
21             s* ← s
22         end
23     end
24 end
25 return s*
```

5.3 Hybrid Variable Neighborhood Search

The second metaheuristic developed is a Hybrid Variable Neighborhood Search, and its described in Algorithm 3. Same as the other metaheuristics already presented, it needs one parameter to run: the total number of iterations (η).

Algorithm 3: Hybrid VNS (η)

```
1  L' ← {1, .. , ℓ}
2  s ← Construct()
3  s* ← s
4  for η iterations do
5  |  k ← Random(L')
6  |  s' ← Perturb(s, k)
7  |  s' ← Formulation(s')
8  |  shuffle(L')
9  |  for ℓ ∈ L' do
10 |  |  for s'' ∈ N_ℓ(s') do
11 |  |  |  s'' ← Formulation(s'')
12 |  |  |  if f(s'') < f(s') then
13 |  |  |  |  s' ← s''
14 |  |  |  |  break
15 |  |  |  end
16 |  |  end
17 |  end
18 |  if Accept(s', s) then
19 |  |  s ← s'
20 |  |  if f(s) < f(s*) then
21 |  |  |  s* ← s
22 |  |  end
23 |  end
24 end
25 return s*
```

The algorithm follows the same steps as Algorithm 2, where an initial solution s is constructed and set as the best solution s^* (Lines 1-3). Also, a perturbation move randomly selected in the set of neighborhoods $\mathcal{L}'$ is applied on the current solution s and s' cost is evaluated (Lines 5-7). Now, comes the difference: before the acceptance of the neighborhood solution, the Randomized Variable Neighborhood Descent (RVND) local search is applied on s' (Lines 9-17). Succinctly, the RVND procedure shuffles the set of neighborhoods $\mathcal{L}'$ (Line 8), and runs the local search following the defined random order of neighborhoods (Lines 9-17). For each move s'' of neighborhood ℓ, the mathematical formulation is executed (Line 11) and the algorithm checks

whether the new neighbor solution s'' is better than s', updating s' accordingly, moving to the next neighborhood if true. After the RVND, the algorithm decides if the new solution s' will be accepted by the same pre-established criterion used in the other algorithms (Lines 18-23). Once accepted, s is updated, and then it is verified if s is better than the best solution s^* obtained so far (Lines 20-22). The method returns the final solution s^* in Line 25.

5.4 Mathematical Formulation

The proposed mathematical formulation is based on the one described by Christiansen and Fagerholt (2009), but fixing routing variables as parameters and including some particular constraints related to the real problem. Based on a given routing solution, we define $\mathcal{M}_i$ as the set of calls on port i, and $\mathcal{M}_{iv}$ as the set of calls on port i performed by ship v. A parameter M_i^F indicates the last call on port i. While the original MIRP problem is defined on the undirected graph $\mathcal{G}$, as presented in Chapter 3, this subproblem is defined on an extended graph $\mathcal{G}'$, considering nodes $(i, m) \in \mathcal{N}_v \times \mathcal{M}_{iv}$, and arcs $(i, m, j, n) \in \mathcal{A}_v \subseteq (\mathcal{N}_v \times \mathcal{M}_{iv}) \times (\mathcal{N}_v \times \mathcal{M}_{iv})$, for each ship v. Moreover, each ship $v \in \mathcal{V}$ starts at port N_v^0 performing call M_v^0. Finally, let V_{im}^B be the ship performing the call just before call m on port i, i.e., $(m-1) \in \mathcal{M}_{iV_{im}^B}$, and V_i^F be the ship performing the last call of port i, i.e., $M_i^F \in \mathcal{M}_{iV_i^F}$.

Recalling Chapter 3, the ships $v \in \mathcal{V}$ transport different oil products $\mathcal{P}$. Each port $i \in \mathcal{N}$ has a known production or demand PD_{ip} for product $p \in \mathcal{P}_i$, where $\mathcal{P}_i \subseteq \mathcal{P}$ is the subset of products that are produced or demanded by port i. Some products $\mathcal{P}_p$ can meet other product's demand since they have the necessary quality. In this case, the model can choose which product to use to meet demand with flexible quality, aiming the lowest inventory violation and cost. In this work, this is called product transformation. This transformation is only allowed when a ship v operates in a call m in a port i. It means that once the ship is in the port, the model has the opportunity to check if any product p meets the quality of other the required product, but if the ship is en route, the transformation cannot be evaluated.

Note that the time index does not exist in this approach, but many parameters and variables have, now, the index p, relative to the multi-product variant. Tables 5.1 and 5.2 show the parameters and the variables of the model. All the parameters are the same, despite the index change. Referring to the variables, there are other changes besides the index, plus three new variables. Two of them refer to the scheduling sub-problem since, in this approach, the model is responsible for deciding the MIRP times. The t_{im} variables indicate

the time in which a call m at port i starts and must respect the planning horizon H. The tw_{im} variables express the waiting time in a call m at port i. These new variables allow to better evaluate the handling and waiting costs and they inform how much time the fleet wait to operate, another advantage of the continuous-time approach. The last ones $r_{imvpp'}$ refer to the product transformation, previously exposed.

Name	Description
PD_{ip}	Production/Demand on port i of each product p
R_i	Product handling rate on port i
S_{0ip}	Initial inventory level on port i of each product p
S_{ip}^{MX}	Maximum inventory level at port i of each product p
S_{ip}^{MN}	Minimum inventory level at port i of each product p
K_v	Ships capacity v
L_{0vp}	Initial total load on-board ship v
L_{iv}^{MX}	Maximum total load on-board each ship v when visiting port i
L_{iv}^{MN}	Minimum total load on-board each ship v when visiting port i
E_{ip}^{U}	Earnings for each product p unit delivered at port i.
C_v^W	Ship v waiting cost
C_i^S	Inventory holding cost at port i
C_{iv}^H	Handling cost of loading or unloading one product unit by ship v at port i
μ	Penalizing parameter

Table 5.1: Parameters - Continuous-Time Model - Multi-Product

Next, the objective function and constraints for the multi-product continuous approach are presented.

$$
\begin{aligned}
\min & \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}_i}\sum_{p\in\mathcal{P}_i} C_i^S s_{imp} + && \text{(inventory cost)} \\
& \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}_v}\sum_{m\in\mathcal{M}_{iv}}\sum_{p\in\mathcal{P}_i} \left(C_{iv}^H(q_{ivmp}^L + q_{ivmp}^U) + C_v^W tw_{im}\right) - && \text{(waiting and handling costs)} \\
& \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}_v}\sum_{m\in\mathcal{M}_{iv}}\sum_{p\in\mathcal{P}_i} E_{ip}^U q_{ivmp}^U + && \text{(delivery earnings)} \qquad (5\text{-}1) \\
& \mu\left(\sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}_i}\sum_{p\in\mathcal{P}_i} s_{imp}^{slack} + \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}_v}\sum_{m\in\mathcal{M}_{iv}} l_{ivm}^{slack}\right) && \text{(infeasibility penalization)}
\end{aligned}
$$

subject to

$$
\begin{aligned}
l_{jvnp} = l_{ivmp} + q_{jvnp}^L - q_{jvnp}^U - & \\
