Iuri Martins Santos

Mathematical Programming Models and Local Search Algorithms for the Offshore Rig Scheduling Problem

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 em Engenharia de Produção.

Advisor: Prof. Luciana de Souza Pessôa

Co-Advisor: Prof. Silvio Hamacher

Rio de Janeiro
March 2018

Iuri Martins Santos

Mathematical Programming Models and Local Search Algorithms for the Offshore Rig Scheduling Problem

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 em Engenharia de Produção. Approved by the undersigned Examination Committee.

Prof. Luciana de Souza Pessôa
Advisor
Departamento de Engenharia Industrial - PUC-Rio

Prof. Silvio Hamacher
Co-advisor
Departamento de Engenharia Industrial - PUC-Rio

Prof. Laura Silvia Bahiense da Silva Leite
Universidade Federal do Rio de Janeiro - UFRJ

Prof. Simone de Lima Martins
Universidade Federal Fluminense - UFF

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico - PUC-Rio

Rio de Janeiro, March 1 st, 2018

Iuri Martins Santos

The author graduated in Industrial Engineering at PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro) in July 2015, with minors in Risk Analysis and Entreoreneurship. Before joining the master program of the Industrial Engineering Department (DEI) of PUC-Rio in the research field of Transportation & Logistics, the author was Chief Operating Officer in an e-commerce startup called WeWaant.com. Futhermore, the author has expertise in the area of Oil & Gas, having completed an extension course on "Structure of the Oil, Gas and Products Industry" in CCE/PUC-Rio and worked as an intern for Petrobras in the department of Maritime Transportation Intelligent and Strategy between 2014 and 2015. The author works as a scholarship researcher at Tecgraf Institute since 2016, working on vessels, oilrigs and PLSVs scheduling projects. Currently, has been accepted in the selection process for the PhD program of Industrial Engineering at DEI of PUC-Rio.

Bibliographic data

Santos, Iuri Martins

Mathematical programming models and local search algorithms for the offshore rig scheduling problem / Iuri Martins Santos ; advisor: Luciana de Souza Pessôa ; co-advisor: Silvio Hamacher. – 2018.
134 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2018.
Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Programação de sondas. 3. Poços marítimos de petróleo. 4. Matheuristics. 5. Busca local. 6. Variable neighborhood descent. I. Pessôa, Luciana de Souza. II. Hamacher, Silvio. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

Acknowledgement

This dissertation could not be possible without the support of several people and institutions. I must thank and dedicate this section to all those people and companies that were in my life over the past years. Nonetheless, there some special contributors that I want to recognize.

First, I have to thank my family (particularly my mother Mariela, my father Saulo and my sister Maíra) for supporting emotionally and financially over these last years, teaching me values and ethics and being by my side in moments of joy, need and pain.

Secondly, I want to express gratitude to my advisors Professor Silvio Hamacher and Professor Luciana Pessôa. This dissertation would never be possible without theirs knowledges, guides, lessons, and motivations.

This research was supported by PUC-Rio, DEI, CAPES, Tecgraf and the studied company. I'm grateful for the great infrastructure of PUC, place where I achieved my bachelor degree in Industrial Engineering and now I am (hopefully) obtaining this Master of Science. It is a great university with wonderful teachers, staff and students. The Department of Industrial Engineering (DEI) was critical to my success supporting my researches. Particularly from the people at PUC-Rio, I thank my advisors and the professors Cyrino, Hugo, José Eugênio, Martinelli, Scavarda, Luiza Martins. I am also grateful for the support of my master colleagues (Diego Moah, Leonardo Bastos, Lucas Condeixa, Pierry Couto and others) and my bachelor colleagues. The financial help of CAPES allowed me and many others students to focus on full time for the post-graduate program and, thanks for its support, I was able to achieve better results and developed several researches during my master. The Tecgraf research institute was also extremely important during this work. Thanks to its resources and people, this research could advance. I thank my colleagues from Tecgraf that deeply supported me and provided several insights. In special, I thank Tiago Andrade for his knowledge in modeling and AIMMS, Luana Carrilho for her SQL and Rig Scheduling knowledge, Gabriela Ribas for her expertise in the Rig Scheduling problem, Victor for his know-how in heuristics and scheduling algorithms, Pedro Lobato for his Oil & Gas expertise and Danuza for her availability in solving several daily problems.

Last, I want to thank to my special friends (Paula, Ernani, Sidney, Bernardo, Lucas, Felipe's, Leon's, Pedro's, Artur, Bruno, Feuer, Alejandro, Guilherme, Gustavo and many others) and all those people that were around me providing emotional support and comfort.

Abstract

Santos, Iuri Martins; Pêssoa, Luciana de Souza (Advisor); Hamacher, Silvio (Co-Advisor). **Mathematical Programming Models and Local Search Algorithms for the Offshore Rig Scheduling Problem**. Rio de Janeiro, 2018. 134p. Dissertação de Mestrado - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

The offshore exploration & production (E&P) of Oil & Gas involves several complex and important operations and relies, mostly, in the use of rigs, a scarce and costly resource that oil companies need to properly plan and schedule. In the literature, this decision is called the Rig Scheduling Problem (RSP). However, there is not any study related to offshore wells and drilling activities with realistic objective functions. Aiming to fulfill this gap, this dissertation studies a rig scheduling problem of a real offshore company and proposes a matheuristic approach to determine a rigs fleet and schedule that minimizes the budget. Two mathematical models – one for rigs fleet minimization and another one that minimizes the rigs budget – and several heuristics – using local search (LS) and variable neighborhood descent (VND) algorithms with three neighborhood structures and also constructive methods – were developed and tested in two instances based on real data of the studied company. In the small instance, the programming model found slightly better solutions than the heuristic, despite requiring more computational effort. Nevertheless, in the large instance, the mathematical programming solutions present large gaps (over 11%) and an elevated computational time (at least 12 hours), while the heuristics can find similar (or even better) solutions in a shorter time (minutes), having 70 of 156 heuristics outperformed the mathematical models. Last, the matheuristic combination of the simplest mathematical model with the heuristics has found the best known solutions (BKS) of the large instance with a moderate computational effort.

Keywords

Rig scheduling problem; offshore wells; matheuristics; local search; variable neighborhood descent; oil & gas.

Resumo

Santos, Iuri Martins; Pêssoa, Luciana de Souza; Hamacher, Silvio. **Modelos de programação matemática e algoritmos de busca local para o problema de programação de sondas marítimas**. Rio de Janeiro, 2018. 134p. Dissertação de Mestrado - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

A exploração e produção (*E&P*) *offshore* de óleo e gás envolve várias operações complexas e importantes, como perfuração, avaliação, completação e manutenção de poços. A maioria dessas tarefas requer o uso de sondas, um recurso custoso e escasso que as companhias de petróleo precisam planejar e programar corretamente. Na literatura, este problema é chamado de Programação de Sondas. Todavia, existem poucos estudos relacionados aos poços marítimos e às atividades de perfuração e nenhum destes com funções objetivo e restrições realistas, como orçamento. Por isso, muitas empresas de petróleo têm fortes dificuldades no planejamento das sondas, resultando em grandes custos para elas.

Com o objetivo de preencher essa lacuna, esta dissertação estuda um problema de programação de sondas em uma empresa petroleira e propõe um método híbrido para determinar a frota de sondas e seu cronograma, que minimize o orçamento da empresa. Dois modelos de programação matemática – um para minimização das sondas e outro para minimizar seu orçamento com variações da unidade de tempo utilizada (dia ou semana) – e várias heurísticas – usando algoritmos de busca local e *variable neighborhood descent (VND)* com três estruturas de vizinhança e duas estratégias de busca (*first* e *best improvemment*) e métodos construtivos- foram desenvolvidos e testados em duas instâncias (uma pequena e uma grande), baseadas em dados reais da empresa do caso de estudo. As três estruturas de vizinhanças são baseadas em movimentos de *insert*, uma delas não permite alterar as datas de alocação das tarefas na solução inicial, outra permite adiar tarefas e a última as posterga.

Os resultados indicaram a dificuldade no desempenho dos modelos matemáticos nas grandes instâncias e uma forte capacidade das heurísticas para encontrar soluções similares com muito menos esforço computacional. Na instância pequena, o modelo exato para minimizar o orçamento encontrou soluções um pouco melhores que a heurística (diferença de entre 0,4% e 5,6%), embora necessitando de mais esforço computacional, principalmente os modelos com unidades de tempo em dias. Porém, na instância maior, as soluções da programação matemática possuíram altos gap (mais de 11%) e altos tempos computacionais (pelo menos 12 horas), tendo o modelo matemático mais completo sido incapaz de encontrar soluções inteiras viáveis ou limites inferiores depois de mais de um dia rodando. Enquanto isso, as heurísticas foram capazes de encontrar soluções similares ou até melhores (desvios de -6% e 14% em relação a melhor solução exata) em um tempo muito menor, tendo 70 das 156 heurísticas desenvolvidas superado os modelos matemáticos. Além disso, os melhores resultados heurísticos foram utilizando algoritmos de *variable neighborhood descent (VND)* com estruturas de vizinhanças que realizavam movimentos de *insert* de tarefas em sondas existentes ou novas e permitiam postergar ou adiantar as tarefas das sondas. A abordagem hibrida foi comparada também com uma abordagem puramente heurística, tendo a primeira obtido melhores resultados.

Por fim, os resultados demonstram que o método híbrido proposto combinando o modelo matemático que minimiza o número de sondas com as heurísticas de busca local é uma ferramenta de suporte a decisão rápida e prática, com potencial para reduzir milhões de dólares para as empresas petroleiras do mercado *offshore*, com capacidade para encontrar cronogramas próximos da solução ótima com pouco esforço computacional, mesmo em instâncias grandes onde a maioria dos métodos exatos é muito complexa e lenta.

Palavras-chave

Programação de sondas; poços marítimos de petróleo; matheuristics; busca local; variable neighborhood descent; óleo & gás.

Table of Contents

List of Algorithms

List of Charts

List of Figures

List of Tables

1
Introduction

The Oil & Gas sector plays an important role in the world and influences severely in the development and the economy of countries. BP (2017a) estimated that Oil & Gas companies have produced more than 57.4% of the global primary energy consumption in 2016. In addition, according to BP (2017b), oil and gas will remain as world's dominant fuels in 2035, supplying 55.1% of world's energy. Plus, petroleum is not only an energy resource, but also a major raw material for industries, such as fertilizers, plastic, road construction and pharmaceutical (Devold, 2013).

Nonetheless, the petroleum is a non-renewable resource found beneath the Earth's surface and, to satisfy the its demands, companies are exploiting oil and gas from increasingly deeper and difficult locations, such as deep-water and ultra-deep-water wells (Nummi, 2017; Manning, 2016). As a result, the Exploration and Production (E&P) of Oil & Gas involves several complex, expensive and risky operations, especially in offshore oilfields (Suslick & Schiozer, 2004).

One of most critical phases of the E&P is the construction and development of wells, which relies mainly in the operation of rigs. Rigs are a costly and scarce resource, whose daily rate can vary between US$ 50,000 and US$ 700,000, depending on the rigs type, market and its operational specifications (Kaiser & Snyder, 2013; Osmudsen *et al.*, 2010; IHS Markit, 2017). Companies hire rigs to perform several important activities in the wells, such as drilling, evaluation, completion and workover. An underestimated rigs fleet can result in oil production delays and opportunities losses that affect the wells profitability. On the other hand, an oversized fleet may lead to idleness costs of over 1 billion *Reais* (R$), about three hundred million dollars, as mentioned by Pamploma (2016). Consequently, these fleets of rigs require to be properly planned and scheduled to ensure that the right rigs will be available in the right place at the right time with the lowest cost possible.

However this decision making process, called as Rig Scheduling Problem, is an extremely hard problem, not only due to the variety and the quantity of activities, but also as a result of the uncertainties related to geological concepts (structure, reservoir seal and hydrocarbon charge), economic evaluations (costs, probability of finding and producing economically viable reservoirs, technology and oil price), the development and production (infrastructure, production schedule, quality of oil and operational costs and reservoir characteristic) (Suslick *et al.*, 2009). In the offshore rig scheduling problem, the risks are even higher as the tasks are harder and the environmental conditions are subject to variable conditions such as weather and waves. All of these uncertainties add complexity to the scheduling problem, that is already naturally hard, and, consequently, escalate the need of decision support tools to assist in the planning and scheduling of rigs, minimizing risks and costs.

According to Reid *et al.* (2016), as a result of the complexity of the problem, most offshore companies failed to meet the delivery, budgetary and performance expectations. They also failed in hitting production targets and those that achieved the results state longer deliveries times and higher budgets. Due to the importance of this problem and industry, there has been a vast number of researches aiming to help those companies in their decision making process. Most of the studies of the rig scheduling problem focus on sub-problems for onshore wells and workover rigs. Only a few researches address issues for offshore drilling or completion rigs. The ones that approach the problem are either too superficial or unpractical for real application. In addition, almost none have objective functions that consider realistic budget parameters.

Aiming to fill this gap, this dissertation approaches a real rig scheduling problem in a major multinational of the Oil & Gas sector. The company of this case study is immersed in such environment full of uncertainties and high investments. In order to perform several of the E&P activities, the company needs to contract a large number of rigs, being a major player in the contract drilling rig market.

The studied Rig Scheduling Problem involves the sizing and scheduling in a medium plan horizon of a homogeneous fleet of offshore rigs, accountable for drilling, evaluation and completion activities in a Brazilian deep-water basin. In order to tackle this problem, this dissertation proposes the combination of mathematical models and heuristics in a matheuristical approach, hybrid algorithm that mix exact methods and heuristics (Raidl & Puchinger, 2008), to efficiently find a schedule that minimizes the rigs budget. The proposed mathematical models use mixed integer linear programming techniques while the heuristics use local search variable neighborhood search algorithms with movements specially adapted for this problem. The methods were tested in two instances based on real life scenarios, a small and a large one, representing two very different cases of the study company. Constructive heuristics and a purely exact method were also proposed and tested and they are compared with the matheuristic methods.

The presented problem and solution methods are important not only to the study company, but also the literature. The study company urgently needs a tool to support its rig scheduling problem, considering the company's special constraints and objective function, to be used in a medium term planning horizon and capable of finding strong solutions in an appropriate time. On the other hand, the current state of art of the literature lacks solution models considering realistic objective function and parameters for offshore rig scheduling problems. Therefore, the results of this study represents gains to the literature and to oil companies.

The dissertation is divided in eight sections, considering the current introductory section. Section 2 briefly describes the Oil & Gas production chain and its Exploration & Production phase, linking these subjects with the construction of offshore wells and the rig scheduling problem. Section 3 presents the Rig Scheduling Problem and performs a deep literature review in the issue. Later, Section 3 defines the study problem, its assumptions and presents the mathematical models. Then, Section 4 proposes the local search and constructive algorithms, describing the possible neighborhood structures, search strategies and others heuristics characteristics. After that, Section 6 presents the two instances used to test the methods and analyzes the computational experiments for the exact methods and the local search and constructive algorithms. Finally, Section 7 contains the final considerations of the study and proposes futures researches to be performed.

2
Oil & Gas Production Chain

In this section, we describe the Oil & Gas Production Chain and its divisions, focusing in the field development, one of the most important phases at the Exploration & Production (E&P) and a key stage for the production success. First, we briefly describe the main three Oil and Gas segments. Follow, we detail the upstream. Then, the E&P stages are presented. Finally, the process that includes the use of drilling rigs is exposed.

The Oil and Gas Chain is a horizontal supply chain commonly divided in two strands: upstream and downstream. The former is accountable for activities of exploration and production of the raw material (oil and natural gas). The latter is related with transportation, storage, commercialization, refine and the distribution of the oil and its products to the final consumer (Pelizaro, 2008; Ferreira Filho & Hamacher, 2015). Lately, others authors have been defining the stream in three segments: upstream, midstream and downstream. According to Ferreira Filho & Hamacher (2015), the upstream is the exploration and production of oil and gas, the midstream is responsible for the refine, transportation and importation of crude oil and natural gas, and the downstream is only the distribution and resale of oil products. Another classification for the oil and gas systems is proposed by Devold (2013), which divides it in: exploration (prospecting, seismic and drilling activities before the wells development); upstream (production and stabilization of oil and natural gas); midstream (gas treatment and oil and natural gas transportation); downstream (stage that process oil and natural gas into it products and distribute and sale it to final costumers); petrochemical (production of chemical products using hydrocarbons). These three different classifications are illustrated in Figure 1. In this work, we will use the classification from Ferreira Filho & Hamacher (2015) that divides the chain in upstream, midstream and downstream. Next, we focus in the upstream part, also known as Exploration & Production (E&P), of oilfields.

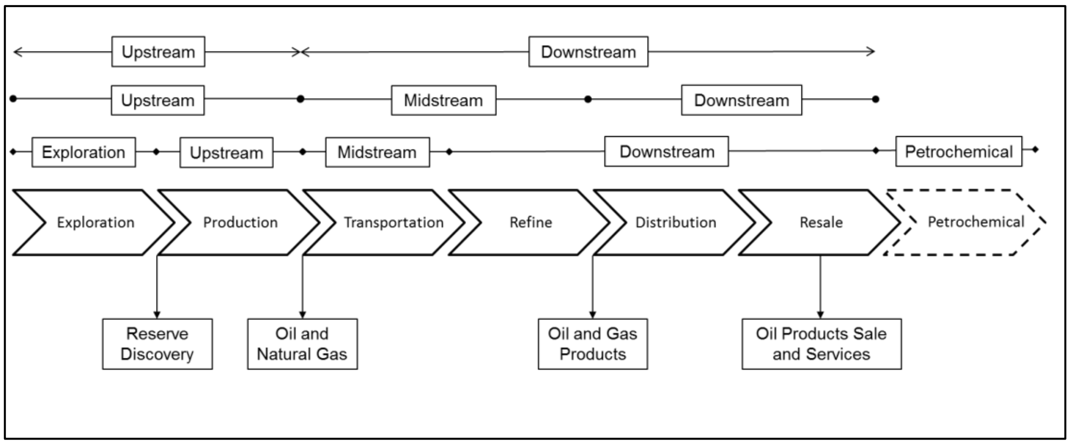

Figure 1 - Oil & Gas Supply Chain
(Source: the author; Ferreira Filho & Hamacher, 2015).

2.1. Exploration & Production of Oil & Gas

Exploring and producing hydrocarbons is a highly difficult and risk business, especially in the offshore sector (Suslick & Schiozer, 2004). Because of that, careful planning and tools to support decision-making are required (Shakhsi-Niaei *et al.*, 2013). However, in order to achieve those accomplishments, it is important to understand the E&P phase properly. Therefore, in this section, we present the Exploration & Production of oil and gas, which goes from the discovery of oil fields to theirs abandonments, as described in Figure 2:

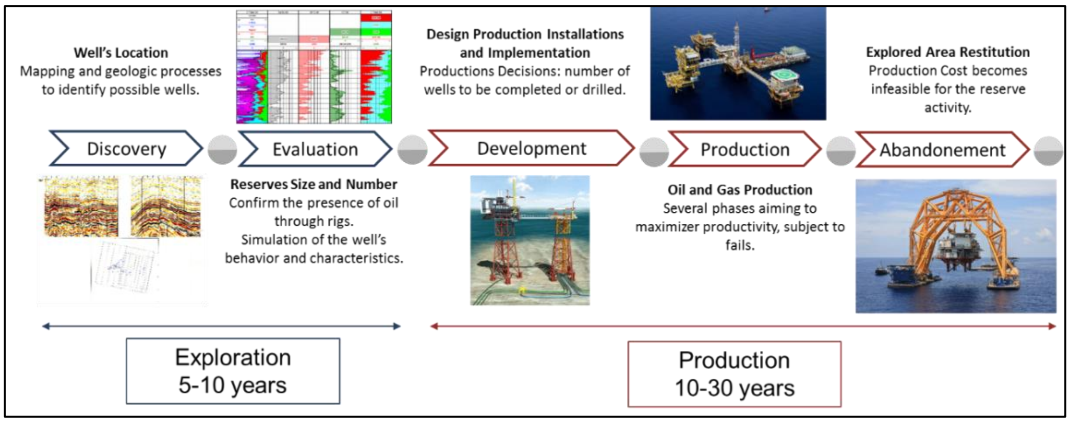
Well's Location
Mapping and geologic processes to identify possible wells.
Design Production Installations and Implementation
Productions Decisions: number of wells to be completed or drilled.
Explored Area Restitution
Production Cost becomes infeasible for the reserve activity.
Discovery
Evaluation
Development
Production
Abandonement
Reserves Size and Number
Confirm the presence of oil through rigs.
Simulation of the well's behavior and characteristics.
Oil and Gas Production
Several phases aiming to maximizer productivity, subject to fails.
Exploration 5-10 years
Production 10-30 years

Figure 2 - Framework for the E&P phase (Source: Santos *et al.*, 2017).

As shown in Figure 2, the E&P can take many years and it is a key part of the process to the company profitability. It can be separated in five main phases (Baker, 1996; IFP School, 2015; Pereira, 2005):

I. *Discovery* phase, which is the mapping and geological processes that identify possible oil fields;

II. *Evaluation* phase, when the possible presence of hydrocarbons is confirmed, or not, and evaluated through exploration wells drillers;

III. *Development* phase, responsible for important production activities and decisions, such as number of wells and if the well will be drilled or completed;

IV. *Production* phase, accountable for the oil production, can be extended through decades and has many different successive operations to increase productivity, to correct oil flow loss and to solve mechanical failures;

V. *Abandonment* phase, when the hydrocarbon production rate becomes economically invaluable and the reservoir is abandoned.

During these phases, wells are designed, constructed, operated and abandoned. Wells can be classified as injectors or producers. The firsts pump gas or water in the reservoir to maximize oil productions, while the seconds produce oil or natural gas. As explained before, a well life cycle can take years or decades with many complex and expensive operations and can be separated into four main stages (Offshore Center Danmark, 2010; IOM3, 2015):

i. *Design* or *Planning* phase, when decisions such as well objectives and schedules are made. Equipments and contracts are usually ordered and placed in stage.

ii. *Construction* phase, accountable for activities after the wells design, such as drilling, testing, completion and commission of a well.

iii. *Operation* phase, after the completion, the well starts its operations, during which operations of workover (activities that temporally stop wells operation for downhole surveillance, improvement in wells performance and repair of failures) can be required.

iv. *Abandonment* phase, when the reservoir becomes uneconomic, the wells are closed after removing the wellhead and installing downhole cement plugs.