\sum_{p'\in\mathcal{P}_p} r_{jvnpp'} + \sum_{p'\in\mathcal{P}_p} r_{jvnp'p} & \quad \forall v\in\mathcal{V}, (i,m,j,n)\in\mathcal{A}_v, p\in\mathcal{P} \qquad (5\text{-}2) \\
L_{iv}^{MN} - l_{ivm}^{slack} \le \sum_{p\in\mathcal{P}} l_{ivmp} \le L_{iv}^{MX} + l_{ivm}^{slack} & \quad \forall v\in\mathcal{V}, i\in\mathcal{N}_v, m\in\mathcal{M}_{iv} \qquad (5\text{-}3)
\end{aligned}
$$

Name	Description
t_{im}	Time in which a call $m \in \mathcal{M}_i$ at port $i \in \mathcal{N}$ starts
tw_{im}	Waiting time in call $m \in \mathcal{M}_i$ at port $i \in \mathcal{N}$
q^L_{ivmp}	Amount loaded by ship $v \in \mathcal{V}$, on port $i \in \mathcal{N}_v$ in call $m \in \mathcal{M}_{iv}$ of each product $p \in \mathcal{P}_i$
q^U_{ivmp}	Amount unloaded by ship $v \in \mathcal{V}$, on port $i \in \mathcal{N}_v$ in call $m \in \mathcal{M}_{iv}$ of each product $p \in \mathcal{P}_i$
s_{imp}	Inventory level on port $i \in \mathcal{N}$ in call $m \in \mathcal{M}_i$ of each product $p \in \mathcal{P}_i$
l_{ivmp}	Total load on-board ship $v \in \mathcal{V}$, on port $i \in \mathcal{N}_v$ in call $m \in \mathcal{M}_{iv}$ of each product $p \in \mathcal{P}$
s^{slack}_{imp}	Inventory levels outside the limit ranges on port $i \in \mathcal{N}$ in call $m \in \mathcal{M}_i$ of each product $p \in \mathcal{P}_i$
l^{slack}_{ivm}	Total load on-board outside the limit ranges by ship $v \in \mathcal{V}$, on port $i \in \mathcal{N}_v$ in call $m \in \mathcal{M}_{iv}$
$r_{imvpp'}$	Amount of product $p' \in \mathcal{P}_p$ replacing the demand of product $p \in \mathcal{P}_i$ during a loading operation in call $m \in \mathcal{M}_{iv}$ of ship $v \in \mathcal{V}$ at port $i \in \mathcal{N}_v$

Table 5.2: Variables - Continuous-Time Model - Multi-Product

$$L^{MN}_{jv} - l^{slack}_{jvn} \leq \sum_{p \in \mathcal{P}} l_{ivmp} \leq L^{MX}_{jv} + l^{slack}_{jvn} \qquad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v \qquad (5\text{-}4)$$

$$q^L_{ivmp} \leq l_{ivmp} \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}, p \in \mathcal{P}_i \qquad (5\text{-}5)$$

$$\sum_{p \in \mathcal{P}} l_{ivmp} \leq K_v - \sum_{p \in \mathcal{P}_i} q^U_{ivmp} \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} \qquad (5\text{-}6)$$

$$t_{jn} = t_{im} + tw_{im} + T_{ijv} + \sum_{p \in \mathcal{P}_i} \frac{\left(q^L_{ivmp} + q^U_{ivmp}\right)}{R_i} \qquad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v \qquad (5\text{-}7)$$

$$t_{im} \geq t_{i(m-1)} + \sum_{p \in \mathcal{P}_i} \frac{\left(q^L_{iV^B_{im}(m-1)p} + q^U_{iV^B_{im}(m-1)p}\right)}{R_i} \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \setminus \{1\} \qquad (5\text{-}8)$$

$$s_{i1p} = S_{0ip} + (PD_{ip} t_{i1}) \qquad \forall i \in \mathcal{N}, p \in \mathcal{P}_i \qquad (5\text{-}9)$$

$$s_{imp} = s_{i(m-1)p} + PD_{ip}(t_{im} - t_{i(m-1)}) + \left(q^U_{iV^B_{im}(m-1)p} - q^L_{iV^B_{im}(m-1)p}\right) \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \setminus \{1\}, p \in \mathcal{P}_i \qquad (5\text{-}10)$$

$$S^{MN}_{ip} - s^{slack}_{imp} \leq s_{imp} \leq S^{MX}_{ip} + s^{slack}_{imp} \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i, p \in \mathcal{P}_i \qquad (5\text{-}11)$$

$$s_{iM^F_i p} - (q^L_{iV^F_i M^F_i p} - q^U_{iV^F_i M^F_i p}) + PD_{ip}(H - t_{iM^F_i}) \leq S^{MX}_{ip} + s^{slack}_{iM^F_i p} \qquad \forall i \in \mathcal{N}, M^F_i \neq 0, p \in \mathcal{P}_i \qquad (5\text{-}12)$$

$$s_{iM^F_i p} - (q^L_{iV^F_i M^F_i p} - q^U_{iV^F_i M^F_i p}) + PD_{ip}(H - t_{iM^F_i}) \geq S^{MN}_{ip} - s^{slack}_{iM^F_i p} \qquad \forall i \in \mathcal{N}, M^F_i \neq 0, p \in \mathcal{P}_i \qquad (5\text{-}13)$$

$$t_{im} \leq H \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \qquad (5\text{-}14)$$

$$t_{\mathcal{N}^0_v v M^0_v} = t_{0v} \qquad \forall v \in \mathcal{V} \qquad (5\text{-}15)$$

$$l_{\mathcal{N}_v^0 v M_v^0 p} = L_{0vp} \qquad \forall v \in \mathcal{V}, p \in \mathcal{P} \tag{5-16}$$

$$q_{ivmp}^L, q_{ivmp}^U, l_{ivmp} \geq 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}, p \in \mathcal{P}_i \tag{5-17}$$

$$l_{ivm}^{slack} \geq 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} \tag{5-18}$$

$$s_{imp}, s_{imp}^{slack} \geq 0 \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i, p \in \mathcal{P}_i \tag{5-19}$$

$$t_{im} \geq 0 \qquad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \tag{5-20}$$

$$r_{imvpp'} \geq 0 \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}, m \in \mathcal{M}_i, p \in \mathcal{P}_i, p' \in \mathcal{P}_p \tag{5-21}$$

As before, the objective function (5-1) minimizes the operational costs, considering holding and inventory costs at the ports, unloading earnings and infeasibility penalties. Constraints (5-2) compute the cargo on-board the ship at each port call. Constraints (5-3) and (5-4) limit the total cargo on-board the ship to be within interval $[L_{iv}^{MN}, L_{iv}^{MX}]$, with the slack variable l_{ivm}^{slack} used to relax the constraints, allowing violations in the limits. Constraints (5-5) - (5-6) limit the cargo on-board the ship to respect the ship capacity and the amount to load and unload. Constraints (5-7) and (5-8) compute the starting time of a port call. Constraints (5-9) and (5-10) compute the inventory level at the ports at each call. Constraints (5-11)-(5-13) force the inventory level to respect limits $[S_{ip}^{MN}, S_{ip}^{MX}]$, but allowing violations using slack variables s_{imp}^{slack} to relax the constraints. Constraints (5-14) limit the time to start visits to be within the planning horizon. Constraints (5-15) - (5-16) set the time from the ship first call and load on each ship, respectively. Constraints (5-17)-(5-21) specify the variables' domains.

As it was done for the discrete-time approach, a single iteration from the developed hybrid-metaheuristic for the continuous-time approach is presented. Figure 5.3A shows the current solution s which has a total cost of C^A. Randomly, the metaheuristic selects the Ships 1 and 2, and the *Relocate* neighborhood. The perturbation consists of removing the element (2, 2) from the Ship 1's route and inserting on Ship 2's route between elements (3,1) and (1,1). Figure 5.3B shows the new solution s'.

The metaheuristic calculates the routing cost. The model receives the changes through the sets $\mathcal{A}_v$, $\mathcal{M}_i$, $\mathcal{M}_{iv}$, $\mathcal{N}_v$ and evaluates the scheduling, inventory, handling and waiting cost. Those costs are provided to the metaheuristic, that calculates a total cost C^B. If the selected metaheuristic is the local search with a Simulated Annealing, it decides if the solution s' will be accepted s and applies a Randomized Variable Neighborhood Descent (RVND) local search. If

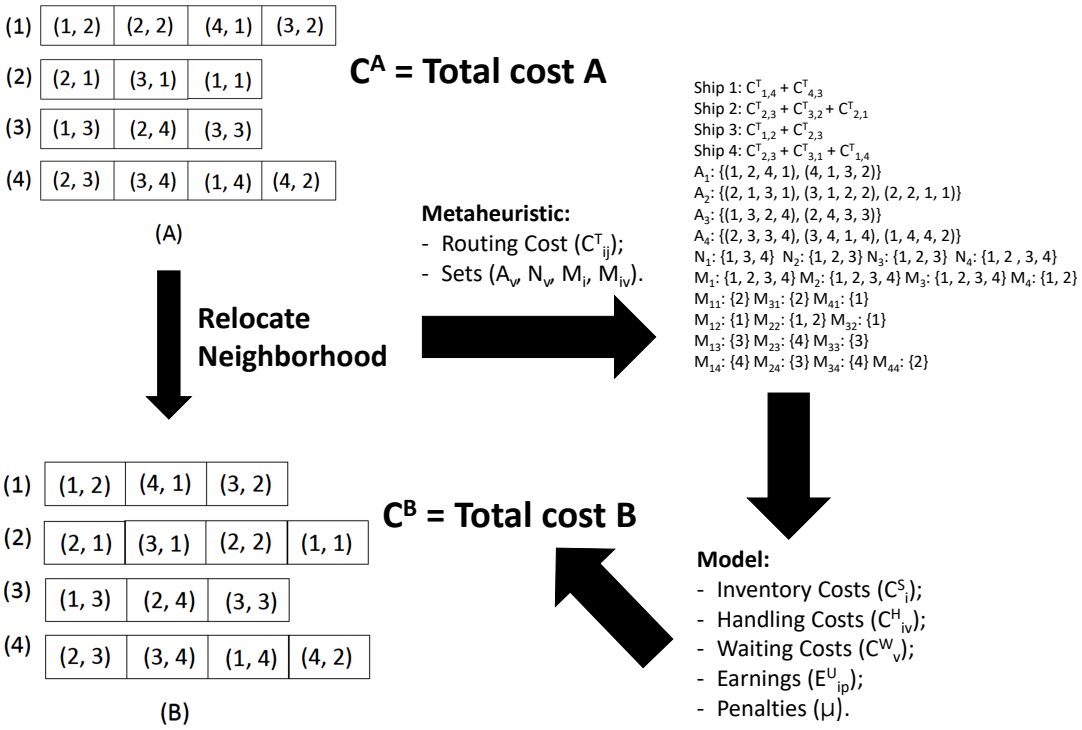

Figure 5.3: Example - Continuous-Time Approach

the selected metaheuristic is the Hybrid Variable Neighborhood search, before the acceptance of the neighborhood solution s', the RVND local search is applied. Lastly, it is verified if solution s is better than the best solution obtained so far s^*.

6
Results

In this chapter, the results of the different methodologies are shown and discussed. First, the discrete-time results are presented, followed by the results from the two developed metaheuristics for the continuous-time approach. For the local search with a Simulated Annealing probability factor both single and multi-product variants are evaluated. Also, a final comparison takes place, not only between the developed methodologies but adding a comparison with the best known solutions from the literature.

6.1
Data and Execution Environment

For all methodologies, the data are the same used by Costa (2018), contemplating ten real different two-month instances, from August to May. For each instance, an approximately 60 days horizon is considered. To make it easier, we can refer to the instances according to their month numbers, i.e., August-September as 08-09. Each instance is composed of 18 ports (15 national and three international ports) in which only a berth is available, a fleet size ranging between 12 and 15 ships and nine different oil products. Information about ports, products, ships, locations, distances, ship's initial load and position, ship's speed and consumption, fuel cost, daily production/demand, initial inventory levels, inventory and draft limits and profits by delivering product are provided. For confidentiality reasons, this information cannot be published. Also, the results consider a penalization parameter, which is defined by the author. The presented results do not represent the company's performance. All the costs results presented in this chapter are divided by a factor.

The metaheuristics and the mathematical formulations were coded using Julia language v1.0.5. For discrete-time approach, the experiments were performed on a computer with an Intel (R) Core (TM) i5-6300U processor with 8.00 GB of RAM and the model was solved by CBC solver. We tested each instance two times, limiting the metaheuristic to 1000 iterations ($\eta = 1000$). For continuous-time approaches, the experiments were performed on a computer with an Intel i7-3960 CPU at 3.3GHz and 64 GB of RAM running Linux. The

model was solved by CPLEX 12.8 solver, running in a single thread. We tested each instance ten times, limiting the metaheuristic to 250 iterations ($\eta = 250$).

6.2 Discrete-Time Approach

Due to the considerable computational time, this section results are the best obtained of two runs. Also, since this approach is a single product variant, we sum up the production/demand and the inventory for all products, considering them as one single product.

Table 6.1 shows the total inventory violations in the 60-day horizon for all evaluated instances in all ports. They are low and non-repetitive between ports (72 violations). For a better analysis of the results, Figure 6.1 presents the results of average, maximum and minimum inventory levels for the 10 different instances. The top line of graphs refers to the maximum level reached, the middle line corresponds to the average level, and, finally, the bottom line to the minimum inventory level. The levels represent the inventory amount divided per inventory capacity. Regarding inventory violations, it is possible to note that only for some cases, there are minimum levels below zero or maximum levels exceeding 100%. For most cases, the maximum occupancy is around 80% to 100%, the average in the range of 40% to 60% and the minimum 0%, with the exception of some instances such as 08-09, 10-11, 02-03, reaching 20%. Thus, the result of the model provides satisfactory inventory levels. An interesting point to be noted is the violation of the 11-12 period is that it occurs right at the beginning of the horizon. Looking at the previous period (10-11), it is possible to observe that this violation does not occur. This indicates that with good planning, carried out in advance, it is possible to reduce violations. It is also important to note that, in practice, it is highly unlikely to obtain a planning that does not violate inventory, leading companies to look for expensive solutions to deal with these violations.

Instance	8-9	9-10	9-10	10-11	11-12	11-12	11-12	12-01
Port	4	6	15	16	4	5	8	3
Violation	8	2	4	14	1	35	5	3

Table 6.1: Inventory Violation - Discrete-Time Model

Although these results are positive, with low and non-repetitive inventory violations, a point of attention and improvement lies in the computational time since, for each instance, a run takes approximately 40 minutes. Also, there are still aspects that are not covered due to the discrete-time model computational time, as mentioned in Chapter 4. They are: single berth, multi-

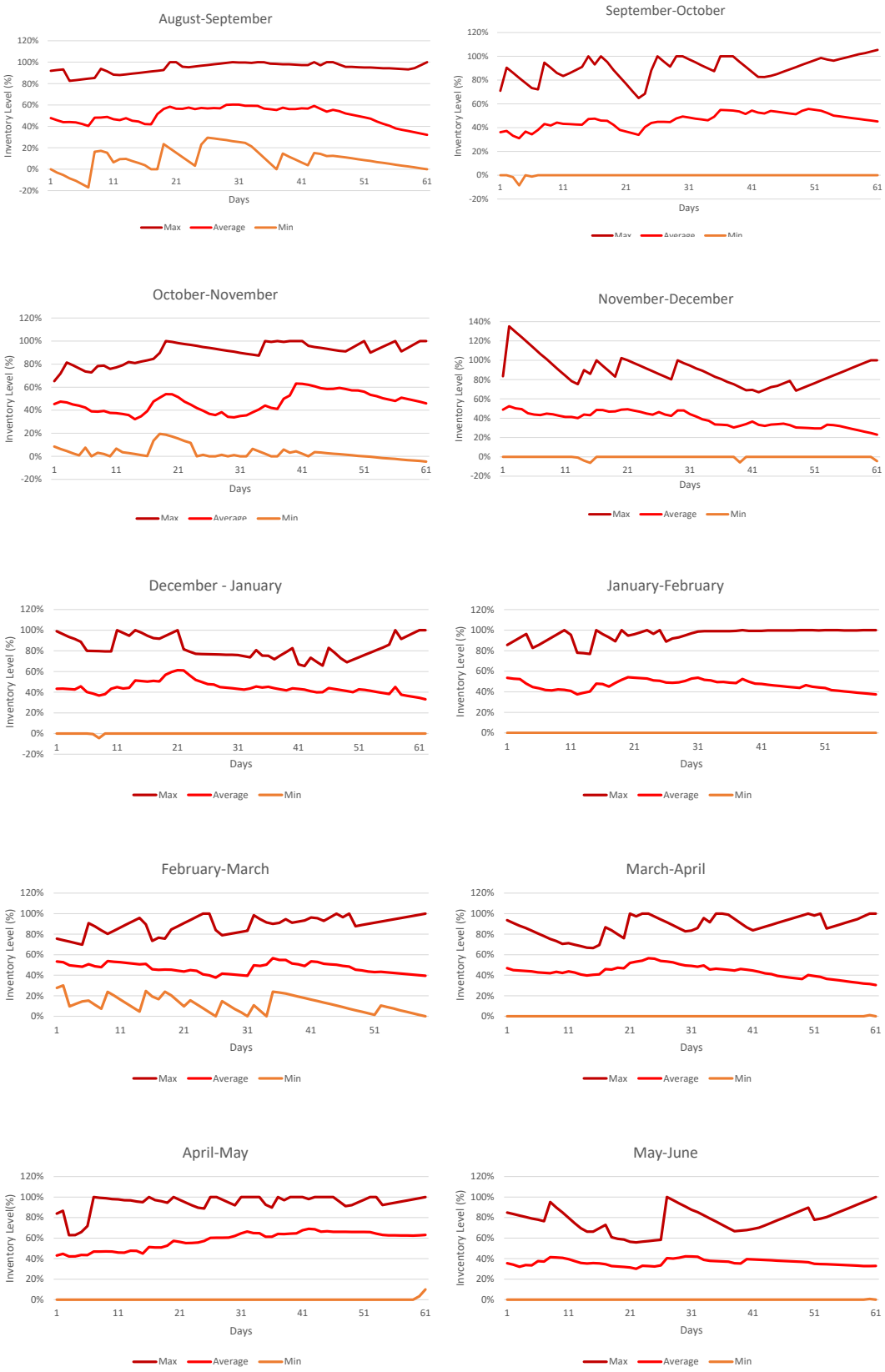

Figure 6.1: Inventory Level- Discrete-Time

product, product transformation, handling and waiting time. So, aiming for a formulation that contains these features, the continuous-time approaches are implemented. Their results are shown in the next section.

6.3 Continuous-Time Approaches

This section presents the results of the continuous-time model for both developed metaheuristics. Its is important to note that the continuous-time approach covers the proposed features (single berth, multi-product, product transformation and handling and waiting time) that the discrete-time was not able due to its computational time, except the variable production/demand rate assumed constant in the continuous-time approach.

Therefore, due to their differences, the continuous-time results are not directly comparable with discrete-time results. Even the single-product variant cannot be compared since now only a berth per port is available instead of several ships being able to stop at the same port at the same time. For the multi-product, the differences are larger because the production/demand rates are specific for a product, impacting ship's routing, scheduling and inventory levels in the ports. Although, regarding computational time and violations, we see remarkable improvements as will be presented next.

6.3.1 Single Product

The first continuous-time approach presented is the simplest, with a single product only. To consider that, we sum up the production/demand and the inventory for all products, considering them as one, as we have done to the discrete-time model. However, all of the other features were contemplated. The metaheuristic used was the local search with a Simulated Annealing probability factor (SA). Furthermore, the presented values are averages obtained in the ten times each instance was run, or their maximum and minimum values.

In Figure 6.2, average costs for all the instances are presented. They are divided in routing cost, inventory cost (including here the handling and waiting cost), total cost and total cost without penalties (total cost deducted by the violations penalties). The first aspect is that the inventory cost has a large influence within the total cost, due to the penalization parameter. For two instances (08-09 and 10-11) that the inventory costs increase, due to violations, the total cost increases too. Still about the inventory cost, for some instances it turns to be negative, representing the occurrence of an earning in these months with the product delivery abroad. In the case of the routing cost, it remains

very constant for all instances, and in the months that the inventory violations do not occur, it is the main portion of the total cost. Finally, as expected, the total cost without the inventory violation penalties tracks the routing cost.