As pointed earlier, some of the main resources used in the exploration of oil and gas are the rigs. These structures are used in critical activities such as

Evaluation, Drilling, Completion and Workover. They are high complexity and expensive ships used to explore wells. There is a variety of oil rigs, each one with a purpose. The main offshore rigs, as illustrated in Figure 3, are: fixed rigs (oil platform used up to 300 meters water profundity); semisubmersibles rigs (floating platforms used up to 2,000 meters water profundity); jackup rigs (platform with elevating legs used up to 150 meters) and drillships (floating platforms constructed in a vessel hull used up to 2,000 meter water profundity) (Petrobras, 2014; IHS Markit, 2017).

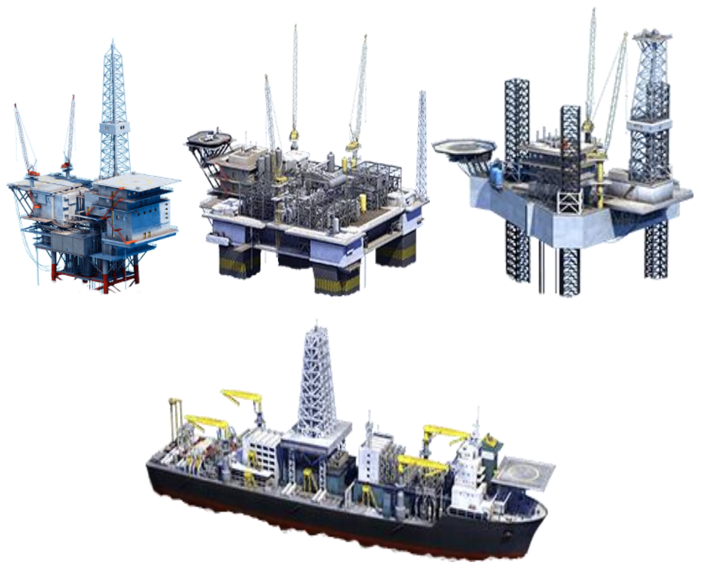

Figure 3. Examples of oil rigs, from left to right: fixed rigs, semisubmersibles rig, jackup rigs and drillships (Source: Petrobras, 2014).

Offshore oil rigs are used mainly in the development and production phases and in the construction and operation of well. Follow, we describe the building of offshore wells.

2.2. Construction of Offshore Wells

According to Culver (1998), the wells construction is one of most expensive and important phases of the Exploration and Production of Wells. As explained before, it can be divided mainly in Drilling and Completion activities. Both of them require the use of oil rigs.

In drilling, the well is created using an oil rig that rotates a drill bit. Follow, a steel pipe is placed inside the drilled hole and secured with cement, providing structural integrity and isolating the high pressures zones from the surface and from each other (Offshore Center Danmark, 2010).

The completion operations are started after the drilling with the purpose of "complete" the well, preparing it to produce oil or natural gas. There are two types of completions: cased-hole and open hole. In the first, perforations are made in the casing that pass through the reservoir, proving flow passages for the oil to pass to the production well. The second consists of just running the casing into the reservoir and leaving the end of the pipe open (Offshore Center Danmark, 2010).

These operations require offshore rigs and are subject to an environment full of uncertainties. Plus, the rigs must perform a variety of others complex tasks with scarce resources and an extensive horizon plan (Suslick *et al.*, 2009). Because of it, planning and scheduling of theirs tasks became key factor to success (Reid *et al.*, 2016).

In the next section, we will describe the Rig Scheduling Problem.

3
Rig Scheduling Problem (RSP)

The Rig Scheduling Problem (RSP) considers a set of wells, which have activities to be executed, and a set of resources available to perform these activities. The goal is to provide a minimum cost schedule to accomplish such activities while taking into account a complex list of operating and engineering constraints, the time window of activities, the rigs' availability and the predefined order to perform these activities. Therefore, a delay in one activity can affect all schedule and, consequently, increase the expenses planned (Bassi *et al.*, 2012).

The problems can be classified according to the oilfield: offshore and onshore. Usually, offshore fields are merge in environment with more complex, expensive and risky operations.

Another possible classification is by the use or not of routing techniques, dividing the problems in: Scheduling, when there is no need to consider the routes made by the rigs, usually when the distance between wells is despicable; Routing and Scheduling, when the distances are considerable and rigs transportations results in significant costs, requiring the rigs to be not only scheduled but also routed.

Other common classification divides the problem according to the types of activities considered: workover, drilling and completion. Some publications focus in only one activity and others take into account more than one. Some researches approach the general field development problem, such as Well Activity Scheduling, where rigs activities and others offshore equipments, such as Pipe Layer Support Vessels (PLSVs), need to be scheduled.

Last, the studies might also be classified by fleet of rigs and the wells: homogeneous and heterogeneous. In the former, all rigs and wells have the same characteristics and any rig can attend a well and perform any activity. The latter considers that not all rigs and wells are equal and a sub-set of rig can only be scheduled to a particularly set of wells or activities. Next, we perform a literature review in the Rig Scheduling Problem with the purpose of understanding the state of art and identifying gaps in the literature.

3.1. Literature Review

The use of Operations Research applied to Oil Rigs Scheduling dates back to over 50 years, to a time when Aronofsky & Williams (1962) addressed the economic planning and scheduling of oil production, proposing two different linear models. One model for scheduling oil production under a fixed rig and drilling schedule, and the other to schedule rigs and drilling operations under a pre-defined production curve. Both models aimed to maximize cumulative cash flow. Few years later, Barnes *et al.* (1977) investigated the workover rigs scheduling problem and proposed two approximate techniques. At that time, these models used to require a huge computational effort, disabling any practical application for the problems as explained by Pittman (1985). The problem discussion was only restarted in the 90's, as shown in Chart 1:

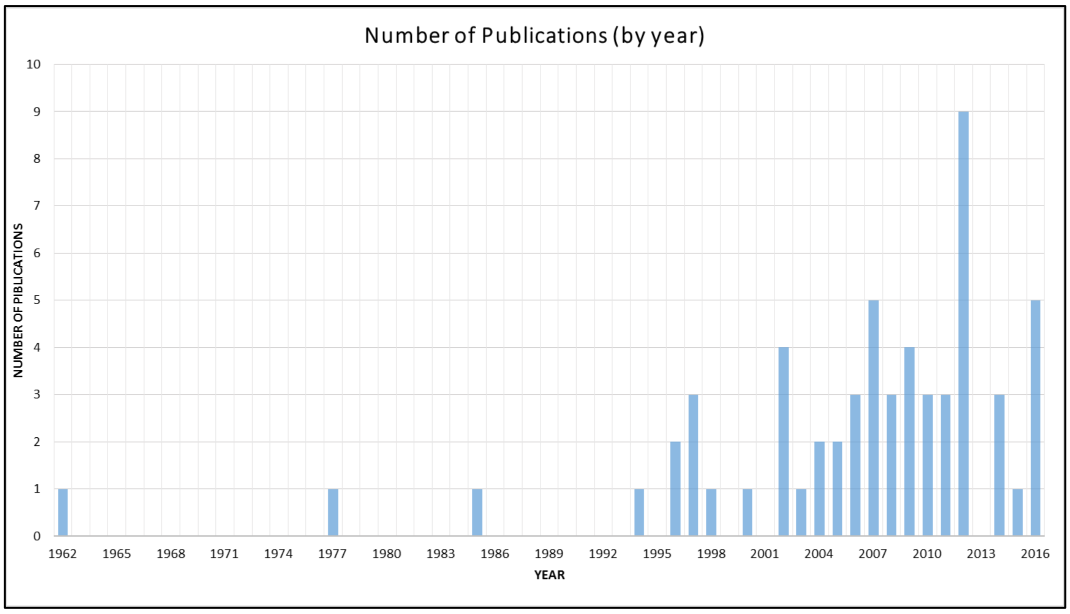

Chart 1. Operations research publications applied to rig scheduling (Source: the author).

With the improvement in computational complexity capacity and optimization methods, de Andrade Filho (1994) restarted the rigs scheduling problem discussion. He proposed the use of simulation through Eclipse reservoir simulator and polytope search to optimize the schedule of oil fields development projects, deciding development main dates plus the drilling rigs allocation and schedules.

Also in the 90's, Hasle *et al.* (1996), Eagle (1996), Iyer *et al.* (1998), Currie *et al.* (1997a) and Currie *et al.* (1997b) issued the rigs scheduling problem. Hasle *et al.* (1996) used the Constraint Reasoning (CR) method for the Well Activity Scheduling (WAS) problem, where resources such as drilling rigs and wireline cranes need to be appointed for several activities, mainly drilling, completation, perforation and logging. Eagle (1996) used Simulated Annealing (SA) to schedule drilling rigs and maximize the net presented value (NPV) in a multi-period horizon. Iyer *et al.* (1998) proposed a multi-period mixed integer linear programming (MILP) model with Branch & Bound technique to design and schedule an offshore oil field also aiming to maximize net present value. Their model aimed to decide the number, location and capacities of the platforms, the number and location of wells, the schedules for well and platforms installations, the scheduling of drilling rigs on well platforms and the oil production rate over the time horizon. Last, Currie *et al.* (1997a) and Currie *et al.* (1997b) presented a simplified mixed integer programming (MIP) model for the redevelopment and reservoir management of wells, deciding the projects, wells and drilling rigs to be used in a yearly basis. After these papers were published, the interest in rig scheduling models and algorithms raised.

Between 2000 and 2006, several papers about the Workover Rig Scheduling Problem were published. Paiva *et al.* (2000) proposed a Simulated Annealing (SA) algorithm for workover rigs scheduling and routing problem aiming to minimize total cost, including rigs expenses and losses in oil production due to the waiting time. The heuristic was based on a previously one from Paiva (1997). Gouvêa *et al.* (2002) proposed two evolutionary metaheuristics (a memetic algorithm and a transgenetic algorithm) for the same problem, having the second one consistently achieved better solutions.

Aloise *et al.* (2002) tested the use of Ant Colony heuristics combined with Path-Relinking to the Workover Rig Routing and Scheduling Problem. The authors tested different variations of the ant colony system with path relinking against other methods (also combined with path relinking), such as Genetic Algorithm (GA) and

Greedy Randomized Adaptive Search Procedure (GRASP). The Ant Colony Systems with path relinking had the best solutions in the majority of the analyzed instances. In addition, the use of path relinking applied to the ant colony heuristic assisted to reduce the computational time compared to the heuristic without it. Maia *et al.* (2002) developed a simplified Tabu Search-based heuristic for the problem and compared its results with those from Aloise *et al.* (2002). The basic Tabu Search found results similar to those obtained by others heuristics with path relinking. Accioly *et al.* (2002) used a constraint satisfaction problem (CSP or constraint programming) to the well activity scheduling with limited resources. Their approach schedule activities such as drilling, completion, workover and pipe lines connections aiming to maximize oil production and respect several constraints, including environmental rules, priorities, precedents, wells and rigs/ships characteristics. To explore the solution space, the authors used different search heuristics enabling the use of optimization solvers.

One year later, Rocha *et al.* (2003) presented a Variable Neighborhood Search (VNS) to the workover rig scheduling and routing problem. The authors tested three VNS variations and the best results found were obtained with a Cooperative Parallel Variable Neighborhood Search with path-relinking (CPVNS-path relinking) metaheuristic. The proposed heuristic selects a random neighborhood between nine different movements mixing swap and insert. The results found in this paper seem to be auspicious, but no comparison with the current state of art was presented, precluding us to compare it with the previous heuristics.

After modelling the WRRSP as a binary linear model, Costa & Ferreira Filho (2004) were able to use simple and intuitive heuristic designed specifically for the problem, named as Maximum Priority Three-Criteria Heuristic (MPTCH) or, in Portuguese, Heurística de Máxima Prioridade Tricritério (HMPT). The algorithm simplicity allowed it to be easily implemented in simulations, sensibility analysis and others decision support tools. However, Costa & Ferreira Filho (2005) tested the MPTCH heuristic against another intuitive and simple heuristic, referenced as Dynamical Assemble Heuristic (DAH) or, in Portuguese, Heurística de Montagem Dinâmica (HMD). The DAH heuristic outperformed the first method in all 300 examples tested and has proved to be a better algorithm that could be applied to small and large instances. These two methods, a Greedy Randomized

Adaptive Search Procedure (GRASP) heuristic and 300 real-life instances are also presented in the master thesis of Costa (2005). The GRASP method was implemented using a local search algorithm that tests all the instants of times in which tasks can begin without changing their execution orders. These three works and their examples have become a reference in the Workover Rig Scheduling Problem.

Trindade & Ochi (2004) proposed six variations of Greedy Randomized Adaptive Search Procedure (GRASP) and used the path relinking on them to improve their results. The authors performed several tests to analyze the heuristic behavior in different scenarios and its sensibility to the parameters variations. The authors proved the importance of using the appropriate local search method and found out that the GRASP model showed high robustness in the experiments. Later, Trindade & Ochi (2005) adapted the model to a hybrid GRASP with Path-Relinking, achieving even better results.

Follow, Aloise *et al.* (2006) used a Variable Neighborhood Search (VNS) heuristic to the Workover Rig Routing and Scheduling Problem. However, unlike the previous works, the authors considered a fleet of heterogeneous workover rigs, where rigs might differ in type and perform different levels of maintenance according to it. The VNS algorithm used several combinations of successive swap and insert movements, changing wells in a particular rig or different rigs or even entire rigs schedules. The heuristic was tested in real-life instances, where the average estimated savings were of approximately US$ 107,000 per 15 days. The projected annual savings were of over 2.5 million dollars. As a result, the heuristic was implemented in the Brazilian oil company (*PETROBRAS*) responsible for the real-life instances.

Meanwhile, a Genetic Algorithm (GA) was proposed by Alves & Ferreira Filho (2006) to the Workover Rigs Scheduling Problem with a homogeneous fleet and negligible travel times between wells. In 90% of the small instances, the heuristic was able to find solutions with GAP from the lower bound of under than 0.1%. For large problems, the metaheuristic was evaluated against Maximum Priority Three-Criteria Heuristic (MPTCH) and Dynamical Assemble Heuristic (DAH) from, respectively, Costa & Ferreira Filho (2004) and Costa & Ferreira

Filho (2005). In these scenarios, the GA method showed competitive results with minimum time.

Also in 2006, another Genetic Algorithm was developed by Vasconcellos & Ferreira Filho (2006) for the Resource Constraint Project Scheduling Problem (RCPSP) of an offshore oil field development. The model dealt with different activity scheduling decisions, each one needing a particular resource. Some activities needed oilrigs, such as well drilling and completion, and others demanded Pipe Layer Support Vessels. The method provided a powerful, practical and ease support decision tool to be used in the RCPSP.

Later, Oliveira *et al.* (2007) proposed a Scatter Search metaheuristic to the Workover Rig Routing and Scheduling Problem. The algorithm was compared against a GRASP heuristic and a Dynamical Assemble Heuristic (DAH), both previously developed by Costa (2005). The Scatter Search achieved better results than the GRASP heuristic in all instances and similar solutions to the ones found on DAH. To improve the GRASP with path relinking heuristic performance, Neves & Ochi (2007) presented an enhanced GRASP with Adaptive Memory Procedures (GRASP+AM). The inclusion of this learning procedure successfully powered up GRASP's efficiency. Others heuristics were introduced in Neves (2007)'s master thesis, such as Tabu Search and Iterated Local Search.

As noticed, just few of the studies above address the general Rig Scheduling Problem. Most researches were made focusing in the Workover Rig Scheduling Problem (WRSP) for onshore wells. However, since 2007, there has been an increase in publications applied to other rigs activity and offshore. First, Litvak *et al.* (2007) developed a procedure for British Petroleum's Top-Down Reservoir Modeling (TDRM) methodology using enhanced Genetic Algorithms to the offshore oil field development that included decisions such as drilling and completion rigs schedule for an available homogenous rigs fleet. Irgens & Lavenue (2007) and Irgens *et al.* (2008) target the drilling rig scheduling problem of a heterogeneous fleet. The authors developed two applications that provides real-time visualization and aims to maximize oil production and to minimize transportation costs through a special Stocastic Local Search. The algorithm used was designed for operational environments and provides fast and good solutions for real instances, but usually does not find optimal values.

Another Genetic Algorithm variation was proposed by Onwunalu *et al.* (2008). The authors enhanced the optimization model proposed by Litvak *et al.* (2007) through statistical proxies' procedures that use clustering-based techniques to build a statistical correlation between objective function and attributes, lowering the execution time.

Also in that year, Lasrado (2008) developed an application and a methodology for the workover rig scheduling problem using an adaptation of the reservoir simulation technique proposed years before by de Andrade Filho (1994). The application generates schedules and aims to minimize the rigs travelling distance and the number of rigs used, reducing transportation and contract costs. However, the methodology is manual and does not use optimization tools.

Falex (2009) proposed a Genetic Algorithm to the drilling rig scheduling problem of a heterogeneous rigs fleet aiming to minimize the total cost, sum of rigs contracts and production loss costs associated with the delay, in terms of present value. Gonçalves (2009) has also used the Genetic Algorithm metaheuristic for the drilling rig routing scheduling problem. Yet, the author took advantage of the ease of genetic algorithm modeling and introduced complex constraints, such as environmental and regulatory laws, and rigs displacement costs. The proposed model maximize the well's Expected Monetary Value (EMV), which is the sum of the products of the net present values and theirs probabilities of occurrence.

Back to the workover rig scheduling problem, Douro & Lorenzoni (2009) tested a Genetic Algorithm using the 2-opt improvement technique for homogeneous rigs fleet. The method used a heuristic based in the MPTCH technique from Costa & Ferreira Filho (2004) to find good initial solutions. The researchers compared their results with the MPTCH and others heuristics from the literature, such as DAH, GRASP and GA, respectively, from Costa & Ferreira Filho (2004), Costa (2005) and Alves & Ferreira Filho (2006). The proposed metaheuristic found better solutions outperforming the previous algorithms MPTCH, DAH and GRASP. Furthermore, the number of best solutions found in the algorithm was highly superior from the quantity found in the GA of Alves & Ferreira Filho (2006), enhancing the superiority of the suggested method.

Pacheco *et al.* (2009) also addressed the workover rig scheduling problem and implemented a heuristic called as Bubble Swap (BS) to minimize the oil production loss associated with the well's maintenance. The proposed Bubble Swap is an improvement heuristic that starts with a preliminary solution constructed by the Maximum Priority Three-Criteria Heuristic (MPTCH) from Costa & Ferreira Filho (2004) and searches for better solutions swapping wells from identical or different rigs and through bubble sorts. When compared with other methods, for instance GA (Alves & Ferreira Filho, 2006), MPTCH (Costa & Ferrreira Filho, 2004), DAH (Costa & Ferrreira Filho, 2005) and Scatter Search (Oliveira *et al.*, 2007), the Bubble Swap quickly found competitive solutions in the small instances and became a superior algorithm as the instances size increased.

Focusing in the Workover Rig Routing and Scheduling issue, Pacheco *et al.* (2010; 2012) propose a hybrid GRASP with Path-Relinking based in the GRASP from Costa (2005). After enabling the construction phase to use infeasible solutions, using the local search movement from Aloise *et al.* (2006) and introducing the path-relinking in the heuristic, the model became a reliable method, obtaining betters solutions than the ones from Costa (2005), Costa *&* Ferreira Filho (2005), Oliveira *et al.* (2007), Douro *&* Lorenzoni (2009) and Pacheco *et al.* (2009).

With the goal of achieving better solutions for the WRRSP, Lorenzoni & Polycarpo (2010) enhanced the Scatter Search (SS) method from Oliveira *et al.* (2007), using the MPTCH of Costa *&* Ferreira Filho (2004) as the solution generator for theirs heuristic. As result, the SS found 11 new best solutions and showed to be as powerful as the GA-2opt metaheuristic.

The rigs operation schedule is highly associated with their resources planning. Aiming to help this particular problem, Mazzini *et al.* (2010) propose a Mixed Integer Linear Programming (MILP) model that decides the rigs equipments and the drilling/completion rigs schedules, respecting rigs planned itinerary. The introduced model aims to minimize the sum of the resources contracts costs and the costs associated with the rigs tardiness and, according to the authors, it can generate millions dollars in annual savings.

Al Gharbi (2011) address the Drilling Rig Routing Scheduling Problem for onshore wells and a homogeneous rigs fleet. The author analyzed several constructive heuristics using the Dijkstra Algorithm with different objectives to be minimized, such as travel distance between wells, total distance, number of operation days and total cost (sum of moving and operation costs). After studying these scenarios, the author developed a heuristic mixing the characteristics with the best fit to a real case problem aiming to minimize the total costs. When compared with the case company's current process, the algorithm resulted in a higher number of operation days, but with 30% and 11.5% of improvement in total moved distance and total cost, respectively.

Aiming to find optimal or near-optimal solutions on the most difficult instances of the Workover Rig Routing and Scheduling Problem, Ribeiro *et al.* (2011) propose a simple (yet robust) Simulated Annealing-based heuristic. They propose a new algorithm, based on the one from Mauri & Lorena (2009) (used in the dial-a-ride problem), to generate the initial solution and apply the SA iterative improvement method using three different neighborhood: Re-order well, Re-allocate well and Swap wells. The first swaps wells from the same rig, the second choose reallocate (insert) the well in others rigs and the last one swap wells from different rigs. During each iteration, the heuristic randomly chooses a neighborhood strategy and generates a strong diversity with small and simple changes. In consequence, the algorithm produces powerful and fast solutions and beats most heuristics. The authors compared their results with other heuristics previously used, such as DAH, GRASP, SS, BS and GA-2opt – respectively from Costa & Ferrreira Filho (2005), Costa (2005), Oliveira *et al.* (2007), Pacheco *et al.* (2009) and Douro & Lorenzoni (2009) –, and the proposed Simulated Annealing outperformed all of the above methods.

Pacheco (2011) presented three heuristics – a Bubble Swap, a GRASP with path-relinking and a Memetic Algorithm (MA) – to minimize oil production losses when scheduling a fleet of homogeneous workover rigs on onshore wells. The author compared these methods with the SA heuristic from Ribeiro *et al.* (2011). For the instances analyzed, the Memetic Algorithm and the Simulated Annealing heuristic were consistently superiors than the others algorithms, with the second one outran the first.