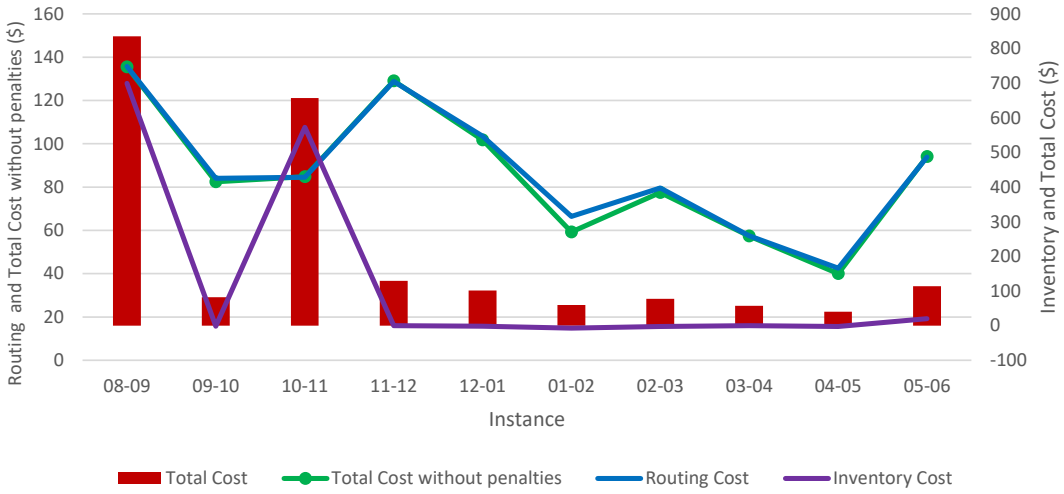

Figure 6.2: Costs - Continuous-Time - Single Product

Figure 6.3 shows the maximum, average and minimum values for violations and total cost for the ten evaluated instances. It is possible to notice that the violations are responsible for increasing the total cost. Also, analyzing two instances (08-09 and 10-11), a considerable variation between maximum and average violation values can be observed. Although the standard deviation is not shown, it can be concluded that when the maximum value is much higher than the average value, the results deviation are high due to the seeds. Taking instance 08-09 as an example: the results for nine seeds violates 1 unit the inventory, but the seed number 10 has a high violation (37). So, this only seed raises the average value and the deviation. The same analysis can be extended to 10-11 instance.

The next step is to evaluate the port's inventory level, but here it is worth making an observation. The amount of results and information obtained is very significant. The planner can evaluate the situation port by port, product by product. However, it is complex to present all the information in text format, so just the instances average values will be presented. But, again, in spreadsheets, all the results can be detailed, being really useful for the planner.

In Figure 6.4 the inventory level per call is presented, which is the inventory amount in a call divided by its capacity, for the ten evaluated instances. Considering one instance, the maximum represents the maximum inventory level, obtained in ten runs. It is the average for all products and ports in the specific call. Same for the average and minimum inventory levels.

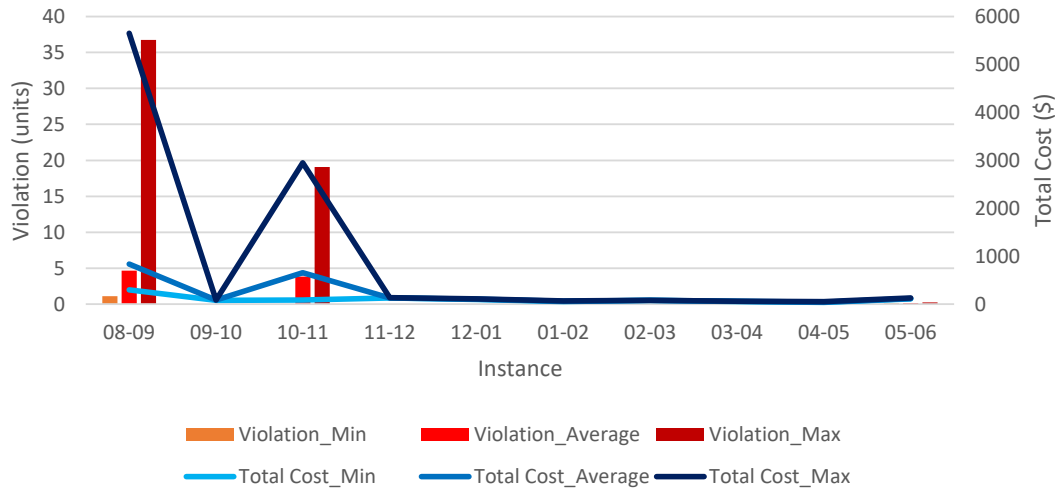

Figure 6.3: Violation - Continuous-Time - Single Product

Notice that for all instances the inventory level does not violate its limits, neither the upper nor the lower limit, in any call.

In Table 6.2 there is some general information for each instance. All the values are averages from the ten times each instance was tested. The total average violation, as we already mentioned, is small (9 units) and, considering the average of all calls, the ship waiting time (tw_{im}) is three and a half days, what is economically favorable once the ships do not waste much time waiting to operate in the ports and may visit other ports instead, counting on the fact that a single berth is always available for the evaluated fleet. The average computational time is 10 minutes.

	Waiting Time (days)	Violation(units)	Computational Time (s)
08-09	4	5	300
09-10	3.3	0	560
10-11	4.6	4	652
11-12	3.6	0	538
12-01	3.3	0	785
01-02	4.0	0	715
02-03	5.7	0	680
03-04	4.0	0	437
04-05	-	0	595
05-06	2.5	0	683

Table 6.2: General Information - Continuous-Time - Single Product

As mentioned before, we cannot compare these results directly with the discrete-time results because the continuous-time approach considers more features, as the availability of only a single berth, the different approach to address waiting time costs. However, it is quite clear that the continuous-time results stand out with a shorter computational time (four times faster)

Figure 6.4: Inventory Level- Continuous-Time - Single Product

and fewer violations (eight times smaller). For the discrete-time approach, the computational time was about 40 minutes and violations 72 units, while for the continuous-time approach, in just ten minutes, the reached violation is only nine units.

6.3.2 Multi-Product - Simulated Annealing

For the results of this section, the MIRP is solved by a hybrid-metaheuristic, using the local search with a Simulated Annealing probability factor procedure. Also, the multi-product feature is added. Nine products are evaluated and some of them can be transformed, i.e., service other products' demands if necessary. In Figure 6.5, we see the average costs obtained for each instance, same as explained for Figure 6.2. As we already know, the inventory costs have huge impact on the total cost. The routing costs are much smaller and the total costs without penalties follow them. For some instances, the total cost without penalties can be smaller than the routing cost. It means that the inventory costs is negative, i.e, the instance earns money delivering products abroad. Also, we can see that the multi-product costs are larger than the single-product costs, once the problem complexity increases considerably and it is more difficult to satisfy the restrictions to avoid violations. Once again, the instance 08-09 presents the larger costs.

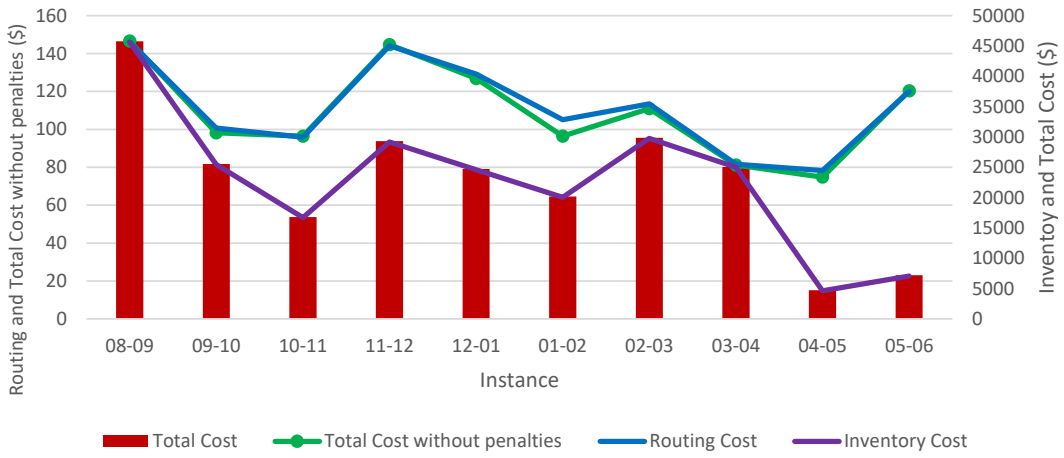

Figure 6.5: Averages Costs - Multi Product - SA

Figure 6.6 presents the maximum, average and minimum values for violations and total cost for the ten evaluated instances, similar as Figure 6.3. It is possible to see that, now, every instance has a violation, even if for some months, this violation is quite small (04-05 and 05-06). Also, the results present some variance between minimum and maximum values for each

instance due to the ten different seeds used, and may be larger or smaller depending on each instance. A seed is a number (or vector) used to initialize a pseudo-random number generator. If a pseudo-random number generator is reinitialized with the same seed, it will produce the same sequence of numbers. It ensures that results are reproducible. Once again, the total cost follows the same trend as the violation. The violations for the multi-product variant are higher from those of the single product variant since it is more complex to satisfy the constraints.

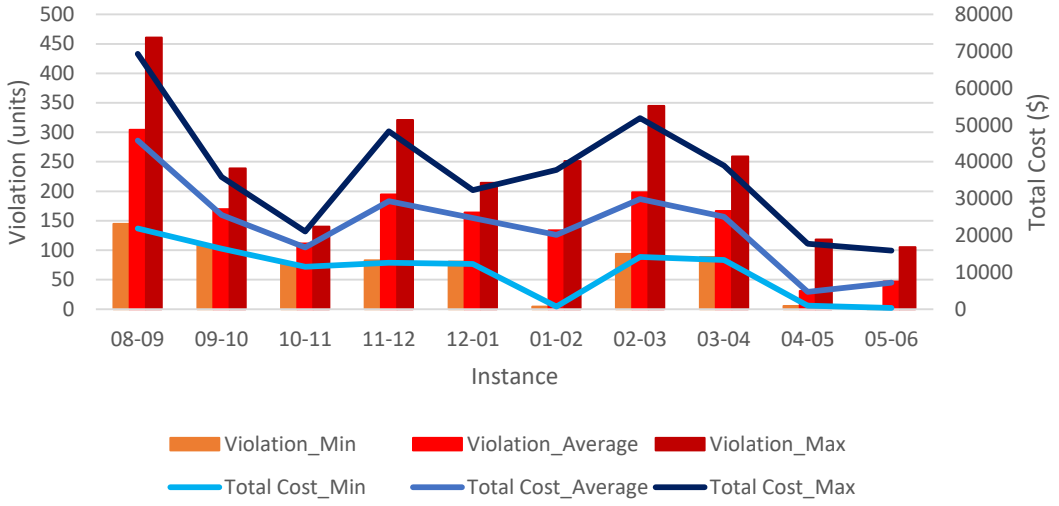

Figure 6.6: Violation vs Cost - Multi Product - SA

The inventory level is shown in Figure 6.7 and most results of average of maximum and minimum inventory levels for all products/ports stay inside limits, except instances 08-09 and 02-03. As has been commented before, these values are averages. The planner can see exactly which port, in which call has violations. Furthermore, the average level stays in range 40% - 60% for most instances.

In Table 6.3, some general information is presented for the SA multi-product approach. The ship waiting time, considering the average of all calls, is almost six days, still being a good result. The average violations are larger than those obtained in the single-product approach. Also, an average value of 16 units is transformed, i.e., 16 units of some products meet other products demands. The average computational time is 3 minutes.

6.3.3 Multi-Product - Variable Neighborhood Search

The Hybrid Variable Neighborhood Search (VNS) is similar to the previous one (SA), except for the time the solution is accepted, as shown

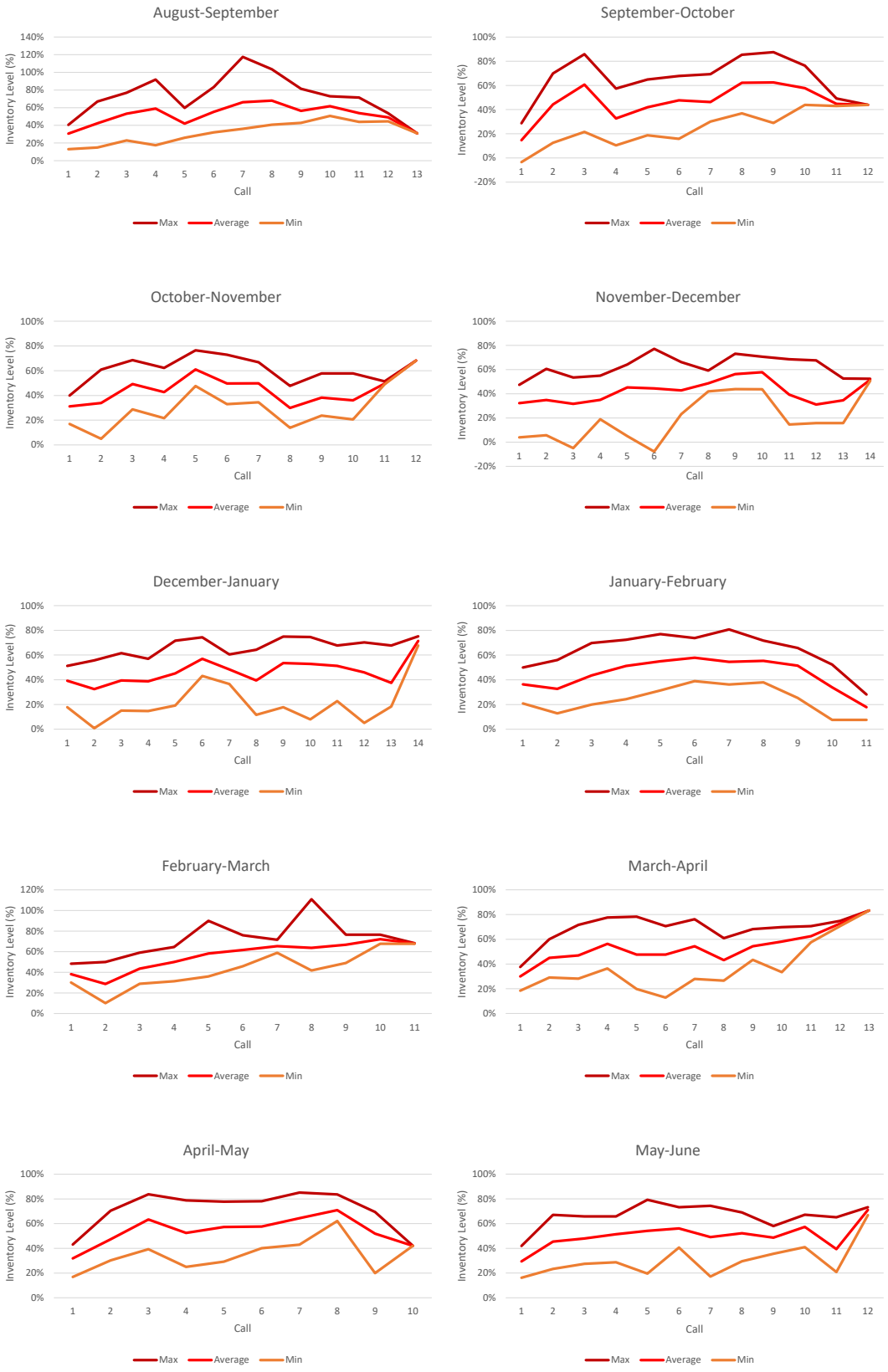

Figure 6.7: Inventory Level - Multi-Product - SA

	Waiting Time (days)	Violation (units)	Transformation (units)	Computational Time (s)
08-09	6.7	304	19	197
09-10	7.4	170	13	147
10-11	6.1	111	16	162
11-12	5.7	194	16	247
12-01	6.3	164	11	186
01-02	6.2	134	20	188
02-03	5.3	198	21	142
03-04	4.5	167	13	136
04-05	4.9	31	19	212
05-06	4.5	47	10	204

Table 6.3: General Information - Multi Product - SA

in Chapter 5. But this apparently little change can bring significant changes, that will be discussed later.

Firstly, Figure 6.8 shows the main costs obtained through the Hybrid VNS methodology, as explained in Figures 6.2 and 6.5 . The analysis remained the same, with the inventory costs standing out impacting the total cost, due the violations. In Figure 6.9, once again as Figures 6.3 and 6.6, shows the maximum, average and minimum values for the violations and total cost for all evaluated instances.

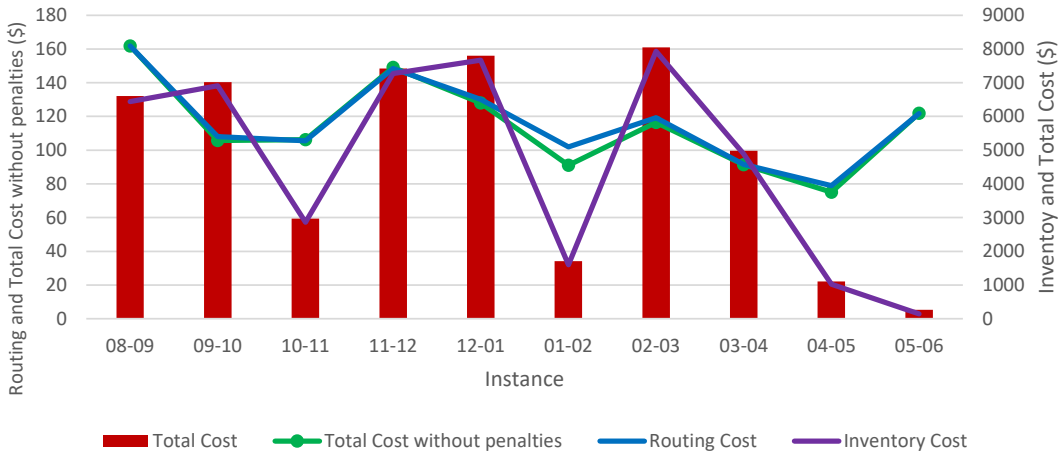

Figure 6.8: Costs - Multi Product - VNS

In Figure 6.10 the maximum, average and minimum inventory level for the ten instances are shown. Except instance 10-11, in which the inventory level stays below 0% for call number three, all other instances are inside the range.