In the next year, Ribeiro *et al.* (2012a) published an article applying three different heuristics to the Workover Rig Routing and Scheduling Problem with a homogeneous fleet and comparing their performance. The three proposed algorithm were: an Iterated Local Search (ILS) from Neves (2007), a Clustering Search (CS) and an Adaptive Large Neighborhood Search (ALNS). According to the authors, both CS heuristic and ALNS heuristic showed superior results than the ones found by the ILS, with the ALNS outperforming the other two methods. Meanwhile, Ribeiro *et al.* (2012b) tried to find exact solutions to an even harder issue, the WRRSP with a heterogeneous fleet. The authors proposed a Branch-Price-and-Cut (BPC) optimization algorithm – based on techniques such as Tabu Search (TS), column generator, ng-path-relaxation and subset-row inequalities –, enabling to solve practical and real-life examples with up to two hundreds wells and ten rigs. The presented BPC method found 50 new optimal solutions for the *benchmark* instances.

Three models and hybrid methods to workover rig problem were proposed by Duhamel *et al.* (2012) for different perspectives. The first was based on Aloise *et al.* (2006) workover rig scheduling MILP model. The authors reduced the number of variables and constraints and developed several improvements, such as allowing the wells to remain unattended and introducing inequalities to strengthen the formulation. The second was an open vehicle routing problem (OVRP) strategy for the WRRSP, with lifted constraints and better bounds. The last was a set-covering (C) formulation, obtained through a Dantzig-Wolfe decomposition of the second model. This C model was enhanced using column generations with GRASP and VND heuristics. All three proposed models aim to minimize the total production loss, associated with wells maintenance stop, under a homogeneous fleet of onshore rigs. When compared, all models were capable on finding optimal or near solution in small instances. However, the set-covering formulation has shown good and fast performance even in larger instances with between 30 and 60 wells, discovering solutions with gaps of just 0.1%.

A new approach for the Workover Rig Scheduling Problem was proposed by Bassi *et al.* (2012). The authors used simulation–optimization techniques, *i.e.*, a GRASP metaheuristic with simulation, to minimize total oil production loss considering a heterogeneous fleet of offshore oil rigs and expressive travel times between wells. First, a simulation phase, based on processing modeling, produces random samples of wells' service times and oil potentials. Then, the optimization algorithms (a simple constructive greedy heuristic and a GRASP model) find near optimal rigs schedules. Those two steps are repeated for several times and, last, a significant number of solutions are generated by simulation-optimization. The authors tested the model for several instances and different rigs fleet sizes. The analysis of the expected costs associated with the oil loss and rigs expenses unveiled an important trade-off between fleet size and total oil loss. On one hand, a larger number of rigs results in better performance measures (makespan, queue size and wells waiting time), but on the other hand increases the operating costs. Bassi *et al.* (2012) also proposed two matrices: a Well-Rig matrix and a Well-Order matrix. The former was used to determine well's probability of being scheduled in a particular rig and to flag the favorable allocations, enhancing the rig and wells analysis and revealing clustering trends between well and rigs according to theirs initial location. The latter determines the rig's chance of supporting a well in each queue position. The authors also provide a deep literature review on the WRSP with rich discussions, being a recommended reading for the Workover Rig Scheduling Problem.

From the oil field management perspective, Serra *et al.* (2012a; 2012b) proposed a Constraint Programming (CP) model to the offshore resource scheduling problem, where major assets, such as heterogeneous fleet of oil rigs and pipe layer support vessels, are scheduled aiming to maximize the oil production. Their model was based on a previously model presented by Serra *et al.* (2011), with few differences to tackle complex scenarios.

Two years later, others studies about the Workover Rig Scheduling Problem were published, such as in Ribeiro *et al.* (2014), Marques *et al.* (2014) and Bissoli (2014). Ribeiro *et al.* (2014) present and compare heuristics in the scheduling and routing of a limited heterogeneous onshore workover rigs fleet, aiming to minimize the oil production loss. The four heuristics analyzed were a Variable Neighborhood Search – from Aloise *et al.* (2006) –, a Branch-Price-and-Cut – extension of Ribeiro *et al.* (2012a) –, an Adaptive Large Neighborhood Search – adapted from Ribeiro *et al.* (2012a) – and a Hybrid Genetic Algorithm. All models consider realistic premises and constraints. For instance, each well have a particular loss rate, need a specific maintenance service and may not be attended. When compared, the BPC, the ALNS and the HGA results were consistently superior to the VNS heuristics, with deviations of more than 9% from the other methods. The BPC algorithm was able to find good solutions faster than the alternative methods, but their qualities are inferior than the ALNS and the genetic algorithm, methods that took more computational time. The HGA was the only method that found all the best solutions and usually outperformed the ALNS when compared over the time. The authors also made sensitivity analysis to detect performance changes due to the algorithms calibration.

Marques *et al.* (2014) presented a decision support system that uses a mixed integer linear programming model to size and schedule a homogeneous fleet of offshore workover rigs aiming to minimize the total number of rigs and to maximize their utilization. The authors used an objective function with weighted terms that forces the model to select rigs according to a priority list and to minimize their idleness. The proposed model was tested in three different scenarios and delivered exceptional results.

Bissoli (2014) addressed the Workover Rig Routing and Scheduling Problem using an Adaptive Large Neighborhood Search (ALNS) with a bi-objective function to minimize the oil production loss and the number of onshore rigs and so, according to the author, minimizing the rigs costs and total costs. This assumption is a simplification; in reality, a minimum fleet does not mean that costs associated with the chartering processes are minimized. The proposed ALNS was based in a previous one from Ribeito *et al.* (2012a) and was implemented in two models: one that considers a heterogeneous fleet and other that takes a homogeneous fleet into account. The authors also analyze the average deviation of 0.78% to the exact solutions found by a BPC algorithm from Ribeiro *et al.* (2012b) and the trade-off between oil production loss and rigs chartering costs.

Monemi *et al.* (2015) addressed the Workover Rig Scheduling Problem with a heterogeneous fleet. The authors proposed a new MILP model, based on arc-time-indexed formulations and two solution techniques: Branch-Price-and-Cut and hyper-heuristic. The former explores the new formulation and takes advantage of its dual decomposable efficiency in the Dantzig-Wolfe decomposition. The latter uses low-level heuristics (a set of constructive, improvement, perturbation and reconstructive heuristics) with a learning mechanism to find solution inside the heuristic space. According to the authors, the hyper-heuristic approach was able to find optimal or near optimal solutions for any instance with just a few seconds. Furthermore, the approximation method found solutions for larger instances, where the exact algorithm failed.

Another Mixed Integer Programming method was later proposed by Silva *et al.* (2016), this time applied to the Rig Routing and Scheduling Problem with a heterogeneous fleet of offshore oil rigs. The proposed model aims to minimize the oil production loss and the rigs' utilization costs, considering different activities, such as drilling, completion and workover, as well as other realistic assumptions, including non-linear routing and scheduling constraints. According to the authors, a branch-and-bound technique was used to solve a small instance based on a real scenario. After analyzing the proposed model, we noticed that the non-linear constraints could be easily eliminated using disjunctive programming. Once Silva *et al.* (2016) did not explained how they managed the non-linear constraints, it is possible that the computational effort required to solve the problem can be reduced using such linear programming formulation tricks, enabling the model to be applied to larger instances.

Peréz *et al.* (2016) proposed a decomposed reformulation of the binary linear model from Costa & Ferreira Filho (2004) for the Workover Rig Scheduling Problem with a homogeneous fleet of onshore oil rigs. The presented model had less variables and constraints and was tested in the benchmarks instances from Costa (2005). The decomposed mathematical model not only found new exact solutions for large instances with 125 wells and 10 rigs, but also outperformed others methods, such as GA-2opt (Douro & Lorenzoni, 2009), Memetic Algorithm (Pacheco, 2011) and Simulated Annealing (Ribeiro *et al.*, 2011), with better solutions and lower computational times. The authors also point the importance of testing such models in offshore wells. Bissoli *et al.* (2016) performed an extensive literature review on the Workover Rig Routing and Scheduling Problem, analyzing its drivers and trends. According to the authors, the current trends are to approximate the problem with the real life scenarios through new objective functions, mathematical formulations and solutions methods and dynamic and stochastic approaches.

In the same year, Flager (2016) addressed the Drilling Rig Scheduling Problem of onshore oil wells with a heterogeneous oil rigs fleet and proposed a multi-objective Genetic Algorithm with Monte Carlo Simulation aiming to, simultaneously, maximize the oil production rate and minimize the production cost. A framework of the proposed model and a description of its inputs and decisions are presented, but the author does not explain the algorithm or mathematical model used, making difficult a deeper analysis in the technique used. It is important to notice that the results found in the paper refer to a realistic and large instance with 237 wells. However, as there was a lack of details in models performance and specifications and no comparison with another model was presented, we cannot conclude about its benefits.

Last, Carilho & Villas Boas (2016) presented a decision support system (DSS) for rigs scheduling that uses a MILP model to simultaneously maximize the allocation of activities in rigs already contracted and minimize the fleet of rigs to be contracted. In order to reduce the complexity of the model, the authors decided to represent the tasks in a block structure and to use a time horizon in weeks. The DSS proposed also provides scenarios analysis with multiple indicators and reports. Besides, it was successfully implemented in a major Brazilian operator that needs to plan dozens of rigs and hundreds of wells in a time horizon of 15 years. According to the authors, the model managed to reduce in half the number of contracted rigs and eliminated several inconsistences from the manual approach that was used before by the company.

We summarize all rigs scheduling publications presented in this review in Table 1. The papers are classified according to authors, year of publication, type of oil field (offshore/onshore), rig activity type (drilling/completion/workover), activities types (single/multiple), problem definition, technique used (heuristic, mathematical programming, simulation, etc.), optimization direction (maximize/minimize) and objective function (net present value, oil production, oil production loss, makespan, etc.).

Table 1. Summary with Rig Scheduling publications and classifications (Source: the author).

Authors	Year	Oil field	Rig Activity	Activities Types	Rigs Fleet	Problem Definition	Technique	Optimization Direction	Objective Function
Aronofsky & Williams	1962	Under-ground	Drilling	Single	Homogeneous	Drilling Rig Scheduling	Parametric Linear Programming (PLP)	Maximize	Discounted Cumulative Cash Flow
Barnes *et al.*	1977	Any	Workover	Single	Homogeneous	Workover Rig Scheduling	Approximate Techniques	Minimize	Oil Production Loss
De Andrade Filho	1994	Offshore	Drilling	Single	Homogeneous	Oil Field Development and Drilling Rig Scheduling	Polytope Search + Reservoir Simulation	Maximize	Cash Flow
Hasle *et al.*	1996	Offshore	Drilling and Completion	Multiple	Homogeneous	Well Activity Scheduling	Constraint Reasoning (CR)	Minimize	Makespan
Eagle	1996	Onshore	Drilling	Single	Homogeneous	Drilling Rig Scheduling	Simulated Annealing (SA)	Maximize	Net Present Value (NPV)
Paiva	1997	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Simulated Annealing (SA)	Minimize	Total Costs = Transportation Costs + Production Loss
Currie *et al.*	1997a 1997b	Offshore	Drilling	Single	Homogeneous	Oil Field Development and Drilling Rig Scheduling	Mixed Integer Linear Programming (MILP)	Maximize	NPV
Iyer *et al.*	1998	Offshore	Drilling	Single	Homogeneous	Oil Field Development and Drilling Rig Scheduling	MILP + Branch and Bound	Maximize	NPV
Paiva *et al.*	2000	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Simulated Annealing (SA)	Minimize	Total Costs = Transportation Costs + Oil Production Loss
Gouvêa *et al.*	2002	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Evolutionary Algorithms (Memetic + Transgenic)	Minimize	Oil Production Loss
Aloise *et al.*	2002	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Ant Colony + Path-Relinking	Minimize	Oil Production Loss
Maia *et al.*	2002	Onshore	Workover	Multiple	Homogeneous	Workover Rig Routing and Scheduling	Tabu Search	Minimize	Oil Production Loss
Accioly *et al.*	2002	Offshore	Drilling, Completion and Workover	Multiple	Heterogeneous	Well Activity Scheduling	Constraint Programming	Minimize / Maximize	Makespan / Oil Production
Rocha *et al.*	2003	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Cooperative Parallel Variable Neighborhood Search (CPVNS) with Path-Relinking	Minimize	Oil Production Loss
Costa & Ferreira Filho	2004	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Maximum Priority Three-Criteria Heuristic (MPTCH)	Minimize	Oil Production Loss
Trindade & Ochi	2004	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Greedy Randomized Adaptive Search Procedure (GRASP) with Path-Relinking	Minimize	Oil Production Loss

Authors	Year	Oil field	Rig Activity	Activities Types	Rigs Fleet	Problem Definition	Technique	Optimization Direction	Objective Function
Costa & Ferreira Filho	2005	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Dynamical Assemble Heuristic (DAH)	Minimize	Oil Production Loss
Trindade & Ochi	2005	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Hybrid GRASP with Path-Relinking	Minimize	Oil Production Loss
Costa	2005	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	MPTCH, DAH and GRASP	Minimize	Oil Production Loss
Aloise *et al.*	2006	Onshore	Workover	Multiple	Heterogeneous	Workover Rig Routing and Scheduling	Variable Neighborhood Search (VNS)	Minimize	Oil Production Loss
Alves & Ferreira Filho	2006	Onshore	Workover	Single	Homogeneous	Workover Rig Scheduling	Genetic Algorithm (GA)	Minimize	Oil Production Loss
Vasconcellos & Ferreira Filho	2006	Offshore	Drilling and Completion	Multiple	Heterogeneous	Well Activity Scheduling	Genetic Algorithm (GA)	Minimize	Make-span
Oliveira *et al.*	2007	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Scatter Search	Minimize	Oil Production Loss
Neves	2007	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Iterated Local Search (ILS) and Tabu Search	Minimize	Oil Production Loss
Neves & Ochi	2007	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	GRASP with Adaptive Memory Procedures	Minimize	Oil Production Loss
Litvak *et al.*	2007	Offshore	Drilling and Completion	Multiple	Homogeneous	Oil Field Development and Drilling Rig Schedule	Genetic Algorithm (GA)	Maximize	Economic Indicators / Oil and Gas Recovery
Irgens & Lavenue	2007	Offshore	Drilling	Single	Heterogeneous	Drilling Rig Scheduling	Stochastic Local Search	Maximize / Minimize	Oil Production / Transportation Costs
Irgens *et al.*	2008	Offshore	Drilling	Single	Heterogeneous	Drilling Rig Scheduling	Stochastic Local Search	Maximize / Minimize	Oil Production / Transportation Costs
Onwunalu *et al.*	2008	Offshore	Drilling	Multiple	Homogeneous	Drilling Rig Scheduling	GA with statistical proxies	Maximize	NPV
Lasrado	2008	Offshore	Workover	Single	Homogeneous	Workover Rig Scheduling	Reservoir Simulation	Minimize	Distance Traveled and Number of Rigs
Falex	2009	Offshore	Drilling	Multiple	Heterogeneous	Drilling Rig Scheduling	Reservoir Simulation	Minimize	Net Present Loss (NPL)
Gonçalves	2009	Offshore	Drilling	Multiple	Heterogeneous	Drilling Rig Routing and Scheduling	Genetic Algorithm (GA)	Maximize	Expected Monetary Value (EMV)
Douro & Lorenzoni	2009	Onshore	Workover	Single	Homogeneous	Workover Rig Scheduling	Genetic Algorithm with 2-opt (GA-2opt)	Minimize	Oil Production Loss
Pacheco *et al.*	2009	Onshore	Workover	Single	Homogeneous	Workover Rig Scheduling	Bubble Swap	Minimize	Oil Production Loss
Mazzini *et al.*	2010	Offshore	Drilling and Completion	Multiple	Heterogeneous	Rigs Equipment Scheduling	Mixed Integer Linear Programming (MILP)	Minimize	Equipment's Charter and Rigs Tardiness Costs
Pacheco *et al.*	2010	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Hybrid GRASP with Path-Relinking	Minimize	Oil Production Loss
Lorenzoni & Polycarpo	2010	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Scatter Search	Minimize	Oil Production Loss
Al Gharbi	2011	Onshore	Drilling	Single	Homogeneous	Drilling Rig Routing and Scheduling	Constructive Heuristics with Dijkstra Algorithm	Minimize	Total Costs = Moving and Operation Costs

Authors	Year	Oil field	Rig Activity	Activities Types	Rigs Fleet	Problem Definition	Technique	Optimization Direction	Objective Function
Ribeiro *et al.*	2011	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Simulated Annealing (SA)	Minimize	Oil Production Loss
Pacheco	2011	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Bubble Swap (BS), GRASP with Path-relinking and Memetic Algorithm (MA)	Minimize	Oil Production Loss
Ribeiro et al.	2012a	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Iterated Local Search (ILS), Clustering Search (CS) and Adaptive Large Neighborhood Search (ALNS)	Minimize	Oil Production Loss
Ribeiro *et al.*	2012b	Onshore	Workover	Multiple	Heterogeneous	Workover Rig Routing and Scheduling	Branch-Price-and-Cut	Minimize	Oil Production Loss
Duhamel *et al.*	2012	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Column Generation, MILP and heuristics	Minimize	Oil Production Loss
Pacheco *et al.*	2012	Onshore	Workover	Single	Homogeneous	Workover Rig Routing and Scheduling	Hybrid GRASP with Path-Relinking	Minimize	Oil Production Loss
Gupta & Grossmann	2012	Offshore	Drilling	Single	-	Oil Field Development and Well Drilling Scheduling	Multi-period Nonconvex Mixed Integer Non Linear Programming (MINLP)	Maximize	Net Present Value (NPV)
Neves *et al.*	2012	Offshore	Drilling and Completion	Multiple	-	Rig Schedule Feasibility	-	-	-
Serra *et al.*	2012a 2012b	Offshore	Drilling, Completion and Workover	Multiple	Heterogeneous	Oil Field Development and Well Drilling Schedule Offshore Resource Scheduling Problem	Constraint Programming	Maximize	Oil Production
Bassi *et al.*	2012	Offshore	Workover	Multiple	Heterogeneous	Workover Rig Scheduling	Simulation-Optimization (w/ GRASP)	Minimize	Oil Production Loss
Ribeiro *et al.*	2014	Onshore	Workover	Multiple	Heterogeneous	Workover Rig Routing and Scheduling	VNS, Branch-Price-and-Cut, Adaptive Large Neighborhood Search and Hybrid-GA	Minimize	Oil Production Loss
Marques *et al.*	2014	Offshore	Workover	Single	Homogeneous	Workover Rig Scheduling	MILP	Minimize	Number of Rigs
Bissoli	2014	Onshore	Workover	Multiple	Homogeneous / Heterogeneous	Workover Rig Routing and Scheduling	MILP and ALNS	Minimize	Oil Production Loss and Rigs Costs
Monemi *et al.*	2015	Onshore	Workover	Multiple	Heterogeneous	Workover Rig Scheduling	MILP, Branch-Price-and-Cut and Hyper-Heuristic	Minimize	Oil Production Loss
Silva *et al.*	2016	Offshore	Drilling, Completion and Workover	Multiple	Heterogeneous	Rig Routing and Scheduling	MIP	Minimize	Oil Production Loss and Rig's Costs
Peréz *et al.*	2016	Onshore	Workover	Single	Homogeneous	Workover Rig Scheduling	Binary Linear Programming	Minimize	Oil Production Loss
Bissoli *et al.*	2016	Onshore	Workover	-	-	Workover Rig Scheduling	-	-	-

Authors	Year	Oil field	Rig Activity	Activities Types	Rigs Fleet	Problem Definition	Technique	Optimization Direction	Objective Function
Flager	2016	Onshore	Drilling	Multiple	Heterogeneous	Drilling Rig Scheduling	Multi-objective GA and Monte Carlo Simulation	Maximize / Minimize	Production Rate / Production Cost
Carilho & Villas Boas	2016	Offshore	Drilling and Completion	Multiple	Heterogeneous	Rig Scheduling	MILP	Minimize	Number of Rig

After this literature review, it is clear that there are some tendencies in papers about the Rig Scheduling Problems. The first publications were in the 60's and 70's and refer to drilling rigs scheduling problem in an oil field development perspective. However, only in the second half of the 90's, the discussions started to gain the deserved attention. Some new mathematical models were proposed, but they were still unpractical for real instances, leaving the use of heuristics as an alternative approach for exact algorithms.

From 2000 to 2011 there was a boom in the researches, proposing new and more efficient heuristics for the issue and most of them focusing on workover rigs. The improvement in the heuristics performance allowed to approach more realistic scenarios, such as heterogeneous fleet of rigs and multiple types of activities to be planned. Beyond 2012, the publications started to propose matheuristics (hybrid methods combining mathematical programming and metaheuristics).

Chart 2 resumes some statistics that can be taken from Table 1. The vast majority of the papers found in this review approached problems of Workover Rigs. Further, most publication are for onshore oil field planning. There is also a balance between models or heuristics for "Routing + Scheduling" and "Scheduling" oil rigs, even though there are more works focusing only in scheduling. Last, most researches propose heuristics to solve their problems. Aiming to fill some of the literature's gap, we propose a mathematical model and a heuristic to the Drilling and Completion Rig Scheduling Problem of a homogeneous fleet of offshore oil rigs. Furthermore, we consider a realistic objective function that minimizes the schedule's budget, taking into account multiple factors. In the next section, we describe the problem tackled in our research with more details.