Table 6.4 shows average values for some VNS information: the computational time is about 22 minutes with violation of only 31 units; the product

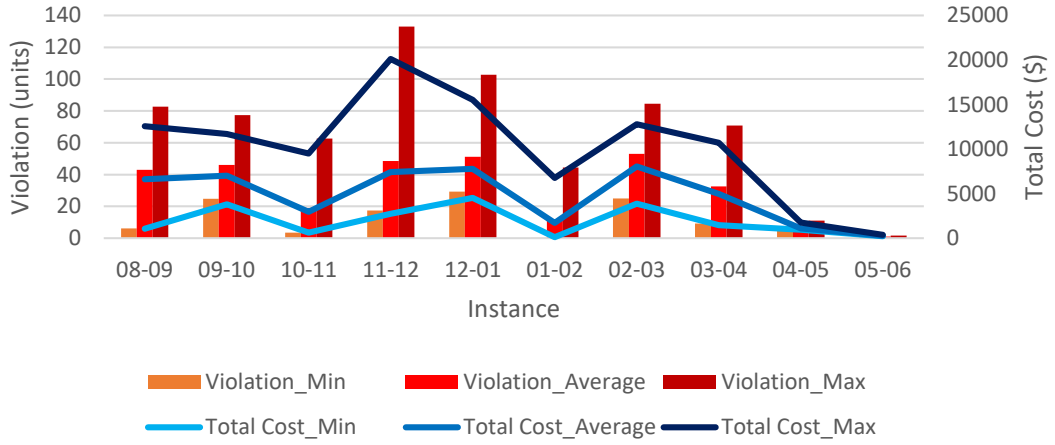

Figure 6.9: Violation - Multi Product - VNS

transformation is applied in 16 units and the ships waiting time stays in 5 days.

	Waiting Time (days)	Violation (units)	Transformation (units)	Computational Time (s)
08-09	5.2	43	18	1824
09-10	5.8	46	18	1166
10-11	6.1	19	14	1155
11-12	5.5	48	15	1368
12-01	3.3	51	10	1185
01-02	4.7	11	18	1246
02-03	4.2	53	26	1617
03-04	5.6	33	17	1512
04-05	5.4	7	16	1226
05-06	4.8	1	13	1153

Table 6.4: General Information - Multi Product - VNS

6.4 Methodologies Comparison

In this section, firstly, the local search with a Simulated Annealing probability factor (SA) and the Hybrid Variable Neighborhood Search (Hybrid VNS) results are compared. The first aspect to compare is the total cost and its maximum, average and minimum values are, shown in Figure 6.11. It is possible to see that even the minimum total cost obtained with the SA method, in many instances, can be higher or very similar to the maximum total cost obtained by the Hybrid VNS method. Besides, the SA range is much larger than the Hybrid VNS, i.e., the maximum and minimum total cost values are closer in the Hybrid VNS, indicating a smaller deviation of the ten runs. Figure 6.12 shows the maximum, average and minimum violation for both metaheuristics and reflects the reason that the Hybrid VNS total costs are much lower than the

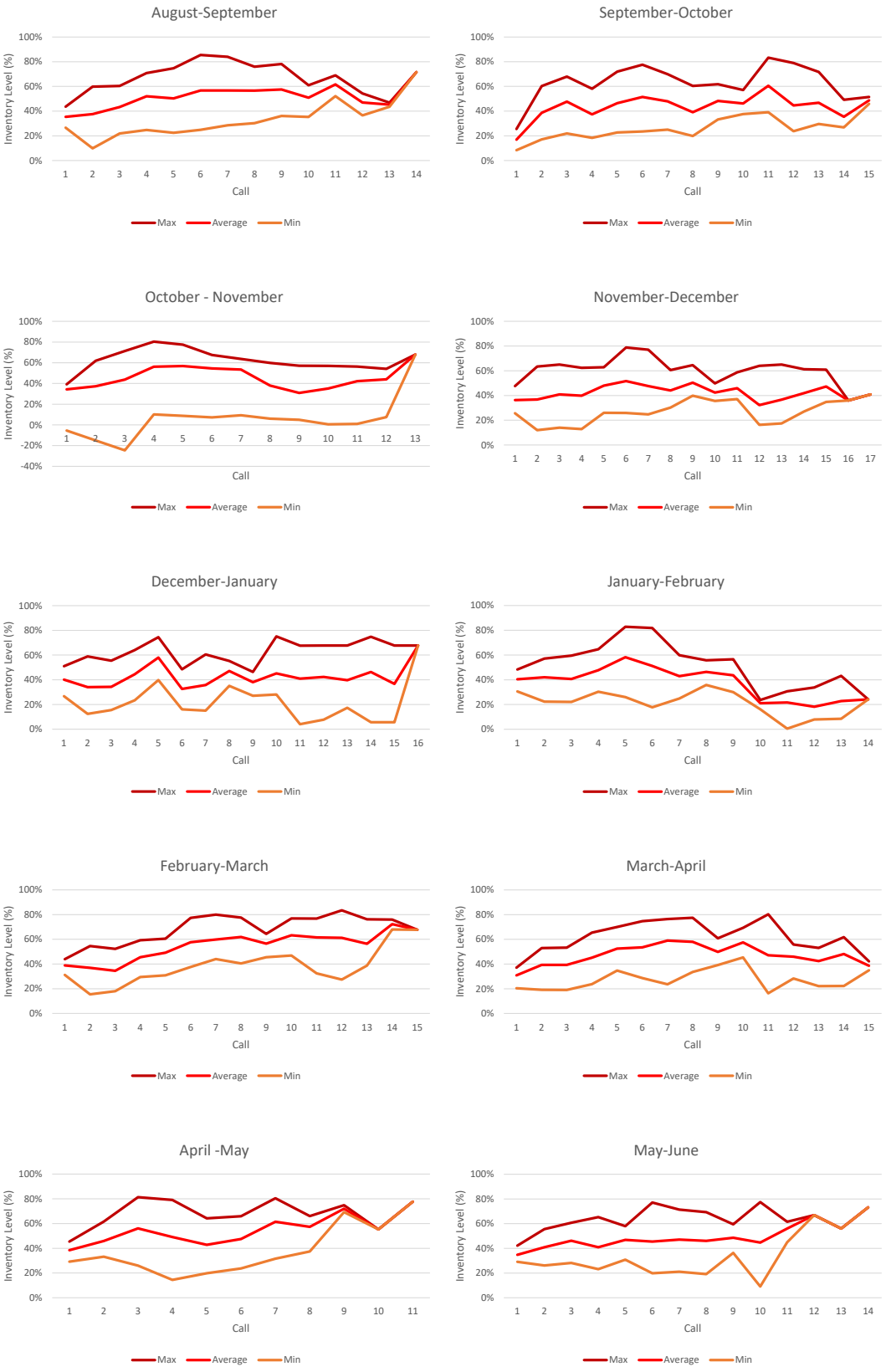

Figure 6.10: Inventory Level - Multi-Product - VNS

SA ones: the violation in Hybrid VNS are about five times smaller than the SA violations. In the Figure 6.13, the violations are divided in two different types: minimum (below 0%) and maximum (above 100%) inventory levels violations, from S_{ip}^{MN} and S_{ip}^{MX} respectively. The minimum violations are larger than the maximum violations, indicating that the most part of violations is due to the lack of product in ports, not satisfying the demand. Another important aspect concerns the low values for standard deviations. It shows that the Hybrid VNS method performs well in realistic instances, providing solutions with low variability in practice, regardless of the chosen seed.