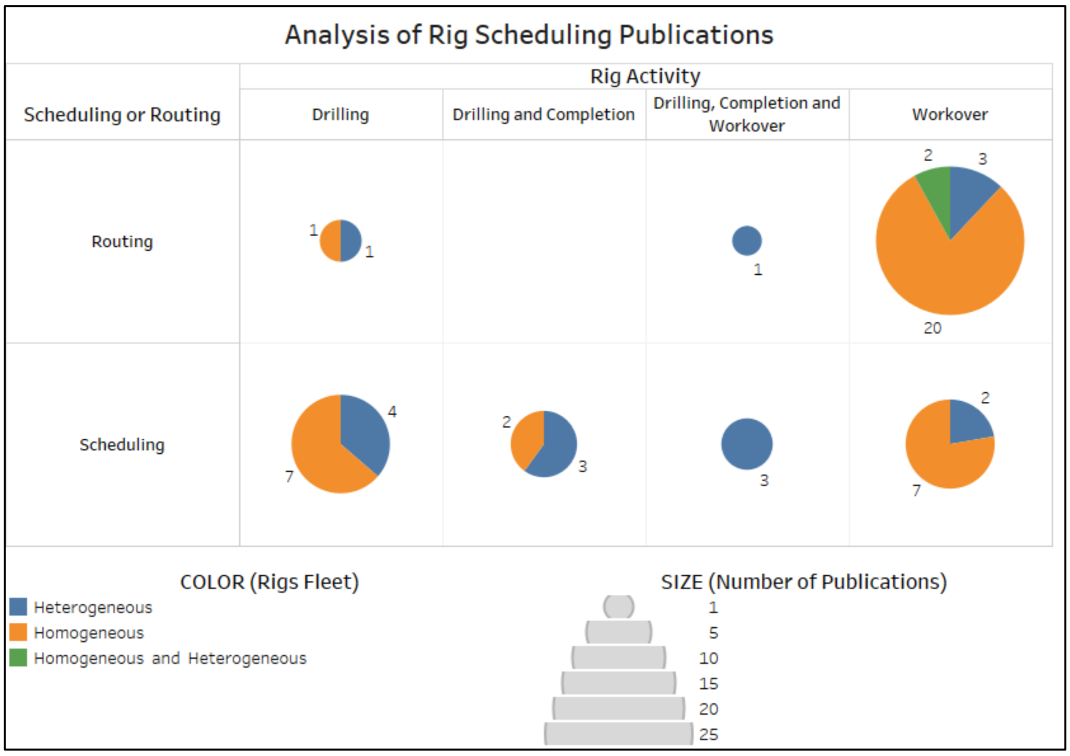

Chart 2. Analysis of rig scheduling publications (Source: the author).

4
Problem Statement

In Section 3, we have presented a literature review of the rig scheduling problem, its importance and its needs. This Section presents the company studied and defines the problem in its perspective.

Offshore companies perform several complex, risky and expensive operations in the Exploration and Production (E&P) of Oil & Gas. The company of this case study, a major multinational of the energy sector, is immersed in such environment full of uncertainties and high investments. In order to perform several E&P activities, the company needs to contract a large number of oil rigs, being one of the major players in the contract drilling rig market (Kaiser & Snyder, 2013). A daily rig rate vary between US$ 50,000 to US$ 700,000, depending on the rigs type, market and its operational specifications ((Kaiser & Snyder, 2013; Osmudsen *et al.*, 2010; IHS Markit, 2017).

Consequently, rigs are a scarce and costly resource contracted by the studied company that requires to be properly scheduled. An underestimated rigs fleet can result in oil production delays and opportunities losses that affect the wells profitability. On the other hand, an oversized fleet may lead to idleness costs of over 1 billion *Reais* (R$), about three hundred million dollars, as mentioned by Pamploma (2016).

Additionally, a vast number of its operations are performed in deep and ultra-deep water. Most of those operations are performed by contracted offshore rigs and they are key activities for the wells development. Such operations can be separated in the four groups:

- Drilling: After the appraisal, a well is drilled using an oil rig that rotates a drill bit. Follow, a steel pipe is placed inside the drilled hole and secured with cement, providing structural integrity and isolating the high pressures zones from the surface and from each other.

- Evaluation: Before completing a well, formation evaluation and logging tests are performed to establish the size and value of the Oil & Gas reserves in order to find out if the reservoir is economically feasible to develop.
- Completion (well): After drilling the well and before beginning its production, the well requires to be "completed". There are two types of completions: cased-hole and open hole. In the first type, perforations are made in the casing that pass through the reservoir, proving flow passages for the oil to pass to the well production. The second consists of just running the casing into the reservoir and leaving the end of the pipe open.
- Workover (well): A completed well requires regular maintenance to correct, maintain and improve productivity. These remedial operations can be a replacement of the tubing, a cleanup or new completions, new perforations and various other maintenance works such as the installation of gas lift mandrels, new packing, *etc*.

The first three groups of activities are part of the wells development phase. Therefore, the studied company plans these tasks before the others in a medium term horizon. On the other hand, the workover operations occur in the production phase of a well, after its construction. These activities, usually executed by workover rigs, are harder to be predicted in advance, as they depend on the outcome of the first schedule. Consequently, they are scheduled separately, later than the others.

Due to this planning structure, the authors decided to consider scenarios with drilling, evaluation and completion activities only. However, it is important to remark that the proposed solution method could be easily applied to scenarios with workover activities, possibly without any change in its rules, and that both approaches are rare in the literature, as seen in Section 3. Nonetheless, even when focusing exclusively in the wells development phase, scheduling rigs is still hard due to the quantity, value, complexity (multiple precedence rules and time windows to consider for each activity) and variety of tasks and uncertainties – related to geological concepts (structure, reservoir seal and hydrocarbon charge), economic evaluations (costs, probability of finding and producing economically viable reservoirs, technology and oil price) and development and production (infrastructure, production schedule, quality of oil and operational costs and reservoir characteristic) (Suslick *et al.*, 2009). Because of this situation, many Oil & Gas companies fail to meet the delivery, budget and performance targets (Reid *et al.*, 2016). Obviously, the use of decision support systems (DSS) is a crucial ingredient to obtain competitive advantages, especially for big companies and for the rig scheduling problem.

Undoubtedly, there has been a vast number of researches aiming to help energy enterprises in their decision making process (Tarhan *et al.*, 2009) and many of these studies were focusing in the rig scheduling and routing problems, as presented in Section 3 of this dissertation. However, most of these papers focus in simplified problems with onshore wells and single activities types, solving it either with exact algorithms or with heuristics. Just a few papers tackle problems that are more complex (offshore wells, heterogeneous rigs or multiple types of activities) and those that do it, hardly use realistic objective functions with a budgetary perspective, usually reducing the cost by just minimizing the rigs fleet size (Bissoli, 2014).

As most multinationals, the studied company has unique budgets formulas and contract rules. The studied company's rig scheduling is not an exception, rigs rates might depend on the time – usually trending to lower in the future (as the company gains more time to negotiate prices) – and according to the rig's characteristics (type, water depth, special equipment's used, *etc.*) and rigs contracts can have minimal durations as well as availability windows.

For these reasons, the company needs a tool to solve the rig scheduling problem, considering theirs special constraints and objective function, to be used in a medium term planning horizon and capable of finding strong solutions in an appropriate time. To tackle this problem, aiming not only to fulfill the company needs but also to fill the literature gaps, the authors propose the use of matheuristics, hybrid algorithms that combine exact methods and heuristics (Raidl & Puchinger, 2008), presented later in Section 5. Next, the company's case study is presented in Section 4.1, which describes the problem and its assumptions. Subsequently, Section 4.2 defines the problem with a mathematical model. Last, Section 4.3 presents the instances used in the problem.

4.1. Assumptions

The purpose of the rig scheduling problem is to schedule a set of tasks to a fleet of rigs with the goal to minimize the company's budget, respecting some precedence rules and strict release and due dates. In this study, each task represent a set of rigs activities that need to be performed in a well of the Brazilian offshore basin. Each rig activity belongs to a group (drilling, evaluation or completion). In addition, each task has a fixed duration (identical for all rigs and tasks sequences) and a time window (with strict release and due dates), determined by the company's production plan, and is associated to an offshore well and a project. A project can have more than one tasks scheduled simultaneously. On the other hand, it is forbidden that more than one tasks are assigned to the same well at a given instant of time. In the same way, there cannot be simultaneously tasks in a rig.

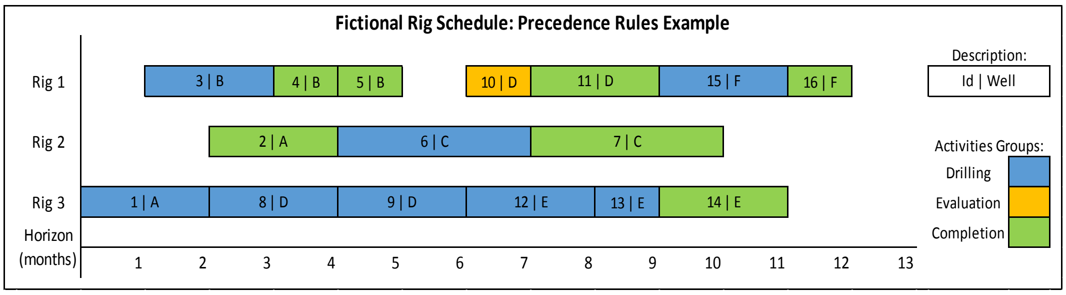

Figure 4. Fictional Rig Schedule for precedence rules illustration (Source: the author).

The precedence rules are subjected to the wells' precedence, *i.e.*, the precedence's relationships between tasks related to the same well, regardless of the rig they are originally allocated in the planning schedule. Considering the rig schedule example of Figure 4 as the original planning of the company, task #1 and task #2 are both associated with the well #1 and, as a result, task #1 must precedes task #2, even though they are in different rigs. Applying these precedence rules for the others tasks would lead to the successor's lists in Table 2.

Table 2. Successor's Lists for the Fictional Rig Schedule in Figure 4 (Source: the author).

Task Id	Successors List
1	2
2	-
3	4 and 5
4	5
5	-
6	7
7	-
8	9, 10 and 11
9	10 and 11
10	11
11	-
12	13 and 14
13	14
14	-
15	16
16	-

Furthermore, the rigs have identical characteristics, any rig can perform any task and an activity has the same duration for all rigs. However, a rig can only have one task assigned per day.

Another particularity of this case study is related to studied company's budget, the objective function represents the rigs' costs. When hiring an offshore rig, the studied company has to pay three taxes to the charterer. The first is a contract charge that is paid in the first year of the contract, usually counted from the start date of the first task. The others two elements represent the hire payed daily for the rig availability and use. This daily rate value depends if an oil rig is used or not in a particular day. In one hand, the hire just for the rig's availability is called as "daily idleness rate". On the other hand, when a rig is rented and used, the hire is called as "daily use rate". The rigs contract is considered to start at the first day of the first task assigned to the contracted rig. The contract has a duration length of at least 2 years, or 730 days, and considers any idleness day after rigs start. As a result, the rig scheduler operator can choose to contract a new rig to the fleet or even to withdrawn (remove) an existent rig of the fleet. Assign a new rig to the fleet allows more availability time for the tasks, but might increase costs. On the contrary, removing a rig of the fleet removes flexibility, but can reduce costs as any rig has a contract cost associated with it and must serve the fleet for at least 2 years, counted from the beginning of the first task assigned to it and considering any idleness time after the start. Figure 5 illustrates an example of a rig schedule budget.

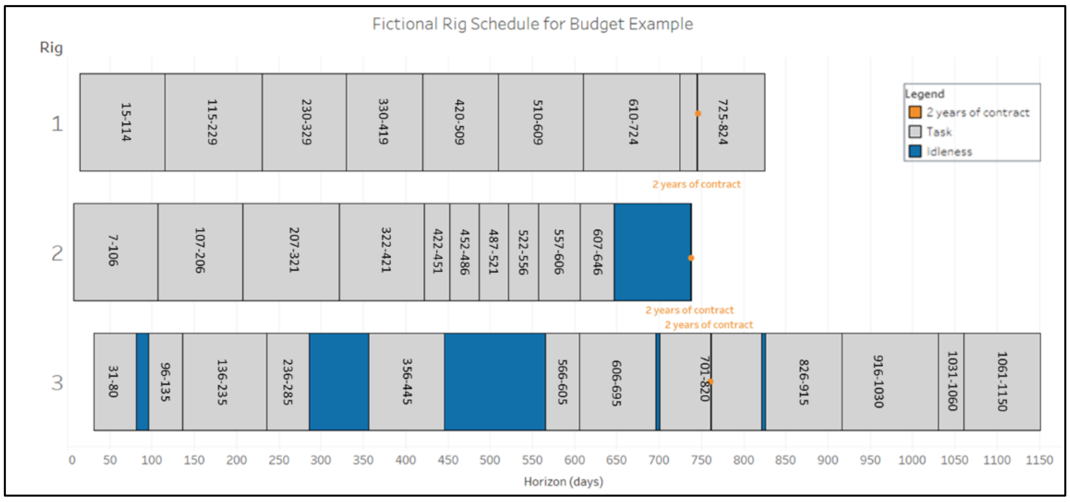

Figure 5. Fictional Rig Schedule for budget formula illustration (Source: the author).

In the schedule example of Figure 5, there are three rigs being contracted for a three years planning horizon. The first rig is programed to be used from day 15 to day 824 without any idleness time between tasks, which results in a contract of 810 days. The second rig is set to operate from day 7 to day 646, resulting in 640 operational days. However, the rigs' contracts must have at least two years (730 days), which leads to an idleness time for the rented rig at the end of its contract. The third rig has several tasks scheduled between day 31 and day 1150. Due to the tasks strict time windows constraint and the precedence rules, the third rig ended up needing a contract of 1120 days with 215 idleness days and 905 activity days.

Most scheduling problems can be represented according to Graham notation, proposed by Graham *et al.* (1979), that describes such problems using a three-field classification $\propto | \beta | \gamma$, where the field $\propto$ represents the machine environment, the β field describes the processing characteristics and its constraints and the field γ expresses the objective function to be optimized. Through the Graham notation for scheduling problems and the framework proposed by Pinedo (2008), this rig scheduling problem can be represented as a $P_m \,|\, \bar{r}_j \bar{d}_j prec \,|\, nonreg$ scheduling problem. This is, the scheduling of identical machines in parallel with strict release and due dates, as well as precedence rules, whose objective is to minimize the budget, a non-regular objective function equal to the weighted sum of idleness time, number of machines and activities durations. By non-regular objective function, we refer to a schedule in which the solution can be improved by delaying some task towards the end, *i.e.*, it is possible to obtain better solutions by deteriorating a part of current solution. According to Dell'Amico *et al.* (2008), the scheduling of parallel machines is to program a set $\{1, 2, .. n\}$ of n jobs, each having an associated processing time p_j (where $j = 1, 2..n$) and a set $\{1, 2..m\}$ parallel identical machines that can process at most one job at a time. In the studied problem, the machines represent the oil rigs contracted by the studied company and the jobs are the activities or tasks performed by these offshore rigs. Each job (task or activity) has a specific duration in days (a processing time p_j), a strict time window to be allocated, respecting its strict release date ($\bar{r}_j$) and strict due dates ($\bar{d}_j$).

In addition, each task is associated with a well and a project and must respect some precedence constraints in chains, where a task can have one or more successor and one or more predecessor. These precedence rules are calculated according to the projects and the wells as explained in Figure 4 and Table 2. Last, following Carilho & Villas Boas (2016) approach, some tasks are planned together according to their wells, project and time windows, forming blocks of tasks that must be performed in sequence without preemption or breaks. As a result, it is possible to reduce the size of the problem to a set $\{1, 2, .. n'\}$ of n' jobs, each one representing a block (group) of tasks, which are used as input data to the mathematical models, grouping tasks with similar characteristics and reducing the model's computational complexity. Following these assumptions, the next section proposes the mathematical models procedures for the studied problem.

4.2. Mathematical models

As seen in the last section, the company's rig scheduling problem can be understood as a scheduling problem of parallel identical machines with precedence rules, strict release and due dates aiming to minimize a non-regular objective function representing the company's budget. The literature review from Section 3.1 has shown a vast number of publication tackling similar problems. For this study, we compared our problem with others scheduling formulations found in the literature from disjunctive programming to integer and binary programming models. The best-fitted formulation was through binary integer linear programming based on models from Peréz *et al.* (2016), Marques *et al.* (2014) and Carilho &

Villas Boas (2016). The first presented an efficient binary mathematical model. The second has similar constraints and characteristics. The third used a block structure that reduces the problem's complexity and has similar characteristics.

The proposed formulation is divided in two mathematical models with different object function and variables that share common solutions' space search. The first model minimizes the number of oil rigs disregarding its costs. This solution determines the minimal number of rigs needed for scheduling the entire set of tasks and serves as an initial upper bound for subsequent procedures. Then, a mathematical model for rigs budget minimization is looped using the minimum number of rigs found in the earlier model as an initial solution and appending a new rig into the fleet at each iteration. The loop procedure stops when the rigs costs rises after adding a rig. This approach was used to avoid non-linear and disjunctive programming approaches that would require high computational effort. Figure 6 illustrates a framework of these procedures and Sections 4.2.1 and 4.2.2 detail these two mathematical models used for minimization of the rig scheduling's budget.

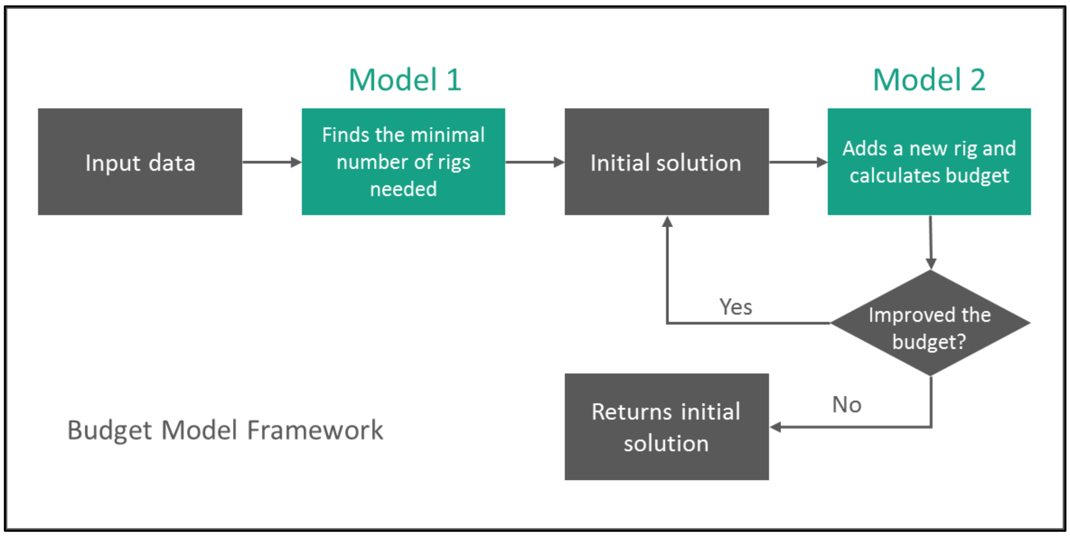

Figure 6. Procedure of the budget minimization mathematical (Source: the author).

It is important to notice that the Model 2 can be used without any input delimitating the size of the set of rigs to *m* equal to the number of wells tasks *n*. However, this approach has a great chance of impacting severally in the computational effort required by the model in real life instances. The mathematical procedure described in Figure 6 aims to provide a solution method alternative with lower computational effort.

4.2.1.
Model 1 – Minimum rigs fleet size

This Section presents the first model used to find an initial viable solution for the studied problem with the minimum rigs fleet size needed to schedule the set of tasks, describing its sets, parameters, variables, objective function and constraints according to the machine scheduling notations.

Sets:

Table 3. Sets for Model 1 (Source: the author).

Set	Index	Name	Description
J	j, j'	Tasks	A set $\{1, \dots, n\}$ of wells tasks.
M	i	Rigs	A set $\{1, \dots, m\}$ representing the rigs available for hiring, where $m = \max\limits_{t\geq 0} \sum_{j\mid\overline{r_j}\leq t\leq \overline{d_j}} 1$.
H	t, t'	Time	A set of time periods in the planning horizon $\{1, \dots, h\}$.

Parameters:

Table 4. Parameters for Model 1 (Source: the author).

Parameter	Name	Description
p_j	Processing time	Processing time of task j in any rig.
$\overline{r_j}$	Strict release date	Minimum period when task j can start.
$\overline{d_j}$	Strict due date	Maximum period when task j can finish.
$prec_{j,j'}$	Precedence Relationship	Binary parameter that defines the precedence between tasks j and j'. Equal to 1 if task j precedes j' (j' can only start after j' finish) and 0, otherwise.

Variables:

Table 5. Variables for Model 1 (Source: the author).

Variables	Name	Description	Type
$TaskStart_j$	Task start	Variable that measures the period in which a task j starts.	*Integer*
$TaskAllo_{i,j,t}$	Task allocation	Binary variable indicating the start of an task j in rig i at period t. Equal to 1 if true and 0, otherwise.	*Binary*
$RigUse_i$	Rig usage	Binary variable that indicates if a rig i is used or not. Equal to 1 if true and 0, otherwise.	*Binary*

The number of rigs available for hiring in this model is determined by largest intersection of tasks availability windows, which is the maximum possible number of rigs need for scheduling all tasks. Two variables are used to define when a task is performed. The first (task start, $TaskStart_j$) stores the start of the task j and belongs to the non-negative integer domain. The second (task allocation, $TaskAllo_{i,j,t}$) relates the period t when task j starts and the rig i in which it is allocated, this variable is binary and is created only and only if period t respects the activity time window interval $[\overline{r_j}, \overline{d_j}]$. The last variable (rig usage, $RigUse_i$) is a binary value that indicates if a rig available to be contracted is used or not. Next, the objective functions and the constraints used at Model 1 are presented.

Objective Functions:

$$Min \sum_{i \in M} RigUse_i \tag{1}$$

***Subject to**:*

$$TaskStart_j = \sum_{i \in M} \sum_{t \in H} TaskAllo_{i,j,t} \cdot t \qquad \forall j \in J \tag{2}$$

$$\sum_{i \in M} \sum_{t \in H} TaskAllo_{i,j,t} = 1 \qquad \forall j \in J \tag{3}$$

$$\sum_{i \in M} \sum_{t' \in H | t - p_j + 1 \leq t' \leq t} TaskAllo_{i,j,t'} \leq 1 \qquad \forall j \in J, t \in H \tag{4}$$

$$TaskStart_{j'} - TaskStart_j \geq p_j \qquad \forall j \in J, j' \in J \mid prec_{j,j'} = 1 \tag{5}$$

$$RigUse_i \geq \sum_{t \in H} TaskAllo_{i,j,t} \qquad \forall i \in M, \forall j \in J \tag{6}$$

$$RigUse_i \leq RigUse_{i-1} \qquad \forall i \in M | i > 1, |M| > 1 \tag{7}$$

$$TaskAllo_{i,j,t} \in \{0,1\} \qquad \forall i \in M, j \in J, t \in H \; | \overline{r_j} \leq t \leq \overline{d_j} - p_j + 1 \tag{8}$$

$$TaskStart_j \in \mathbb{Z}_{\geq 0} \qquad \forall j \in J \tag{9}$$

$$RigUse_i \in \{0,1\} \qquad \forall i \in M, j \in J \tag{10}$$

The objective function for rigs minimization is shown in equation (1), which minimize the number of rigs used and, consequently, reduces the idleness. The expression from (1) is subject to constraints (2), (3), (4), (5), (6), (7), (8), (9) and (10). Constraint (2) defines the period in which a task starts and connects variables $TaskStart_j$ and $TaskAllo_{i,j,t}$. Constraint (3) forces the model to execute all tasks once and in only one rig. Constraint (4) is related with the task processing time and forces that while a task is being executed in a rig, there will not be any other task scheduled to that same rig. Constraint (5) assures that tasks respect theirs precedence relationships. Constraint (6) determines that a rig hired only if it is used and *vice versa*, connecting variables $RigUse_i$ and $TaskAllo_{i,j,t}$. Constraint (7) is a symmetry breaking constraint that causes the model to generate rigs in an ascending order from one to the total number of rigs used. Last, constraints (8), (9) and (10) define the variables universe domain. In the mathematical procedure described in Figure 6, the solution generated by this model is given as initial solution to Model 2 (warm start), which is described in the next section.

4.2.2.
Model 2 – Minimum rigs budget

As mentioned earlier in Section 4.2, Model 2 is used to find a schedule with minimal budget using a pre-determined fleet of available rigs. This section describes this mathematical model by its sets, parameters, variables, objective function and constraints, following the machines scheduling notations.

Sets:

Table 6. Sets for Model 2 (Source: the author).

Set	Index	Name	Description
J	j, j'	Tasks	A set $\{1, \dots, n\}$ of wells tasks.
M	i	Rigs	A set $\{1, \dots, m\}$ representing the available fleet of rigs, where m can be given by the solution from Model 1 or by largest intersection of tasks time windows (when no initial solution is given).
H	t, t'	Time	A set for the time periods in the planning horizon $\{1, \dots, h\}$

Parameters:

Table 7. Parameters for Model 2 (Source: the author).

Parameter	Name	Description
p_j	Processing time	Processing time of task j in any rig.
$\overline{r_j}$	Strict release date	Minimum period when task j can start.
$\overline{d_j}$	Strict due date	Maximum period when task j can finish.
$prec_{j,j'}$	Precedence Relationship	Binary parameter that defines the precedence between tasks j and j'. Equal to 1 if task j precedes j' (j' can only start after j' finish) and 0, otherwise.
$cost^{idle}$	Idleness rate	Cost value charged for each period of rig idleness.
$cost^{oper}$	Operational rate	Cost value charged for each period of rig operation, rig utilization.
$cost^{hire}$	Hiring cost	Cost value charged for hiring a rig for any extend of time.

Parameter	Name	Description
rig^{min}	Contract minimum length	Contract minimum length required to hire a rig (2 years or 730 days).

Variables:

Table 8. Variables for Model 2 (Source: the author).

Variables	Name	Description	Type
$TaskStart_j$	Task start	Variable that measures the period in which an activity j starts.	*Integer*
$TaskAllo_{i,j,t}$	Task allocation	Binary variable indicating the start of an task j in rig i at period t. Equal to 1 if true and 0, otherwise.	*Binary*
$RigUse_i$	Rig usage	Binary variable that indicates if a rig i is used or not. Equal to 1 if true and 0, otherwise.	*Binary*
$RigAllo_{i,t}$	Rig start	Binary variable indicating if rig i starts at period t. Equal to 1 if true and 0, otherwise.	*Binary*
$RigEnd_i$	Rig end	Variable that measures the period in which a rig i operation ends.	*Integer*

Besides variables $TaskStart_j$, $TaskAllo_{i,j,t}$ and $RigUse_i$, the second model for rig scheduling budget minimization uses two others variables: $RigAllo_{i,t}$ and $RigEnd_i$, both used to store rigs information. The first (rig start, $RigAllo_{i,t}$) is a binary variable that marks the period t when rig i starts. The second stores the end of the rigs contract j and belongs to the non-negative integer domain. The other variables ($TaskStart_j$, $TaskAllo_{i,j,t}$ and $RigUse_i$) are the same of Model 1. $TaskStart_j$ stores the start of the task j and belongs to the non-negative integer domain. $TaskAllo_{i,j,t}$ relates the period t when task j starts and the rig i in which it is allocated, this variable is binary and is created only if period t respects the task time window interval $[\overline{r_j}, \overline{d_j}]$. Last, $RigUse_i$ contains a binary variable that indicates if a rig available for contracting is used or not.

Rigs Budget Objective Function

$$Min \quad \begin{aligned} & cost^{hire} \cdot \sum_{i\in M} RigUse_i + \\ & cost^{oper} \cdot \sum_{i\in M}\sum_{j\in J}\sum_{t\in H} TaskAllo_{i,j,t} \cdot p_j + \\ & cost^{idle} \cdot \sum_{i\in M}\left[RigEnd_i - \sum_{t\in H}\left(RigAllo_{i,t} \cdot t - \sum_{j\in J} TaskAllo_{i,j,t} \cdot p_j\right) + RigUse_i\right] \end{aligned} \tag{11a}$$

As mentioned in Section 4.1, the rigs budget can be divided in three segments and, consequently, the objective function from Model 2, described by equation (11a), is the sum of three costs. The first term refers to the sum of the rigs hiring costs (a fixed charged payed when contracting an oil rig, $cost^{hire}$). The other two terms represent variable costs subjected to the rigs utilization and the length of the contract. One term calculates the total operational costs, the sum of a rate charged for each day of utilization of a rig ($cost^{oper}$). The other is accountable for idleness costs, the sum of the rigs idleness time multiplied by a rate charged for having a rig availability contract ($cost^{idle}$).

However, as all tasks must be schedule and the rigs fleet is considered to be homogeneous, the sum of the tasks processing times will be fixed for any solution of a particular instance. Consequently, the equation (11a) can be simplified to just the sum of the idleness rate and the rig hiring cost, disregarding the daily use rate that relies only on the duration of the tasks and does not depend on the number of rigs available. Follow, we present the resulting objective function and its constraints:

$$Min \quad \begin{matrix} cost^{hire} \cdot \sum_{i \in M} RigUse_i + \\ cost^{idle} \cdot \sum_{i \in M} \left[RigEnd_i - \sum_{t \in H} \left(RigAllo_{i,t} \cdot t - \sum_{j \in J} TaskAllo_{i,j,t} \cdot p_j \right) + RigUse_i \right] \end{matrix} \qquad (11b)$$

Subject to:

$$TaskStart_j = \sum_{i \in M} \sum_{t \in H} TaskAllo_{i,j,t} \cdot t \qquad \forall j \in J \qquad (2)$$

$$\sum_{i \in M} \sum_{t \in H} TaskAllo_{i,j,t} = 1 \qquad \forall j \in J \qquad (3)$$

$$\sum_{i \in M} \sum_{t' \in H | t - p_j + 1 \le t' \le t} TaskAllo_{i,j,t'} \le 1 \qquad \forall j \in J, t \in H \qquad (4)$$

$$TaskStart_{j'} - TaskStart_j \ge p_j \qquad \forall j \in J, j' \in J \, | Seq_{j,j'} = 1 \qquad (5)$$

$$RigUse_i \ge \sum_{t \in H} TaskAllo_{i,j,t} \qquad \forall i \in M, \forall j \in J \qquad (6)$$

$$RigUse_i \le RigUse_{i-1} \qquad \forall i \in M | i > 1, |M| > 1 \qquad (7)$$

$$RigAllo_{i,t} \le 1 - \sum_{t' \in H | t' < t} TaskAllo_{i,j,t'} \qquad \forall i \in M, \forall j \in J, \forall t \in H \qquad (12)$$

$$RigEnd_i \geq \sum_{t \in H} TaskAllo_{i,j,t}(t + p_j - 1) \qquad \forall i \in M, \forall j \in J \qquad (13)$$

$$RigEnd_i + RigUse_i \geq \sum_{t \in H} \left(RigAllo_{i,t} \cdot t + \sum_{j \in J} p_j \cdot TaskAllo_{i,j,t} \right) \qquad \forall i \in M \qquad (14)$$

$$RigEnd_i - \sum_{t \in H} RigAllo_{i,t} \cdot t + 1 \geq rig^{min} \cdot RigUse_i \qquad \forall i \in M \qquad (15)$$

$$\sum_{t \in H} RigAllo_{i,t} = RigUse_i \qquad \forall i \in M \qquad (16)$$

$$TaskAllo_{i,j,t} \in \{0,1\} \qquad \forall i \in M, j \in J, t \in H \,|\overline{r_j} \leq t \leq \overline{d_j} - p_j + 1 \qquad (8)$$

$$TaskStart_j \in \mathbb{Z}_{\geq 0} \qquad \forall j \in J \qquad (9)$$

$$RigUse_i \in \{0,1\} \qquad \forall i \in M, j \in J \qquad (10)$$

$$RigAllo_{i,t} \in \{0,1\} \qquad \forall j \in J, t \in H \qquad (17)$$

$$RigEnd_i \in \mathbb{Z}_{\geq 0} \qquad \forall i \in M \qquad (18)$$

The objective functions (11a) and (11b) are subject to sixteen constraints: (2-10) and (12-18). Constraints (2-10) are the same of Model 1. The new constraints in the model are applied for rigs budget calculation. Constraint (12) calculates the period t in which a rig i starts its first task assigned, defined by variable $RigAllo_{i,t}$. Constraints (13), (14) and (15) are related with the end of rig contract. Constraint (13) imposes that rig i contract end date must be later than any task execution finish allocated in that rig. Constraint (14) assures that the time window between the rigs i contract start and finish is long enough to perform all the tasks assigned to the rig. Constraint (15) forces that hired rig i contract length is at least equal to the minimal hiring duration. Constraint (16) obligates that a rig task can only begin in one period and if there is a contract start, the rig must be hired. Last, constraints (17) and (18) are for the definition of variables domain.

To summarize, these constraints assure the solution from Model 2 to respect the assumptions of the studied rig scheduling described in Section 4.1, and allow to calculate many of the variables used in the budget objective function. In theory, it is possible to execute just the Model 2 in one run without executing Model 1. However, the constraints and variables need to calculate the rigs budget are complex and tend to increase the computational effort. Therefore, a mathematical programming procedure described earlier in Figure 6 was developed to tackle this problem. In this procedure, the Model 2 uses the initial solution provided by the

Model 1 as a warm start and finds a rig schedule with minimum budget through a loop procedure presented previously in Figure 6. Nonetheless, this process can still result in a large computational effort in large instances. Another alternative approach is the use of heuristics such as local search algorithms or matheuristics, when combining exact and approximation techniques. In the next section, we present and describe a Local Search heuristic that can be used to efficiently improve initial solutions such as those provided by Model 1 in a matheuristic approach and a constructive algorithm that perhaps can be used as initial solution generator.

5
Local Search Algorithm

As mentioned earlier, the studied rig scheduling problem has several constraints and parameters that rise its complexity and size and turn the problem hard to solve with exact algorithms. A procedure using two mathematical models was shown in Section 4.2. However, the model that minimizes the budget (Model 2) requires great computational effort, even when using relaxed assumptions (such as time horizon in weeks, tasks in block structure and homogeneous fleet of rigs).

Heuristic techniques become an alternative, since they are able to generate solutions close to the optimal with realistic assumptions. It is also possible to combine heuristics with mathematical programming algorithms, generating the so-called “matheuristic”. Matheuristics can be classified in two types: collaborative combinations of exact algorithms and heuristics, and; integrative combinations. The first group refers to methods where the algorithms exchange information, but are not part of each other, being executed sequentially, intertwined or in parallel. The second group represents combinations in a master-slave relationship, where one algorithm is a subordinate embedded component of the other (Raidl & Puchinger, 2008).

A possible matheuristic approach consists in using local search heuristics, which are iterative algorithms that search for better solutions moving from one solution S to another solution S' according to some neighborhood structures. Aiming to provide an efficient method for the issue, this section proposes a local search heuristic to be used within a matheuristic procedure of collaborative combination with sequential execution.

According to Talbi (2009), local search algorithm is one of the oldest and simplest metaheuristic methods. In short, this heuristic receives an initial solution S_0 – in our case, an initial solution from a mathematical model – and improves it through several small modifications. At each iteration, the local search tries to find possible improvements in the current solution exploring similar solutions (neighbor solutions) through small movements, selecting the neighbor solution according to a strategy. When a stop criterion (no improvements, time or number of iterations) is reached, the algorithm stops and returns the best solution found. Algorithm 1 illustrates the steps of the basic local search used in this problem.

Algorithm 1. Basic Local Search Heuristic

```
1:  $s_0 \leftarrow InitialSolution('Mathematical\ Model')$;
2:  $s, s^* \leftarrow s_0$;
3:  $strategy \leftarrow DefineNeighborSelectionStrategy()$;
4:  repeat:
5:      $s^* \leftarrow s$;
6:      $N(s) \leftarrow GenerateNeighborhood(s)$;
7:      $s \leftarrow SelectNeighbor(N(s), strategy)$;
8:  until $StopCriterion(s, s^*)$;
9:  return $s^*$;
```

In other words, the heuristic, in this study, first receives an initial solution from a mathematical model (Step 1) and stores it as the current and best solutions (Step 2). Also, the heuristic needs to define a neighbor selection strategy, *i.e.*, the approach used to select a new current solution between a set of neighbor solutions (Step 3). Then, the algorithm enters in a loop searching for better neighbor solutions (Steps 4 to 8). At each iteration of this loop, the method stores the current solution as the best solution found so far (Step 5), generates a set of neighbor solutions $N(s)$ (Step 6) and selects a solution s in this set $N(s)$ according to a pre-defined *strategy* (Step 7), which will be explained later. A neighborhood represents a set of possible solutions that can be found performing some small pre-defined movements (changes) from the current solution. Finally, if a stopping criterion is reached (Step 8), which can be the lack of better solutions, the execution time or the number of iterations, the heuristic returns the best solution found in the search procedure (Step 9). It is expected that this simple approach allow making fast improvements in a solution with just a few computational effort. Next, further details on the neighborhood movements, neighbor selection strategies and the stop criterion are presented.

5.1. Neighborhood Structures

As explained earlier, the local search algorithm is an iterative method that starts with an initial solution and iteratively explores neighbor solutions searching for better solutions. This neighborhood structure is defined by a set of solutions that can be found after performing some neighborhood movements at the current solution. In essence, these neighborhood movements represent some small systematic changes in a solution that leads it to a different one, yet similar.

As seen in the literature review from Section 3.1, the most common neighborhood structures for rig scheduling and machine scheduling problems apply *insert* and *swap* movements. Basically, the first one removes a task from a position and inserts it in another place, while the other move swaps the position of two different tasks. Obviously, a *swap* movement can be executed with two consecutive *insert* moves. There is also a particular case of *swap* movements where the tasks switched are adjacent in the schedule order, which is classified as an *interchange* move (Dong *et al.*, 2009).

In this rig scheduling problem, a solution is defined not only by the order of the tasks in the rigs (machines) but also by the date in which they are scheduled. Therefore, these classical movements require some modifications. The author tested a series of movements and defined three core neighborhood moves based on the typical *insert* movement:

- *Insert with fixed dates*: this type of movements removes a set of tasks from a position in a rig and inserts them at the end of another existing rig or in a new rig, while keeping the same start time of each task being moved.
- *Insert with dates anticipation*: differently of the previous movement, these movements will try to anticipate a set of task in a rig and then insert these tasks in another rig (an existing one or even a new one) or insert a block of tasks of another rig in the end of this rig whose tasks were anticipated. These approaches are defined in the setup of the movement.
- *Insert with dates postponement*: similar to the second movements, these movements will try to postpone a block of task in a rig and

then insert these postponed tasks in another rig (an existing one or even a new one) or insert a set of tasks of another rig in the end of this rig whose tasks were postponed. These approaches are defined in the setup of the movement.

Next, we specify and illustrate the mechanism of these movements, considering the fictional rig schedule example of Figure 7 as an initial solution for the movements' illustration, with 47 tasks allocated in three rigs.

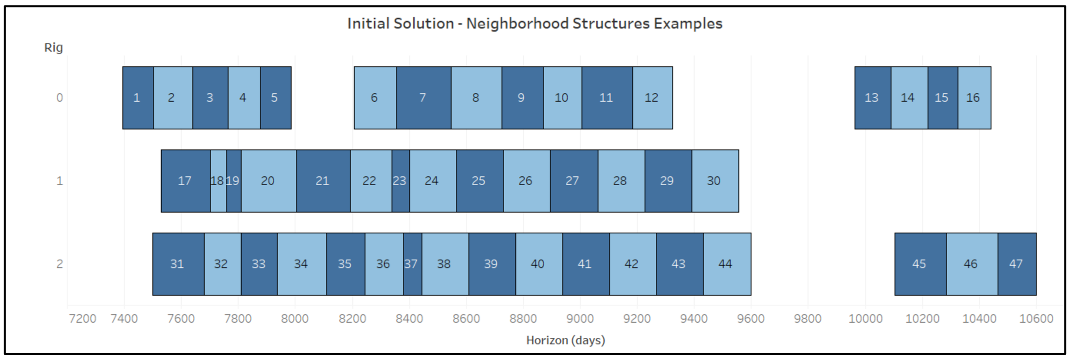

Figure 7. Initial solution for movement examples (Source: the author).

5.1.1. Insert with fixed dates

As mentioned earlier, the *insert with fixed dates* movements remove a set of tasks from a position in a rig and inserts them at the end of another existing rig or in a new rig, adding and removing a rig from the fleet if necessary, eliminating or reducing idleness by increasing or reducing the number of rigs, while maintaining dates of activities without overlapping.. This movement can be inserting tasks *in an existing rig* or *in a new rig*.

The first type, *insert with fixed dates in an existing rig*, shifts activities from the end of a rig to the end of another rig already hired, aiming to minimize the budget without changing the tasks allocation dates or overlapping activities. An example of this movement is illustrated in Figure 8 using the initial solution from Figure 7, in which the heuristic selected tasks #13, #14, #15 and #16f from rig #0, and reallocates them at the end of rig #1. As the previous tasks in rig 1 were closer from the block's start date, the *insert with fixed dates in an existing rig* movement was able to reduce the idleness cost of the schedule without changing the tasks date. It is important to notice that, aside from when the block of tasks moved are accountable for an entire rig schedule, the *insert with fixed dates in an existing rig* movement does not have impact on the hiring costs and its main purpose is to reduce the idleness costs.

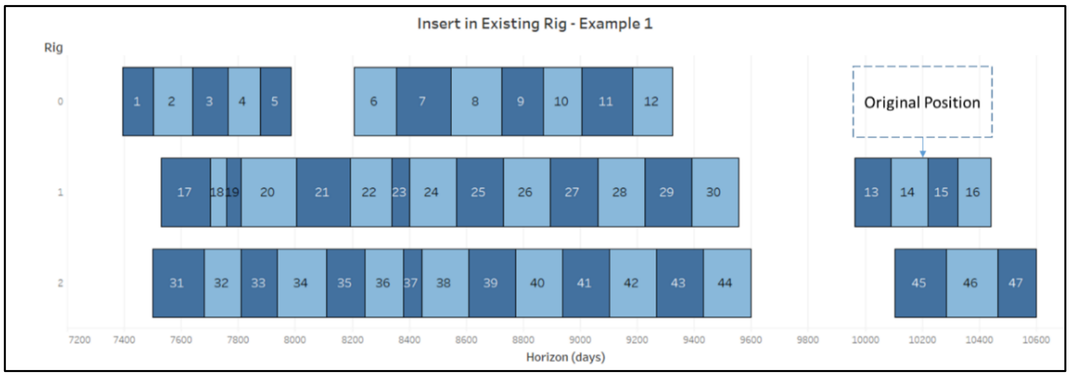

Figure 8. *Insert with fixed dates in an existing rig* move example (Source: the author).

The second type of movement, *insert with fixed dates in a new rig*, reallocate activities from the end of a rig to a new rig, also seeking to minimize the budget and without changing the allocated dates of the activities. On the contrary of the previous move, this movement raises the fleet size, using the trade-off between the idle time cost and the hiring cost. An example of this movement is illustrated in Figure 9.

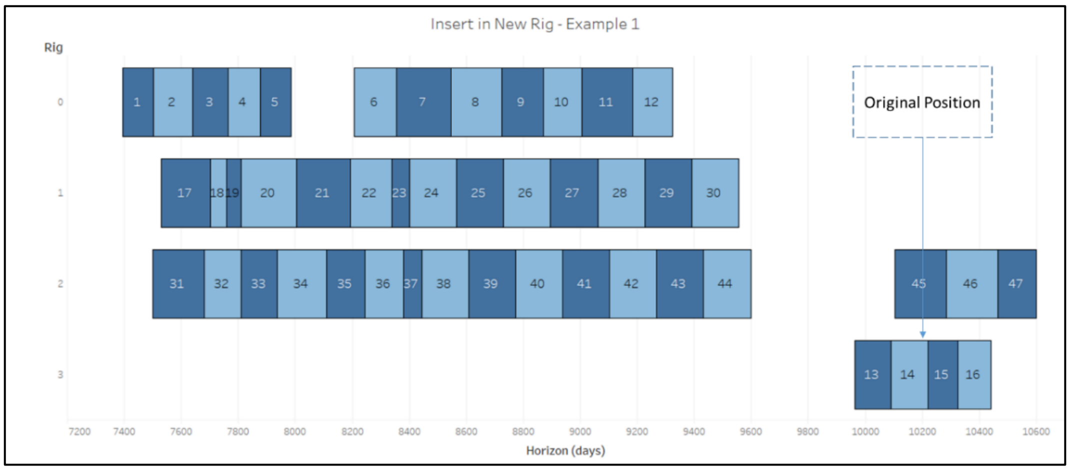

Figure 9. *Insert with fixed dates in a new rig* move example (Source: the author).