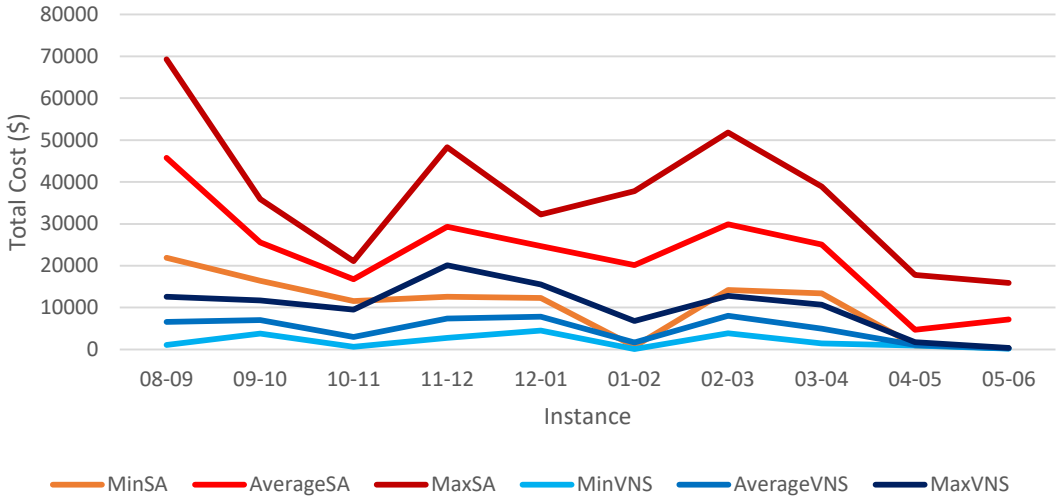

Figure 6.11: Costs Comparison SA vs VNS

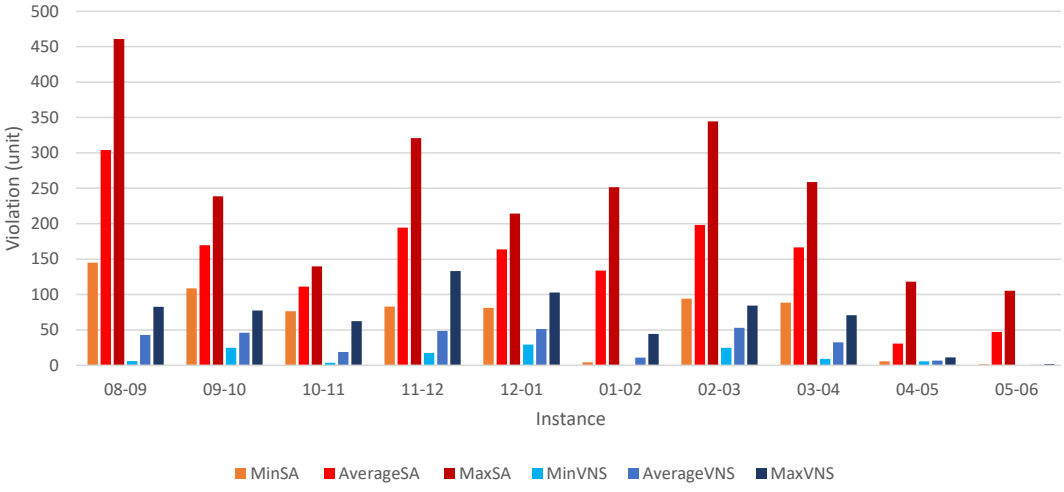

Figure 6.12: Violation Comparison SA vs VNS - Instance

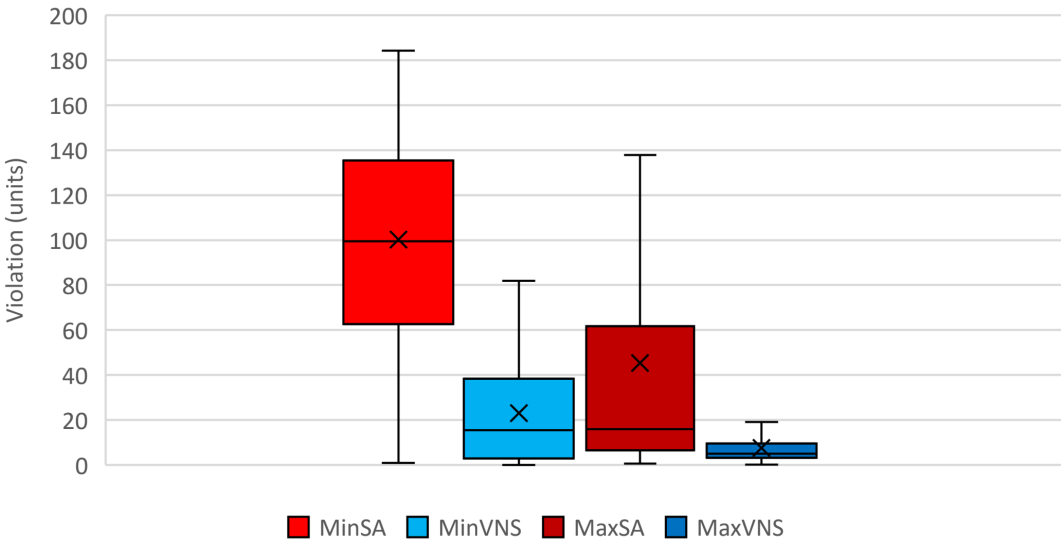

Figure 6.13: Violation Comparison SA vs VNS- general

Figure 6.14 shows the penalties costs and the total cost deducted by the penalties for SA and Hybrid VNS approaches. The costs, not considering the violation penalties, are very similar for both methodologies.

Figure 6.14: Penalties Comparison SA vs VNS

To conclude this first comparison between both metaheuristics, the number of iterations of the SA is increased to reach the VNS average computational time. In Figure 6.15 the results for inventory violation are shown, considering now the increased iteration number of SA. It is possible to see that now SA results are very competitive with VNS results, since more searches occur. Despite that, VNS results are still better, i.e., the total inventory violation amount is smaller. Therefore, the following analyzes will consider the VNS metaheuristic.

Aiming to compare the obtained results with methodologies developed in this work, we took the final routes generated by Costa (2018) and calculated

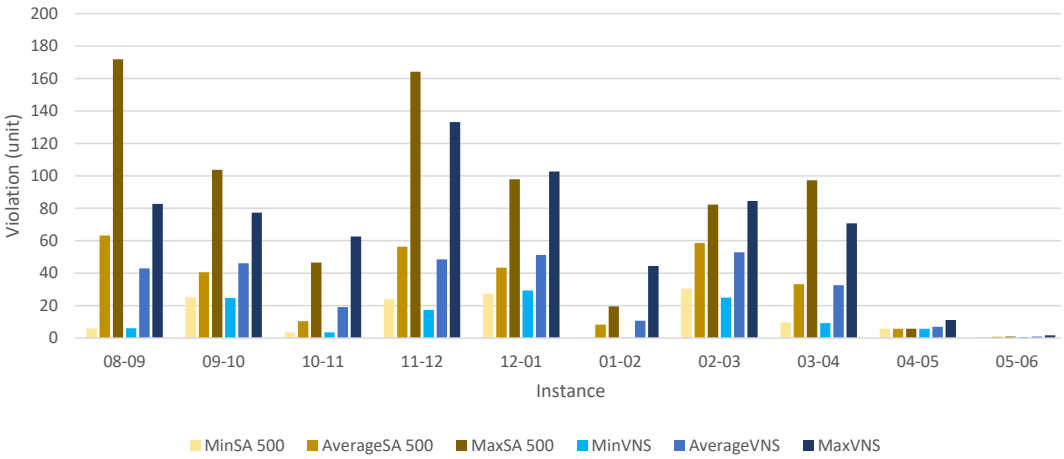

Figure 6.15: Violation increasing SA iteration number

their solution values using this work's objective function for the continuous-time model, and call it *Best Known Solutions (BKS)*. It is important to note that, despite Costa (2018) final routes were obtained using a different approach, they are the best available results for the evaluated instances to make a comparison.

The total costs without the penalties are shown in Figure 6.16. The figure shows the costs, considering routing and operational costs, disregarding penalization costs for Hybrid VNS and BKS. The results indicate a significant difference in the costs of our solutions and the ones obtained by Costa (2018).

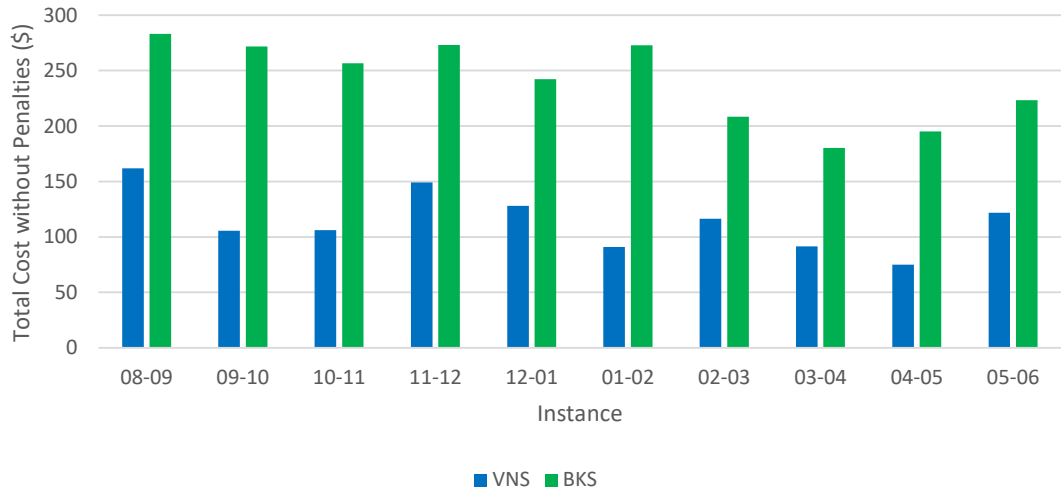

Figure 6.16: Costs Comparison VNS vs BKS

The final comparison is between the violations found by each approach, regarding inventory and draft limits in ports. In Figure 6.17, we analyze the distributions of the violations for the minimum inventory level at each port (S_{ip}^{MN}), and for the maximum inventory limit at each port (S_{ip}^{MX}). Note that,

in both cases, Hybrid VNS performs significantly better than the method proposed by Costa (2018).