In the example of Figure 9, the heuristic also selects tasks #13, #14, #15 and #16 (that were scheduled in the end of rig 0), but reallocates this set of tasks in the new rig 3. Due to long idleness period between the block's start date and the previous tasks at rig 0, the *insert with fixed dates in a new rig* could reduce the idleness cost of the schedule, in spite of the cost of hiring a new rig to the fleet. Different from the Figure 8, where the hiring cost remained constant, in the current example, the addition of a new rig into the fleet resulted in a raise on the hiring cost, but cause of the idle time reduction the movement reduced the rigs budget.

Finally, these two previous *insert with fixed dates* neighborhood structures can also be combined together in a double search for each current solution. This approach will be called by *insert with fixed dates (existing+new rig).*

5.1.2.
Insert with dates anticipation

The *insert with dates anticipation* movements is divided in two successive and optional steps: *anticipation* and *insert*. The first refers to the anticipation of some tasks in a rig, which can be from the start to some position in the rigs allocation list (*anticipation before*) or from this position to the end (*anticipation after*). The second involves the insertion of some tasks from the anticipated rig to the end of another rig, an existing one or new one, (*insert from*) or the insertion of some others tasks from the end of another rig in the end of the anticipated rig (*insert in*). Different from the previous movement (*insert with fixed dates*), this movement allows changing in the anticipation step, which enable new insertions and to find more solutions. Next, we describe some examples of these movements.

The first example is an *insert in with dates anticipation before*, which searches new solutions anticipating blocks of activities from the start of a rig to some position (the *anticipation before* step) and then inserting or not a set of tasks from end of another rig in its end (the *insert in* step). Figure 10 illustrates an example of this movement, where, in the *anticipation before* step, task #17 is anticipated and then, in the *insert in* step, some blocks of tasks from rig 0 (composed by tasks #13, #14, #15 and #16) are inserted in the end of rig 1. It is also important to state that this movement only reduces the fleet of rig, never adding rigs.

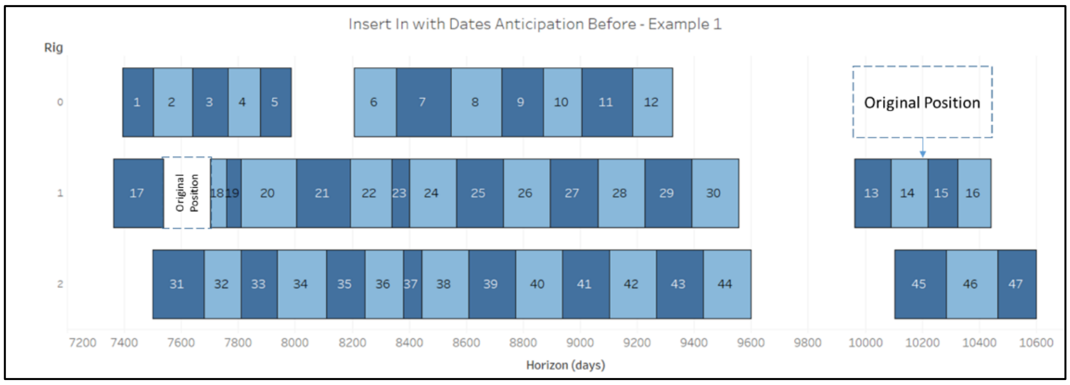

Figure 10. *Insert in with dates anticipation before* move example (Source: the author).

The second example is an *insert from with dates anticipation before*, which after searching new solutions anticipating blocks of activities from the start of a rig to some position (the *anticipation before* step), this movement tries to insert a set of tasks from end of this rig to another existing rig, or even a new rig if necessary (the *insert from* step). Figure 11 presents an example of this movement. In the *anticipation before* step, the first sixteen positions of the rig 0 (tasks #1 to #16) are anticipated and then, in the *insert from* step, some tasks from rig 0 (tasks #13, #12, #13, #14, #15 and #16) are inserted in a new rig 3. This movement allows adding new rigs to the fleet if needed – always analyzing the trade-off between the idleness costs and the hiring costs.

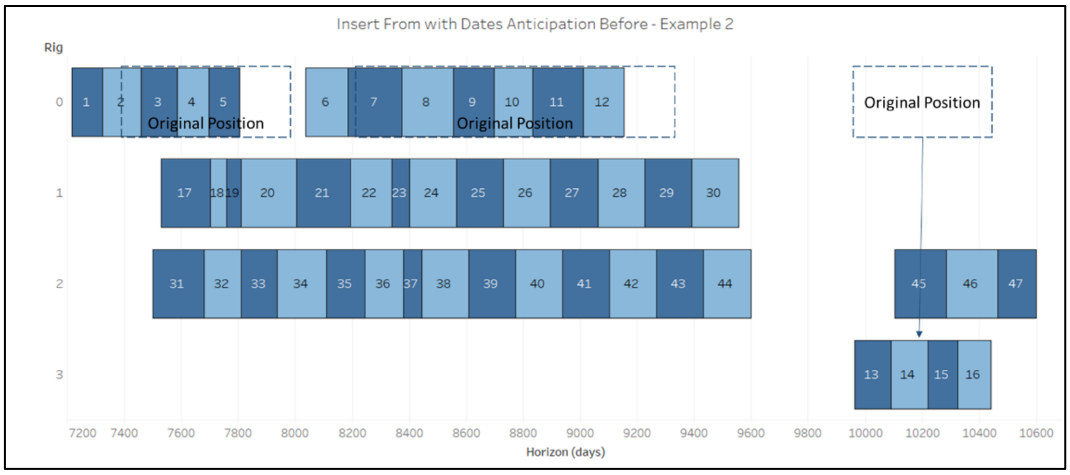

Figure 11. *Insert from with dates anticipation before* move example (Source: the author).

The third example is an *insert in with dates anticipation after*, which searches new solutions anticipating blocks of activities from some position of a position to its end (the *anticipation after* step) and then inserting or not a set of tasks from end of another rig in its end (the *insert in* step). Figure 12 illustrates an example of this movement. In the step *anticipation after*, tasks between the start of the rig 1 and its end (#17 to #30) are anticipated and then, in the *insert in* step, some tasks from rig 0 (tasks #13 to #16) are inserted in the end of rig 1. Just the previous example movement, this movements does not adds new rigs to the fleet and can only reduce the fleet of rigs.

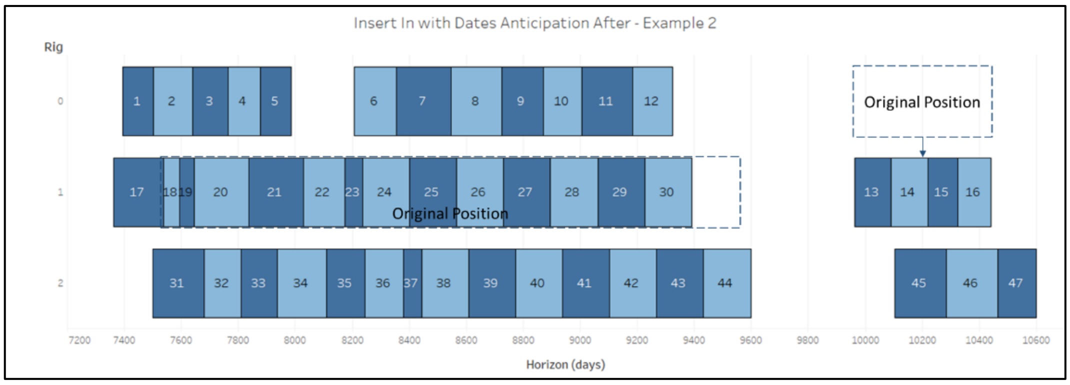

Figure 12. *Insert in with dates anticipation after* movement example (Source: the author).

The *insert from with dates anticipation after* movement has same first step from the previous move (*anticipation after*). But, after searching new solutions anticipating blocks of activities from the i^{th} position of rig to its end (the *anticipation after* step), this movement tries to insert a set of tasks from end of this rig to another existing rig, or even in a new rig if necessary (the *insert from* step). Figure 13 exemplifies this movement. In the *anticipation after* step, the last eleven positions of the rig 0 (tasks #6 to #16) are anticipated and then, in the *insert from* step, some tasks from rig 0 (#13 to #16) are inserted into the end of rig 1. This movement can adds new rigs into the fleet, if necessary.

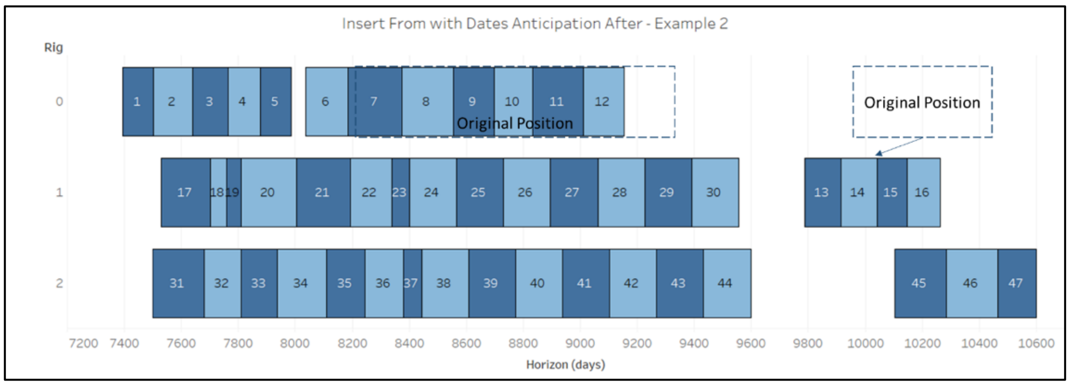

Figure 13. *Insert from with dates anticipation after* move example (Source: the author).

5.1.3. Insert with dates postponement

Last, the *insert with dates postponement* movements is very similar with the previous neighborhood structure type, but instead of anticipating tasks, it delays them with the goal of finding new and better insert movements and it is divided in two successional and optional steps: *postponement* and *insert*. The first refers to the postponement of some tasks in a rig, which can be from the start of a rig to some position (*postponement before*) or from this position to end (*postponement after*). The second involves the insertion of some tasks from the current rig to the end of another rig, an existing one or new one, (*insert from*) or the insertion of some others tasks from the end of another rig in the end of the current rig (*insert in*). Next, we briefly describe some examples considering the initial solution from Figure 7.

The first example is an *insert in with dates postponement before*, which searches new solutions delaying blocks of activities from the start of a rig to some position (the *postponement before* step) and then inserting or not a set of tasks from the end of another rig in its end (the *insert in* step). Figure 14 illustrates an example of this movement. In the *postponement before* step, the first tasks from rig 1 (#17 to #30) are delayed and then, in the *insert in* step, the last tasks from rig 0 (#13 to #16) are inserted in the end of rig 1.

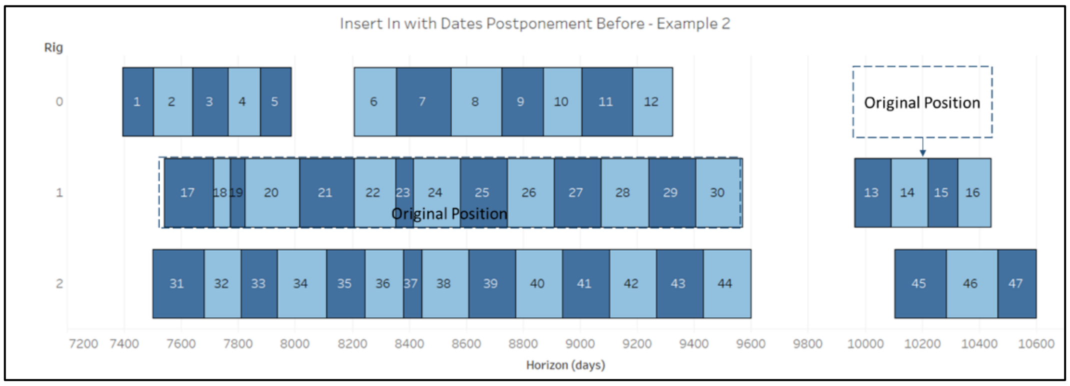

Figure 14. *Insert in with dates postponement before* move example (Source: the author).

The second example is an *insert from with dates postponement before*, which is similar to the *insert in with dates postponement before* movement since both have the same first step (*postponement before*). However, after searching new solutions delaying blocks of activities from the start of a rig to some position (the *postponement before* step), this move tries to insert a set of tasks from the end of this rig to another existing rig, or even a new rig if necessary (the *insert from* step). Figure 15 illustrates this movement. In the *postponement before* step, the algorithm tries to delay the first tasks from rig 0, but task #5 due date does not allow it. Then, in the *insert from* step, some tasks from rig 0 (#13-16) are inserted in a new rig 3.

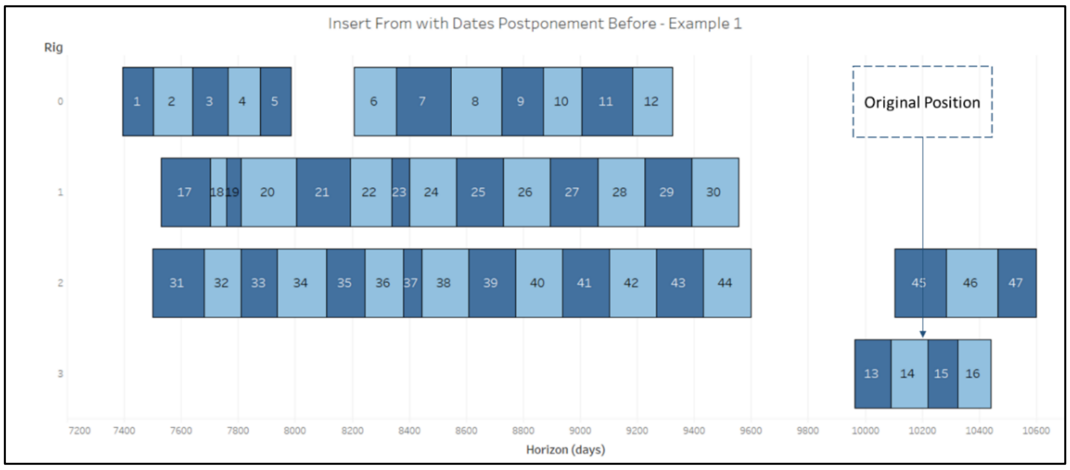

Figure 15. *Insert from with dates postponement before* move example (Source: the author).

The third example is an *insert in with dates postponement after*, which searches new solutions delaying blocks of activities from some position of a rig to its end (the *postponement after* step) and then inserting or not a set of tasks from the end of another rig in its end (the *insert in* step), and is shown in Figure 16. In the step *postponement after*, tasks #17 to #30 are delayed and then, in the *insert in* step, tasks #13 to #16 are inserted in the end of rig 1.

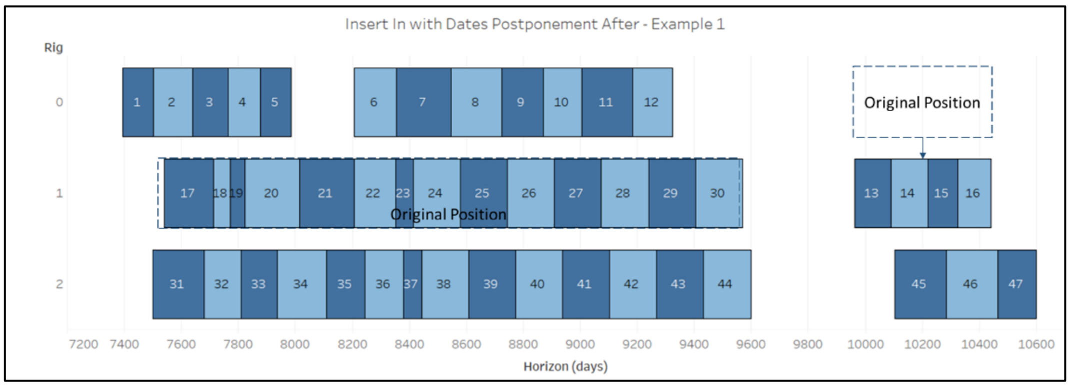

Figure 16. *Insert in with dates postponement after* move example (Source: the author).

Last, the *insert from with dates postponement after* movement has the same first step from the previous move, but, after delaying some tasks of a rig, this move tries to insert a set of tasks from the end of this rig to an existing or new rig (the *insert from* step), as illustrated in Figure 17 example. In the *postponement after* step, the tasks #13 to #16 are slightly delayed and then, in the *insert from* step, these tasks are inserted into the end of a new rig 4.

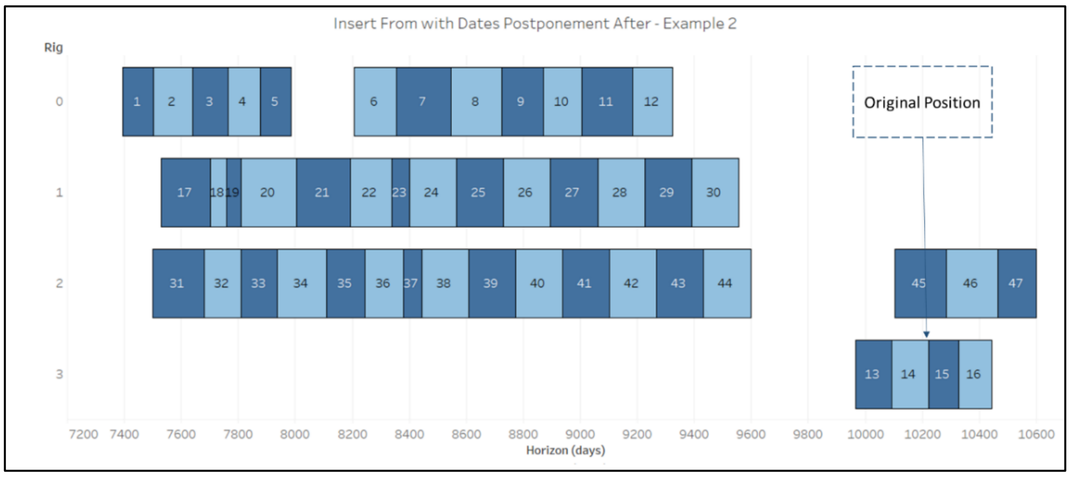

Figure 17. *Insert from with dates postponement after* move example (Source: the author).

In general, both the *insert with dates postponement* move and the *insert with dates anticipation* movement try to change the tasks allocation dates in the rig and, as result, generating new insert neighbors that wouldn't be available in a simple *insert with fixed dates* neighborhood structure.

5.2. Search Strategies

As mentioned earlier, each neighborhood structure can have multiple neighbor solutions that pass in an acceptance criterion (which will be defined later in Section 5.4). Therefore, it is important to define a search strategy to be applied in the selection of a better neighbor. In this local search algorithm, two main strategies, illustrated in Figure 18, are applied:

- *First improvement*: in this strategy, the first improving neighbor that pass in the acceptance criterion is chosen to replace the current solution. Consequently, it is a cyclic exploration search and, only in the worst-case scenarios, all neighbor solutions are analyzed. The left graph of Figure 18 illustrates an example of the first improvement strategy in which the search visits solution 1 and, as it has a better cost; it is selected to be the new current solution. Then, in the second neighborhood, solutions 1.1 and 1.2 are visited and the latter is chosen. Finally, as no improvements are found in the third neighborhood (solutions 1.2.1 and 1.2.2), the current solution is accepted as the best solution found.

- *Best improvement*: this strategy, also known as steepest descent, exhaustively explores the entire neighborhood, and selects (if possible, *i.e.*, if there is any solution that passes the acceptance criterion) the neighbor that improves the most the objective function. In the example of Figure 18, the local search analyzes all the solutions in the first neighborhood (nodes 1, 2 and 3) and selects the second node as current solution. In the next neighborhood, solutions 2.1, 2.2. and 2.3 are visited, the best solution (2.3) is chosen and is selected to be the best solution as there is no improvement in the third neighborhood (2.3.1).

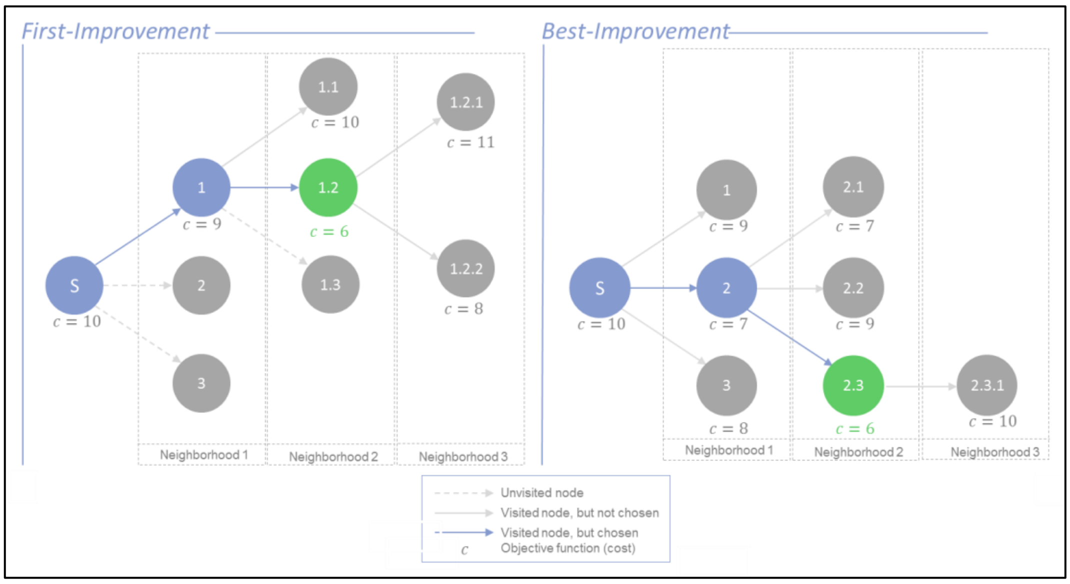

Figure 18. Local search strategies examples (Source: the author).

Usually, the first improvement strategy tends to achieves strong results faster and efficiently, even though the best improvement approach is more likely to find better solutions in a much longer computational time. However, both methods present similar results and it depends on the problem. Aiming to achieve better results, some combinations of these local search strategies were also tested:

- *Best-First*: In this strategy, the local search algorithm changes the neighbor selection strategy at each iteration, starting with the *best* improvement and, then, changing to *first* improvement, iteratively.