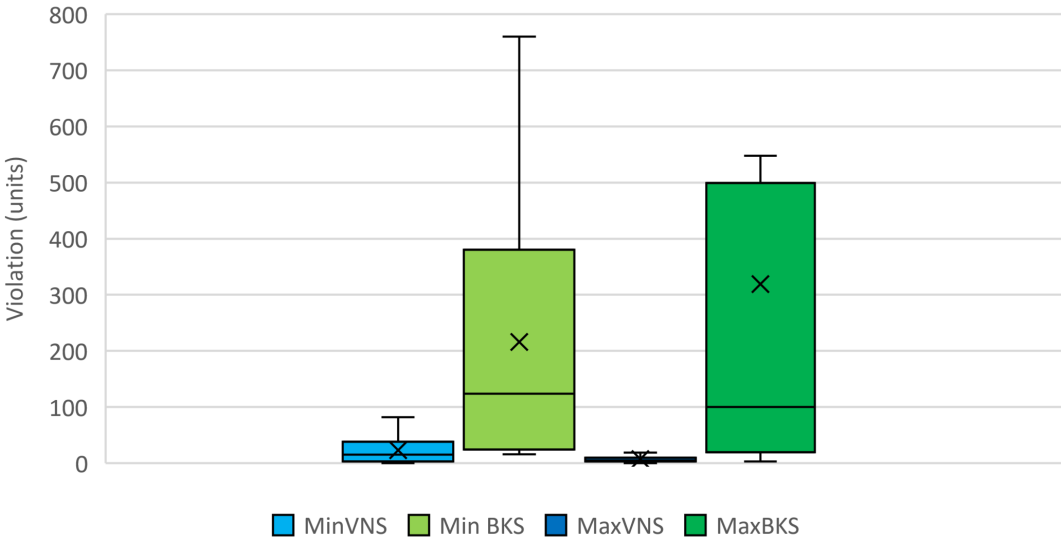

Figure 6.17: Violation Comparison VNS vs BKS

7
Conclusions and Future Works

The Maritime Inventory Routing Problem is widely studied by several authors considering different extensions of the basic case (Christiansen and Fagerholt 2009), as well as different methods of solution. Through the literature review it was noticed that there are opportunities for improving the solution methods and extending formulations to consider realistic aspects of the problem.

Through metaheuristics combined with mathematical models it was possible to obtain satisfactory results for a real-life multi-product maritime inventory routing problem (Multi-product MIRP) applied to the oil and gas industry. The method applies random movements to modify the routing solution, running mathematical formulations to optimize the inventory levels iteratively in the search for better solutions.

Experiments were conducted in a set of ten real-life instances with considerably large size, with nine products, 18 ports, and a fleet size ranging between 12 and 15 ships. A relevant aspect is the difficulty of finding feasible solutions to these instances. Another interesting characteristic regards the fact that some ports do not have mandatory demands, while in other cases, products can be replaced by others with similar or higher quality without affecting the cost of the solution, giving more opportunity to meet the demands in some ports. Our approaches present good results with low variability, showing to be useful in practice.

As positive results of this work, we highlight low, non-repetitive inventory violation in ports. For the methodology using a continuous-time model, the results are even better once, in reduced computational time, inventory violations remain low or non-existent, depending on the evaluated scenario and the metaheuristic used. For the Hybrid Variable Neighborhood Search, average times are around 22 minutes per run, while for Simulated Annealing, the average times are around 3 minutes per round. These results are remarkable and allow its practical application for real cases.

To evaluate our approach, we compared it with the solutions provided by Costa (2018). All routes proposed by the author were reevaluated according to our criteria, making it possible to compare with the SA and hybrid

VNS methods. New best solutions were provided for all instances with fewer inventory violations and at a lower cost. In addition, the computational time is satisfactory.

There are still aspects not covered in this work and maybe the object of study of future works, such as the stochastic version of MIRP. Few studies take into account the various uncertainties associated with this problem (Agra et al. 2015). In maritime transport, uncertainties are frequent and often related to climatic conditions, ship reliability, port delays. These uncertainties impact the ship's time, whether in navigation or operation, and can cause undesirable effects such as higher costs, fleet inefficiency, and greater inventory violations. Another important characteristic of real problems that could be evaluated is the trans-shipments in terminals. It would also be interesting to assess the impact of cargo segmentation on ships.

Bibliography

Agra, A., Christiansen, M., Delgado, A., and Hvattum, L. M. (2015). A maritime inventory routing problem with stochastic sailing and port times. *Computers & Operations Research*, 61:18–30.

ANTAQ (2019). Agência Nacional de Transportes Aquaviários – Anuário Estatístico. `http://portal.antaq.gov.br/wp-content/uploads/2020/02/Anu%C3%A1rio-2019-vFinal-revisado.pdf`. Acessado em 12/06/2020.

Archetti, C. and Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246.

Bertazzi, L., Coelho, L. C., De Maio, A., and Laganà, D. (2019). A matheuristic algorithm for the multi-depot inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 122:524–544.

Bertazzi, L., Savelsbergh, M., and Speranza, M. G. (2008). Inventory routing. In *The vehicle routing problem: latest advances and new challenges*, pages 49–72. Springer.

Christiansen, M. and Fagerholt, K. (2009). Maritime inventory routing problems. *Encyclopedia of optimization*, 2:1947–1955.

Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, Ø., Kloster, O., and Lund, E. H. (2011). Maritime inventory routing with multiple products: A case study from the cement industry. *European Journal of Operational Research*, 208(1):86–94.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

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., Laganà, D., Musmanno, R., and Vocaturo, F. (2015). A decomposition-based heuristic for the multiple-product inventory-routing problem. *Computers & Operations Research*, 55:153–166.

Costa, L. G. V. d. (2018). Matheurísticas para a roteirização de navios com estoques e múltiplos produtos. Master's thesis, Departamento de Engenharia Industrial. PUC-Rio.

Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Per Olav, M., Teistklub, G., and Reistad, A. (2007). Omya hustadmarmor optimizes

its supply chain for delivering calcium carbonate slurry to european paper manufacturers. *Interfaces*, 37(1):39–51.

Diz, G. (2017). *Maritime Inventory Routing: a practical assessment and a robust optimization approach*. PhD thesis, Departamento de Engenharia Industrial. PUC-Rio.

Hemmati, A., Hvattum, L. M., Christiansen, M., and Laporte, G. (2016). An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. *European Journal of Operational Research*, 252(3):775–788.

Hemmati, A., Hvattum, L. M., Fagerholt, K., and Norstad, I. (2014). Benchmark suite for industrial and tramp ship routing and scheduling problems. *INFOR: Information Systems and Operational Research*, 52(1):28–38.

Homsi, G., Martinelli, R., Vidal, T., and Fagerholt, K. (2020). Industrial and tramp ship routing problems: Closing the gap for real-scale instances. *European Journal of Operational Research*, 283(3):972–990.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Mjirda, A., Jarboui, B., Macedo, R., Hanafi, S., and Mladenović, N. (2014). A two phase variable neighborhood search for the multi-product inventory routing problem. *Computers & Operations Research*, 52:291–299.

Moin, N. H., Salhi, S., and Aziz, N. (2011). An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *International Journal of Production Economics*, 133(1):334–343.

Munguía, L.-M., Ahmed, S., Bader, D. A., Nemhauser, G. L., Shao, Y., and Papageorgiou, D. J. (2019). Tailoring parallel alternating criteria search for domain specific mips: Application to maritime inventory routing. *Computers & Operations Research*, 111:21–34.

Papageorgiou, D. J., Cheon, M.-S., Harwood, S., Trespalacios, F., and Nemhauser, G. L. (2018). Recent progress using matheuristics for strategic maritime inventory routing. In *Modeling, Computing and Data Handling Methodologies for Maritime Transportation*, pages 59–94. Springer.

Papageorgiou, D. J., Nemhauser, G. L., Sokol, J., Cheon, M.-S., and Keha, A. B. (2014). Mirplib–a library of maritime inventory routing problem instances: Survey, core model, and benchmark results. *European Journal of Operational Research*, 235(2):350–366.

Popović, D., Vidović, M., and Radivojević, G. (2012). Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Expert Systems with Applications*, 39(18):13390–13398.

Rodrigues, V. P., Morabito, R., Yamashita, D., Da Silva, B. J., and Ribas, P. C. (2016). Ship routing with pickup and delivery for a maritime oil transportation system: Mip model and heuristics. *Systems*, 4(3):31.

Siswanto, N., Essam, D., and Sarker, R. (2011). Solving the ship inventory routing and scheduling problem with undedicated compartments. *Computers & Industrial Engineering*, 61(2):289–299.

Song, J.-H. and Furman, K. C. (2013). A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 40(3):657–665.

Stanzani, A. d. L., Pureza, V., Morabito, R., Silva, B. J. V. d., Yamashita, D., and Ribas, P. C. (2018). Optimizing multiship routing and scheduling with constraints on inventory levels in a brazilian oil company. *International Transactions in Operational Research*, 25(4):1163–1198.

Uggen, K. T., Fodstad, M., and Nørstebø, V. S. (2013). Using and extending fix-and-relax to solve maritime inventory routing problems. *Top*, 21(2):355–377.

UNCTAD (2019). United Nations Conference on Trade and Development – Review of Maritime Transport. `https://unctad.org/en/PublicationsLibrary/rmt2019_en.pdf`. Acessado em 12/06/2020.