- *First-Best*: In this strategy, the local search algorithm changes the neighbor selection strategy at each iteration, starting with the first improvement and, then, changing to *best* improvement, iteratively.
- *Random*: In this strategy, the local search algorithm changes the neighbor selection strategy at each iteration, randomly selecting the search strategy (choosing between *first* or *best* improvement).

5.3. Variable Neighborhood Descent

Although the local search algorithms are easy and simple to be implemented and able to give good solutions very quickly, they will frequently converge toward local optimal solutions, so their qualities strongly depends on the quality of the initial solution (Talbi, 2009). There are several ways to escape from the local search optimums. One particular method, Variable Neighborhood Descent (VND), explores solutions in more than one type of neighborhood, diversifying the range of solutions found (Talbi, 2009). Aiming to improve the local search, a variable neighborhood descent approach, shown in Algorithm 2, was also tested.

Algorithm 2. Variable Neighborhood Descent

```
 1:    $s_0 \leftarrow initialSolution('MathModel')$; //current solution//
 2:    $N_l \leftarrow \{1, \dots, l_{max}\}$; //define a set of neighborhood structures//
 3:    $strategy \leftarrow NeighborSelectionStrategy()$;
 4:    repeat:
 5:        for each $l$ in $N_l$ do:
 6:            $N(s) \leftarrow GenerateNeighborhood(s, l)$;
 7:            $s \leftarrow SelectNeighbor(N(s), strategy)$;
 8:        endfor;
 9:    until $stopCriterion(s)$;
10:    return $s$;
```

In short, this local search variation receives an initial solution from the mathematical models, which is stored as the current (s_0), line 1. Then, a set of l_{max} neighborhood structures types (line 2) and the selection strategy (line 3) are defined. The algorithm recursively searches for better solutions until a stop criterion is met (loop from line 4 to 9). At each iteration of this loop, the algorithm generates a set of neighbor solutions (line 6) and selects a new current solution (line 7) according to a search strategy (line 7) for each neighborhood structure of the set N_l (loop between line 5 and 8). Last, the final solution is returned (line 10).

We separate the variable neighborhood descent algorithm tested is two basic types of neighborhood structures: *VND insert with fixed dates* and *VND insert with dates change*.

The former are those variable neighborhood descent algorithms that adopt neighborhood structures of *insert with fixed dates*:

- *VND insert fixed (new rig-existing):* For each iteration of the VND, there is an *Insert with fixed dates in a new rig* followed by an *insert with fixed dates in an existing rig*.
- *VND insert fixed (existing rig-new):* For each iteration of the VND, there is an *Insert with fixed dates in an existing rig* followed by an *insert with fixed dates in a new rig.*
- *VND insert fixed (random)*: At each iteration, the local search selects one randomly *insert with fixed dates* (*in a new rig* or *in an existing rig*) to be used as neighborhood structure.

The latter represents variable neighborhood descent methods using neighborhood structures of *insert with dates anticipation* or *insert with dates postponement* and can be divided in:

- *VND insert dates (anticipation-postponement)*: For each iteration of the VND there is one fixed *insert with dates anticipation* followed by another fixed *insert with dates postponement*.
- *VND insert dates random (anticipation-postponement)*: For each iteration of the VND there is one random *Insert with dates anticipation rig* followed by another random *insert with dates postponement.*
- *VND insert dates random*: At each iteration, the local search select one randomly *insert with dates anticipation/postponement* to be used as neighborhood structure.

5.4. Acceptance Criterion

As mentioned earlier, during the local search in a neighborhood structure, several solutions are found, some might be better or worse than the current solution. In order to determine if a new neighbor solution can be used as the new current solution, before defining the search strategy, it is important to define an acceptance criterion for taking new neighbor solutions. In this study, the algorithm was implemented to accept changing of neighbor only if the proposed solution reduces the rigs budget or if reduces the number of rigs without enhancing the total costs.

5.5. Stop Criterion

Regardless of the neighborhood structure or search strategy chosen, the algorithm relies on a stop criterion to determine when to interrupt its procedures. The proposed heuristic supports different stop criterions*: maximum execution time, maximum iterations numbers* or *lack of neighbor solutions* that pass in acceptance criterion (*i.e.*, that improves the current solution). These stop criterions (with the exception of the *lack of neighbor solutions*) are provided by the user before the execution of the program and can be used simultaneously. Once a stop criterion is match, the algorithm stops the search for new solutions.

5.6. Constructive Heuristic

Any local search algorithm needs to receive an initial solution. In this study, this solution can be provided by any of the mathematical models described in Section 4.2 or by an alternative constructive algorithm. Two constructive greedy heuristics were developed for this problem: *Constructive1* and *Constructive2*. The first tries to build a feasible solution with a low number of rigs, without analyzing the schedule budget. The second method also tries to build a feasible solution, but at each iteration the algorithm analyzes the partial cost of the solution looking forward to generate a schedule with a relatively low budget.

The heuristic *Constructive1* can be described in three steps, as shown in Figure 19. Step 1 sorts the tasks in an insertion queue list. Step 2 inserts the tasks according to their position in the sorted list. In Step 2.1, for each existing rig, the greedy algorithm tries to insert the new tasks in the earliest position as possible after the last task in that rig. If the solution is feasible, it skips to Step 2.3. Otherwise, it goes to Step 2.2, in which there is none available rig that allows a viable insertion of that tasks. Therefore, in this step, the algorithm assigns a new rig into the fleet and inserts the tasks in it as soon as possible. In Step 2.3, the algorithm checks if there are new rigs to be inserted from the sorted list, if this condition is true, it returns to Step 2.1, but if there is none insertion to be made, Step 2 is finalized and Step 3 returns the final solution.

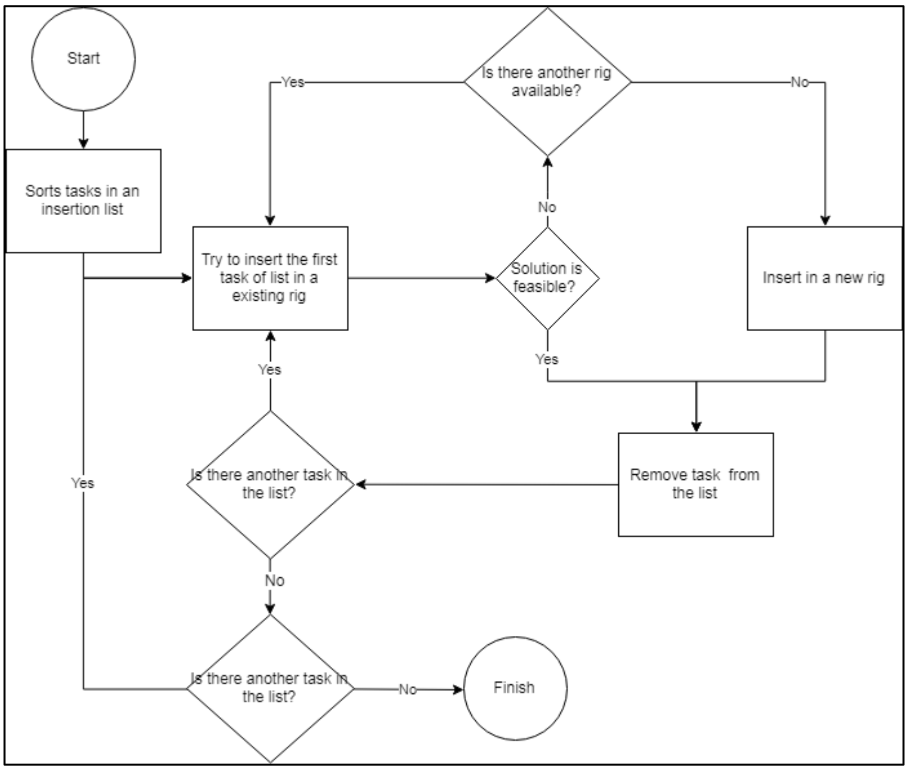
Start
Sorts tasks in an insertion list
Yes
Is there another rig available?
No
Try to insert the first task of list in a existing rig
No
Solution is feasible?
Yes
Insert in a new rig
Yes
Is there another task in the list?
Remove task from the list
No
Yes
Is there another task in the list?
No
Finish

Figure 19. Flowchart for the heuristic *Constructive1* (Source: the author).

Just as the first heuristic, the *Constructive2* has three steps, as shown in Figure 20. The Step 1 is identical with the previously algorithm, sorting the tasks in an insertion queue list. The main difference is in the Step 2, in which the algorithm tries not only to find a feasible solution but iteratively analyzes the solution cost. In Step 2.1, the new tasks from the list are tested in each existing rig, verifying if it is possible to insert them in the earliest insertion possible after the last task in that rig. Case positive, the algorithm stores the budget function. Regardless of the previous step, the algorithm inserts the tasks as soon as possible in a new rig and stores the solution cost in Step 2.2. Then, in Step 2.3, the algorithm selects the viable solution of that iteration with the lowest cost. Additionally, in Step 2.4, the algorithm verifies if there are others tasks from the list to be scheduled, returning to step 2.1 if true or finishing Step 2 otherwise. Last, in Step 3, the algorithm returns the final solution.

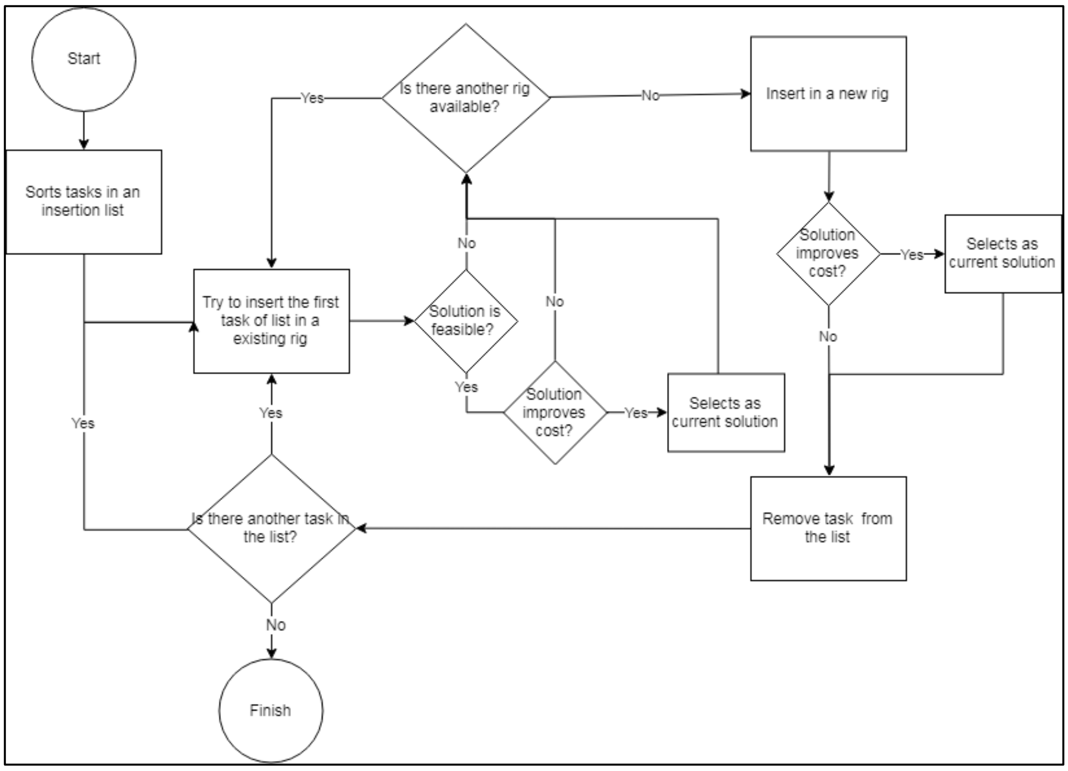
Start
Sorts tasks in an insertion list
Is there another rig available?
Yes
No
Insert in a new rig
Solution improves cost?
Yes
Selects as current solution
No
Try to insert the first task of list in a existing rig
Solution is feasible?
No
Yes
Solution improves cost?
No
Yes
Selects as current solution
Yes
Yes
Is there another task in the list?
Remove task from the list
No
Finish

Figure 20. Flowchart for the heuristic *Constructive2* (Source: the author).

Both methods make use of an insertion list, which can be sorted according by the following criterions, in sorting order:

- **Sort1:** least predecessors; lower $\overline{r_j}$; lower $\overline{d_j}$; higher p_j; lower *Id*.
- **Sort2:** most successors; lower $\overline{r_j}$; lower $\overline{d_j}$; higher p_j; lower *Id*.
- **Sort3:** lower $\overline{r_j}$; lower $\overline{d_j}$; most successors; higher p_j; lower *Id*.
- **Sort4:** lower $\overline{r_j} + p_j$; lower $\overline{d_j}$; higher p_j; lower $\overline{r_j}$; most successors; lower *Id*.

Others sort criteria were also tested, but due to the precedence rules and strict time windows of the tasks, they presented a great difficult to find feasible solutions and, therefore, were omitted of this study.

6 Computational Experiments

In this section, a matheuristic combining the mathematical models, from Section 4.2, and the local search algorithms, from Section 5, is developed and evaluated. A constructive heuristic is also tested as a possible alternative initial solution generator method. All the computational experiments presented in this study were executed in a *Microsoft Windows 7* (*v6.1.7601*) machine of 64-bits with *Intel® Core™ i7-5960x CPU @ 3.00GHz* processors and a 64*GB-RAM* memory. The mathematical models were implemented using *AIMMS® (v3.14*) optimization modelling software and executed using *CPLEX® (v12.6*) optimization solver. The heuristic were developed using C++ computational language in a *Microsoft Visual Studio®* with full optimization (*/Ox*). Two instances, a small (*Instance01*) and a larger one (*Instance02*), were generated based on real life data and are used in these tests. To protect the studied company data, as some information are confidential and sensitive, the parameters were demeaned. The dates were transformed to a daily count from a random origin date. The tasks, wells and blocks were coded to a meaningless id. Last, the costs values were converted to a fictional monetary unit ($m.u.$). The section divides the results according to the instance. Section 6.1 refers to the small instance's computational experiments, and Section 6.2 to those using the large instance. Both sections start with a brief instance's description, followed by the mathematical models and constructive results and ended with the local search stages results. Last, Section 6.3 analyzes the matheuristics results considering both instances, comparing this approach with a purely heuristic one.

6.1. Small instance (*instance01*)

The first instance represents data from a real-life scenario of an important operational block in the Brazilian pre-salt basin of the company studied. Hence, this instance is accountable for a 10-year-old planning horizon with 163 tasks from 4 different projects and from 62 distinct wells. These activities refer to 2 different groups of tasks: completion and drilling. Hence, there are 3 types of drilling considered in this instance. However, as mentioned in Section 4.1, the rigs fleet is considered to be homogeneous and, therefore, there is no distinction between the rigs as long as they respect the precedence relationships and the release/ due dates.

Following Carrilho & Villas Boas (2016) block structuring approach, the 163 activities of the problem were aggregated in bigger tasks (blocks) according to theirs wells and time windows, which resulted in only 62 tasks to be scheduled. Each block represents a subset of activities that must always be scheduled together without any type of preemption or break. Further information about the activities (their id, well, block, processing time, time window and precedence relationship) is described in Appendix II. Despite being an instance based on a real life scenario, the number of tasks is relatively reduced comparing with others cases. Therefore, it is a small instance and will be used to verify the capacity of the heuristics to improve solutions from the mathematical models. First, we analyze the results from the linear programming. Then, a greedy constructive heuristic is examined. Last, the local search is tested using an initial solution obtained by the math models.

6.1.1. Mathematical Programming

Earlier in Section 4.2, two mathematical models were developed to solve the Rig Scheduling Problem of the studied company: Model 1 minimizes the number of rigs and Model 2 minimizes the rigs costs. To avoid nonlinear programming, a procedure described in Figure 6 (Section 4.2) using Models 1 and 2 was created to find a schedule that minimizes the budget. In this mathematical programming procedure, the solution from Model 1 serves as initial solution for the Model 2, which iteratively adds rigs to fleet and determines the fleet size and its schedule that minimizes the rigs budget As one of the budget functions elements is the rigs' contracts costs, the Model 1 finds a solution with minimal contract costs and, due to the tasks short time windows and their precedence constraints, this solution tends to be a tight schedule and a good initial solution for any method that aims to reduce the rigs budget. In order to satisfy the particularities of studied rig scheduling problem, the proposed mathematical models contain a large number of complex constraints, variables and parameters, and require an exhaustive computational effort. Therefore, aiming to reduce the model complexity, some variations and combinations of the mathematical models were implemented and tested, changing the time units (weeks or days) or the mathematical model executed.

Table 9 details the models size for the current *instance01*, in which results #3 and #6 refer to the budget minimization procedure, while results #1 and #4 represent the solutions founds by the Model 1 and results #2 and #5 are obtained running the Model 2 to minimize the budget without receiving an initial solution from Model 1. Results #1, #2 and #3 uses the time horizon in days and results #3, #4 and #5 are in weeks. It is important to notice that the tasks processing times might slightly change when approximating their durations from days to weeks and, therefore, the mathematical models might analyze slightly different solutions spaces. The computational times presented in Table 9 used a stop criterion using the executing time and gap. In Model 1 the solver is interrupted if: the gap is zero and the solver time is lower than 300 seconds; the gap is lower than 2% and the solver time lower than 900 seconds; the gap and the solver time are, respectively, lower than 5% and 1800 seconds, or the solution is feasible and the time more than 1,800 seconds. The Model 2 solving is stopped if: the gap is zero and the solver time is lower than 300 seconds; the gap is lower than 3% and the solver time lower than 1,800 seconds; the gap and the solver time are, respectively, lower than 5% and 3,600 seconds, or the solution is feasible and the time greater than 14,400 seconds. In addition, the computational times with asterisk mean that the solver was interrupted before the stop criterion due to lack of computer memory.

Table 9. Mathematical models sizes for *instance01* (Source: the author).

#	Mathematical models	Time unit	Small instance (*instance01*)			
			Constraints	Variables	Nonzero coefficients	Computational time
1	Model 1 (*min. rigs*)	Day	59,553	41,483	3,690,456	136
2	Model 2 (*min. budget*)	Day	184,856	2,877,489	489,464,541	21,217*
3	Model 1 (*min. rigs*) $\rightarrow$ Model 2 (*min. budget*)	Day	316,271	33,545	43,061,177	26,468*
4	Model 1 (*min. rigs*)	Week	8,595	6,052	102,039	7
5	Model 2 (*min. budget*)	Week	538,873	113,985	103,686,278	137,547
6	Model 1 (*min. rigs*) $\rightarrow$ Model 2 (*min. budget*)	Week	45,482	4,920	980,737	28,892

Table 9 results emphasize the rig scheduling problem's complexity and evidence the difference in size between the two models. Clearly, the mathematical model minimizing the rigs budget is much larger than the one minimizing the number of rigs. As a result, the computational effort required to solve Model 2 is a lot bigger. The mathematical procedure using Model 1 as initial solution and set limiter to Model 2 was able to reduce the budget model size, but it was still much bigger than the Model 1. Hence, the mathematical models using time units in week are much smaller than the same model considering the horizon periods in weeks, which also affects the executing time. The option closest to the real problem would be to execute the mathematical models minimizing the rigs budget with the periods in days, which are the realistic parameters. However, as noticed, these model (results #2 and #3) are extremely complex and requires a huge computational effort, having finished with the computer memory before its stop criterion, even for this small instance. Next, Table 10 details the solutions found by these models. As mentioned earlier, the objective functions of Model 1 and Model 2 are different, the first minimizes the fleet of rigs and the second minimizes the rigs budget. Also, the mathematical models in weeks approximate the tasks durations and time windows and, as a result, the real budget of the schedule (in days) might differ from the objective function found by the model in weeks. Further details on the solutions found by the mathematical models are presented in Appendix III.

Table 10. Mathematical models results for *instance01* (Source: the author).

#	Mathematical models	Time unit	Small instance (*instance01*)				
			Rigs Fleet	Budget in days $(u.m.\times 10^6)$	Objective Function $^{**}(u.m.\times 10^6)$	Lower Bound $^{**}(u.m.\times 10^6)$	GAP (%)
1	Model 1 (*min. rigs*)	Day	4	1,108.6	10	10	0.00
2	Model 2 (*min. budget*)	Day	-	-	-	-	-
3	Model 1 (*min. rigs*) $\rightarrow$ Model 2 (*min. budget*)	Day	4	1,028.4	1,028.4**	974.0**	5.29*
4	Model 1 (*min. rigs*)	Week	4	1,108.9	10	10	0.00
5	Model 2 (*min. budget*)	Week	5	1030.1	1,034.9	974.0	5.88
6	Model 1 (*min. rigs*) $\rightarrow$ Model 2 (*min. budget*)	Week	4	1,029.1	1,032.4**	980.6**	5.02

It is important to notice that the Model 2 alone in days (results #2) was stopped due to lack of memory and did not found feasible solutions or even a lower bound. Hence, the solution quality was directly related to the model's complexity, *i.e.*, models of larger size have found better solutions than the simpler models, but required more computational effort. As expected, the models using time units in days are capable of finding better solutions than the models in weeks, but, clearly, these gains in costs are too low to compensate the computational effort. Therefore, if the decision maker needs a solution in a practical time, it is better to use the model in weeks. The results also show that even though the Model 1 does not find good solutions as the Model 2, it is able to find a schedule with the same rigs fleet and just around 7-8% worse with little computational effort. It is evident that the Model 1 is the appropriate method to generate solutions for the heuristics tests. As budget solutions are very similar between Model 1 in weeks and Model 1 in days, the local search experiments in the next section will be performed with the fastest model, *i.e.*, the Model 1 in weeks. Last, the mathematical procedure using Model 1 and Model 2 was able to reduce the computational effort required by the model and, as a result, allow to find better solutions than the Model 2 alone in a much smaller time. Next, the constructive heuristic results are presented for this instance.

6.1.2. Constructive Heuristic

Earlier in Section 5.6, two constructive heuristic algorithms were described as an alternative initial solution generator method. These two greedy methods were tested in the current instance varying the sorting criteria, also presented in Section 5.6. These results of the constructive methods are presented in Table 11, which shows, the budget function, the computational time and the deviation from the best known solutions, found by the mathematical models in Table 10.

Table 11. Constructive heuristics results for small instance 01 (Source: authors).

	Constructive Heuristic Results				
	Type	Sorting Criter.	Budget ($m.u.\times 10^6$)	Comp. time (sec.)	Dev. from BKS (%)
Small instance 01 BKS: $m.u.\,1{,}028.4\times 10^6$	1	1	1,515.6	0.003	32.1
		2	1,558.6	0.001	34.0
		3	1,181.9	0.001	13.0
		4	1,151.8	0.001	10.7
	2	1	1,657.6	0.201	38.0
		2	1,687.2	0.204	39.0
		3	1,161.0	0.137	11.4
		4	1,132,4	0.144	9.2

Analyzing Table 11 it is possible to observe that the constructive heuristics require an extremely low computational time and that the quality of the solutions generated relies mainly in the sorting criterion used. Clearly, there is a superiority between the sorting criteria 3 and 4 (which use the indicators based on the release date as the first sorting criterion) compared with criteria 1 and 2 (that are based in the precedence relationships). On the other hand, it is hard to notice a performance difference between the constructive heuristic 1 and 2, in some sorting criteria the first heuristic has found better results and in others cases, the second algorithm was superior to the first one. Despite the low computational efforts, only half of the greedy heuristics tested were able to create valuable solutions, with a relatively low cost benefit, finding solution with around 10% deviation from the BKS in an insignificant time. Next, the local search from Section 5 are compared with the mathematical models in the current instance.

6.1.3.
Local Search

Earlier in Section 5, several local search possibilities were mentioned. Aiming to determine the heuristic capacity and compare them with the results found by mathematical models, a variety of tests was carried out with the proposed algorithm – using different neighborhood structures and search strategies and using as stop criterion the lack of improvement in neighbor solutions, *i.e.*, interrupting the search when no better solution was found in the chosen neighborhood. Overall, 156 variations of the heuristic were tested for the current instance using the results #4 from Table 10 as an initial solution. These results are fully discriminated in the Appendix IV (which numerate the methods) and summarized in Table 12, considering the results #3 from Table 10 as the best known solution.

Analyzing the results of the neighborhood structures, it is clear that, regardless of the neighborhood structure chosen, the heuristic was able to reduce millions of dollars, improving between 2.1% and 6.9% the initial solution received from the mathematical model 1 in weeks. Besides, in this instance, all the local search tested have a relatively low deviation from the best known solution found by Model 1+Model 2 in days, with deviation varying from 0.36% to 5.61%. Furthermore, the computational effort required by the algorithm to obtain strong solutions is much smaller than the one needed by the mathematical models minimizing the rigs budget. While these models need at least 7 hours, the heuristics are able to achieve very similar solutions in just a few minutes.

Table 12. Mean results for each neighborhood structure in *instance01* (Source: the author).

Group	Heuristic	Neighborhood		*Instance01* (Initial Solution: um 1,108.9 × 10^6) (BKS, Best Known Solution: um 1,028.4 × 10^6)		
				Avg. Budget ($u.m.\times 10^6$)	Dev. from BKS (%)	Avg. time (sec.)
Insert with Fixed Dates	*LS*	*Existing Rig*		1,086.1	5.61	0.11
		New Rig		1,073.5	4.39	0.16
		New Rig + Existing		1,072.2	4.26	0.33
	VND	*Existing – Rig*		1,069.9	4.04	0.42
		New Rig – Existing		1,069.9	4.04	0.39
		Random		1,069.9	4.04	0.42
Insert with Dates Change	*LS*	*Anticipation (After-From)*		1,045.2	1.63	27.92
		Anticipation (After-In)		1,051.5	2.25	14.29
		Anticipation (Before-From)		1,059.4	3.01	37.35
		Anticipation (Before-In)		1,084.3	5.44	27.75
		Postponement (After-From)		1,072.1	4.25	29.97
		Postponement (After-In)		1,084.2	5.43	15.41
		Postponement (Before-From)		1,071.4	4.18	33.09
		Postponement (Before-In)		1,084.0	5.41	23.35
	VND	*Anticipation (After-From)*	*Postponement (After-From)*	1,046.6	1.77	40.06
		Anticipation (After-From)	*Postponement (After-In)*	1,043.1	1.43	47.96
		Anticipation (After-From)	*Postponement (Before-From)*	1,034.9	0.64	81.37
		Anticipation (After-From)	*Postponement (Before-In)*	1,032.1	0.36	77.94
		Anticipation (After-In)	*Postponement (After-From)*	1,042.3	1.36	40.54
		Anticipation (After-In)	*Postponement (After-In)*	1,051.5	2.25	23.20
		Anticipation (After-In)	*Postponement (Before-From)*	1,033.6	0.51	55.95
		Anticipation (After-In)	*Postponement (Before-In)*	1,032.1	0.36	49.04
		Anticipation (Before-From)	*Postponement (After-From)*	1,059.4	3.01	56.47
		Anticipation (Before-From)	*Postponement (After-In)*	1,051.5	2.25	83.73
		Anticipation (Before-From)	*Postponement (Before-From)*	1,038.3	0.97	104.64
		Anticipation (Before-From)	*Postponement (Before-In)*	1,032.1	0.36	111.19
		Anticipation (Before-In)	*Postponement (After-From)*	1,069.4	3.99	70.25
		Anticipation (Before-In)	*Postponement (After-In)*	1,084.3	5.44	38.24
		Anticipation (Before-In)	*Postponement (Before-From)*	1,051.9	2.29	70.32
		Anticipation (Before-In)	*Postponement (Before-In)*	1,049.0	2.01	63.74
		Random		1,049.1	2.02	34.82
		Random (Anticipation-Postponement)		1,037.1	0.85	66.02

It is also possible to verify some performance differences for each heuristic group. For instance, the *Insert with fixed dates* neighborhood structures seem to find fast solutions practically with no computational effort, especially the *VND* heuristic that had slightly better improvements than the simpler local search approach. Still, these solutions had a significant deviation from the best and, therefore, this group is only recommended to be used, for example, in genetic algorithms and simulators where the computational effort must be as lower as possible. On the other hand, the heuristics groups using the *Insert with dates change* neighborhood structures can find stronger solutions, despite usually needing more computational effort. It is also possible to observe differences between the local search and the *VND* algorithms for this neighborhood structure. The first could find some better solutions than the fixed dates, but not always and with a much larger computational efforts. The best results seem to be in the *VND* heuristics using the *Insert with dates change*, which were able to find, in minutes, solutions with less than one percent of deviation to the best known solutions. Nonetheless, it is very difficult to compare the heuristics in a deeper level or even to decide the neighborhood structure that best fit to the problem just by analyzing the results from Table 12. Further tests are needed in a larger instance, which will be performed in the next section.

6.2. Large Instance (*instance02*)

In the earlier section, the heuristics and mathematical models were tested in a small instance, in which the heuristic proved to be extremely powerful and efficient, capable of find solutions similar from the exact methods with much less computational effort. This capacity is strongly need in larger and complex instances where the linear programming models are unable to find good solution with a practical executing time. Therefore, the current section presents a larger instance to be used in test with the mathematical models (Section 6.2.1), the greedy algorithm (Section 6.2.2) and the local searches (Section 6.2.3).

Just as the first instance, the second instance was generated using data from real-life scenarios of the company studied over a 10-year-old planning horizon. However, this instance is a much larger one and considers multiple operational blocks of the Brazilian pre-salt basin. Therefore, this instance is accountable for 326 tasks from 23 different projects and from one hundred seventy-three distinct wells. Overall, these activities refer to three different groups of tasks: completion, drilling and evaluation. Hence, there are four types of completion and three types of drilling considered in this instance. However, equally as in the previous instance, the rigs fleet is consider to be homogeneous and, therefore, there is no distinction between the rigs and tasks as long as they respect the precedence relationships, the release dates and the due dates. The 326 activities of the problem were aggregated in larger tasks (blocks) according to its wells and times windows, which resulted in 186 blocks to be scheduled. Each block represents a subset of activities that must always be scheduled together without any type of preemption or break. Further activities information (id, well, block, processing time, time window and precedence relationship) are described in Appendix II. This aggregation of tasks reduces the problem's size and enables the application of exact methods.

6.2.1. Mathematical Programming

As mentioned earlier, the proposed mathematical model contains a complex and large number of constraints, variables and parameters. Consequently, the mathematical model requires extensive computational effort. The matheuristic approach explained in Section 5 combines the use of mathematical programming tools and heuristic algorithms, being a possible alternative to reduce the problem complexity. Nonetheless, it is still important to use the correct mathematical models. In addition, in order to obtain a proper evaluation of the heuristics performance, the results need to be compared with a pure exact approach.

Just as in the small instances, several variations and combinations of the mathematical models were implemented and tested, varying the time periods (weekly or daily) or the mathematical model executed. Table 13 details the models sizes for the current *instance02*, in which results #3 and #6 refer to the budget minimization procedure using mathematical models 1 and 2, results #1 and #4 represent the solutions founds by model 1 and results #2 and #5 use the model 2 alone. Results #1, #2 and #3 refer tp the time horizon in days and results #4, #5 and #6 are in weeks. It is important to notice that the tasks processing times might slightly change when approximating their durations from days to weeks and, therefore, the mathematical models can analyze slightly different solutions spaces.

Table 13. Mathematical models size for *instance02* (Source: the author).

#	Mathematical models	Time unit	Large instance (*instance02*)			
			Constraints	Variables	Nonzero coefficients	Computational time
1	Model 1 (*min. rigs*)	Day	250,122	271,632	17,446,630	10,701
2	Model 2 (*min. budget*)	Day	-	-	-	-*
3	Model 1 (*min. rigs*) → Model 2 (*min. budget*)	Day	3,463,387	185,004	490,183,931	101,857*
4	Model 1 (*min. rigs*)	Week	36,074	40,071	557,105	395
5	Model 2 (*min. budget*)	Week	451,203	487,816	26,367,197	21.217*
6	Model 1 (*min. rigs*) → Model 2 (*min. budget*)	Week	639,435	35,552	14,230,037	57,278

The computational times presented in Table 13 were using a stop criterion according to the executing time and the gap. In Model 1 the solver is interrupted if: the gap was zero and the solver time was lower than 300 seconds; the gap was lower than 2% and the solver time lower than 900 seconds; the gap and the solver time were, respectively, lower than 5% and 1800 seconds, or the solution was feasible and the time more than 1,800 seconds. As to the Model 2's solver, it is stopped if: the gap was zero and the solver time was lower than 300 seconds; the gap was lower than 3% and the solver time lower than 1,800 seconds; the gap and the solver time were, respectively, lower than 5% and 3,600 seconds, or the solution was feasible and the time more than 14,400 seconds. In addition, the computational times with asterisk mean that the solver was interrupted before the stop criterion due to lack of computer memory. In the case of the Model 2 alone in days, the solver was unable to run due to the problem size that was too big even for the pre-solver execution. Again, the sizes differences between the models variations are evident. Just as in the small instance, the mathematical model minimizing the rigs budget is much bigger than the one minimizing the number of rigs and, consequently, the computational effort required to solve Model 2 is higher. Also like in the previous scenario, the mathematical models using time units in week are much smaller than the same model considering the horizon periods in weeks, needing more executing time. Due to the instance size, the model 2 in days had to be interrupted before its stop criterion as after more than one day was unable to find bound and new viable solutions, exhausting the computer's memory.

Next, Table 14 details the solutions found by these models at the current instance. As mentioned earlier, the objective functions of the Model 1 and Model 2 are different, the first minimizes the fleet of rigs and the second minimizes the rigs budget. Also, the mathematical models in weeks approximate the tasks durations and windows and as a result the real budget of the schedule (in days) might differ from the objective function found by the model in weeks. Due to size of the problem, the model was unable to save solution found by the Model 2 in weeks (results #5) and calculate the budget function in days. Further details on the solutions found by the mathematical models are presented in Appendix III, which describes the schedules found in each result.

Table 14. Mathematical models results for *instance02* (Source: the author).

#	Mathematical models	Time unit	Large instance (*instance02*)				
			Rigs Fleet	Budget in days $(mu \times 10^6)$	Model's Objective $^{**}(mu \times 10^6)$	Lower Bound $^{**}(mu \times 10^6)$	GAP (%)
1	Model 1 (*min. rigs*)	Day	7	2,164.6	28	3.7	86.66
2	Model 2 (*min. budget*)	Day	-	-	-	-	-
3	Model 1 (*min. rigs*) → Model 2 (*min. budget*)	Day	7	2,164.6*	-	-	-
4	Model 1 (*min. rigs*)	Week	7	2,153.9	28	28	0.00
5	Model 2 (*min. budget*)	Week	-	-	14,225.1**	1,554.5**	89.07
6	Model 1 (*min. rigs*) → Model 2 (*min. budget*)	Week	8	1,741.9	1,751.2**	1,554.5**	11.23

In this larger instance, the performance difference between the models has gotten worse. As a result, the Model 2 that minimizes the budget using the periods in days was unable to find any integer solution (results #3), even in the procedure with Model 1 and 2 (results #2), in which Model 2 receives an initial solution from Model 1 (results #1), it was unable to improve it. Also, due to the models' sizes, all the mathematical models using daily time unit have had difficulties to find integer solutions and to close theirs gaps and, therefore, they were outperformed by the models in weeks. Hence, it is important to notice that the number of rigs in the fleet is very similar in both Model 1 and Model 2, as shown in results #1, #4 and #6. However, the difference in the objective function is more than a billion of dollars, which might justify the computational effort required to obtain Model 2 results. Last, the experiments emphasize the necessity of an alternative method to the mathematical programming for the studied problem. In the small instance the mathematical models were able to provide strong solutions in a relatively short time, but when using larger instances, the models are not able to find good solutions and require a very large amount of computational effort.

In conclusion, the goal of this study is to propose a practical and efficient matheuristic method for solving the company's rig scheduling problem. Therefore, both the matheuristic and the mathematical model used need to be the efficient as possible. In order to fulfill this requirement, the most suitable mathematical model is the rigs fleet minimization model using time units in weeks. Nonetheless, there is the possibility to use the constructive heuristic to generate faster initial solutions. Next, more tests are performed aiming to determine the best approaches for this problem.

6.2.2.
Constructive Heuristic

In the small instance, Section 6.2, two constructive heuristics combined with four tasks sorting structures were tested. In those experiments, some of these greedy heuristics were able to find relatively good solutions in an insignificant computational time. However, when compared with the mathematical models in these small instances, the linear programming computational time was still relatively low enough to justify its use and the solution quality improvement. This situation might not be true in the large instance such as the current one. Therefore, computational experiments with the two constructive algorithms were performed on the large instance and their results are shown in Table 15.

Table 15. Constructive heuristics results for small instance 01 (Source: authors)

	Constructive Heuristic Results				
	Type	Sorting Criter.	Budget ($m.u.\times 10^6$)	Comp. time (sec.)	Dev. from BKS (%)
Large instance 02 BKS: $m.u.\,1{,}741.9 \times 10^6$	1	1	2,968.0	0.005	41.3
		2	3,070.8	0.003	43.3
		3	2,169.5	0.002	19.7
		4	2,139.8	0.002	18.6
	2	1	3,064.0	1.039	43.1
		2	2,836.7	0.904	38.6
		3	1,949.3	1.115	10.6
		4	1,923.9	1.311	9.5

Just as in small instance, the quality of the solutions relies mainly in the sorting criteria used, where the best constructive results are with sorting criterion 3 and 4, and the computational effort continues to be almost insignificant. Even though, it may be hard to notice a performance difference between the constructive heuristic 1 and 2, when comparing the these heuristics while using the best sorting criterion (3 and 4), it is possible to observe that the combination of the sorting criterion 3 or 4 with the constructive heuristic 2 is clearly superior than the others greedy algorithms, with a relatively low computational effort for a solution with just 10 percent deviation to the BKS.

Nevertheless, despite the low computational time, only half of the greedy heuristics tested were able to create valuable solutions, with a relatively low cost benefit. Next, the local search from Section 5 are compared with the mathematical models in the current instance.

6.2.3. Local Search

As mentioned earlier, the mathematical models have great difficult in solving large instances such as the current one. Aiming to provide an alternative method to replace the exact approaches and support oil and gas companies in theirs rig scheduling problem, several local search algorithms were developed and presented in Section 3. These algorithms were tested in a small instance and have achieved powerful results in Section 6.2, which encourages to test them in a larger instance. The current instance is a very large scenario where mathematical models tend to require a lot of computational effort to find solutions, as seen in Section 6.2.1. Consequently, the heuristics are probably the only viable and practicable method available for this problem.

In order to verify this, a variety of tests in 156 variations of the heuristic was carried out with the proposed algorithm – using different neighborhood structures and search strategies and using as stop criterion the lack of improvement in neighbor solutions – with the results #4 from Table 14 as an initial solution and the results #6 as best known solution. These results are fully discriminated in the Appendix IV (which numerate the methods) and summarized in Table 16.

Table 16. Mean results for each neighborhood structures (Source: the author).

Group	Heuristic	Neighborhood Structure		*Instance02* (Initial Solution: *mu* 2,153.9 × 10^6) (BKS, Best Known Solution: *mu* 1,741.9 × 10^6)		
				Avg. Budget (*mu* × 10^6)	Dev. from BKS (%)	Avg. time (sec.)
Insert with Fixed Dates	*LS*	*Existing Rig*		1,923.1	10.4	2.68
		New Rig		1,953.7	12.2	0.54
		New Rig + Existing		1,811.8	4.0	8.20
	VND	*Existing – Rig*		1,808.9	3.8	7.43
		New Rig – Existing		1,827.2	4.9	8.94
		Random		1,808.9	3.8	7.41
Insert with Dates Change	*LS*	*Anticipation (After-From)*		1,757.3	0.9	449.14
		Anticipation (After-In)		1,827.0	4.9	155.10
		Anticipation (Before-From)		1,760.8	1.1	471.10
		Anticipation (Before-In)		1,748.9	0.4	247.81
		Postponement (After-From)		1,805.0	3.6	487.87
		Postponement (After-In)		1,916.2	10.0	235.87
		Postponement (Before-From)		1,813.2	4.1	572.92
		Postponement (Before-In)		1,917.2	10.1	192.48
	VND	*Anticipation (After-From)*	*Postponement (After-From)*	1,691.0	-2.9	1,241.18
		Anticipation (After-From)	*Postponement (After-In)*	1,769.6	1.6	577.80
		Anticipation (After-From)	*Postponement (Before-From)*	1,691.1	-2.9	1,489.75
		Anticipation (After-From)	*Postponement (Before-In)*	1,706.1	-2.1	1,431.97
		Anticipation (After-In)	*Postponement (After-From)*	1,685.3	-3.2	864.70
		Anticipation (After-In)	*Postponement (After-In)*	1,826.9	4.9	215.17
		Anticipation (After-In)	*Postponement (Before-From)*	1,697.5	-2.6	1,161.08
		Anticipation (After-In)	*Postponement (Before-In)*	1,810.8	4.0	673.09
		Anticipation (Before-From)	*Postponement (After-From)*	1,665.2	-4.4	1,646.01
		Anticipation (Before-From)	*Postponement (After-In)*	1,737.0	-0.3	979.74
		Anticipation (Before-From)	*Postponement (Before-From)*	1,664.0	-4.5	1,924.12
		Anticipation (Before-From)	*Postponement (Before-In)*	1,682.2	-3.4	1,591.55
		Anticipation (Before-In)	*Postponement (After-From)*	1,690.4	-3.0	855.07
		Anticipation (Before-In)	*Postponement (After-In)*	1,748.3	0.4	375.33
		Anticipation (Before-In)	*Postponement (Before-From)*	1,671.3	-4.1	1,277.02
		Anticipation (Before-In)	*Postponement (Before-In)*	1,725.0	-1.0	853.02
		Random		1,744.4	0.1	395.17
		Random (Anticipation-Postponement)		1,678.1	-3.7	1,144.95

Analyzing the average local search results for each neighborhood structure it is clearly that the heuristic was able to reduce millions of dollars, regardless of the neighborhood structure chosen. Even so, it is possible to separate observe different performance in each heuristic group. The local searches using *insert with fixed dates* neighborhood structures were extremely fast, finding new solutions with almost none computational effort. However, despite the improvement of the initial solution, the deviation to the *BKS* were usually very high. When using the *VNDs* with the previously neighborhood structure, this deviation is much lower and requires a similar computational effort. Basically, the heuristic groups using the *insert with fixed dates* neighborhood structures seem to find fast solutions practically with no computational effort, especially the *VND* heuristic that had better improvements than the simple local search heuristics. Just as in the small instance, these groups are recommendable to be used in genetic algorithm and simulations where it is important to find solutions fast.

On the other hand, the heuristics groups using the *insert with dates change* neighborhood structures can find stronger solutions, despite needing much more computational effort. It is also possible to observe performance difference between the two neighborhood structures *anticipation* and *postponement* of the local searches using *insert with dates change*. Usually, the local searches using only *anticipation* movements had better results with less computational effort, finding solutions very similar to the BKS of model 2 with a low computational effort. Meanwhile, the local searches using only postponement had much more difficult to find strong solutions, requiring a lot more executing time to find worse solutions. The best results seem to be in the *VND* heuristics using the *insert with dates change*, these heuristics performed exhaustive local searches that were able to obtain new better solutions, lower than the best known solutions, with a computational effort satisfactory and much smaller than the one need for the mathematical models. Next, more tests are performed aiming to select the best matheuristic approaches, analyzing its performance on both instances.

6.3. Matheuristic Results

In the last sections, the heuristics and mathematical models were tested in a small and in a large instance. In both instance, the heuristic performance was extremely satisfactory, especially the VND heuristics with the *insert with dates change* neighborhood structure that have outperformed the mathematical models in the large instance. Still, it is very difficult to decide which heuristic to be used in the matheuristic just by analyzing these last results or if the use a constructive algorithm in place of the mathematical programming as an initial solution generator for the local search methods would be a better approach. Therefore, this section tests the heuristics and models in a matheuristic perspective. First, a Pareto frontier chart (Chart 3) was generate for the mean between instances 01 and 02 results presented in the Appendix IV, in which the results are also numbered and each point in Chart 3 refers to a result numbered in the Appendix IV. The Pareto frontier chart allows to analyze the trade-off between different objectives diving the solutions in two groups: the *Pareto frontier* set, solutions that are not outperformed by another solutions, and *not-Pareto frontier* set, solutions that are outperformed by another solutions (Lotov & Miettinen, 2008).

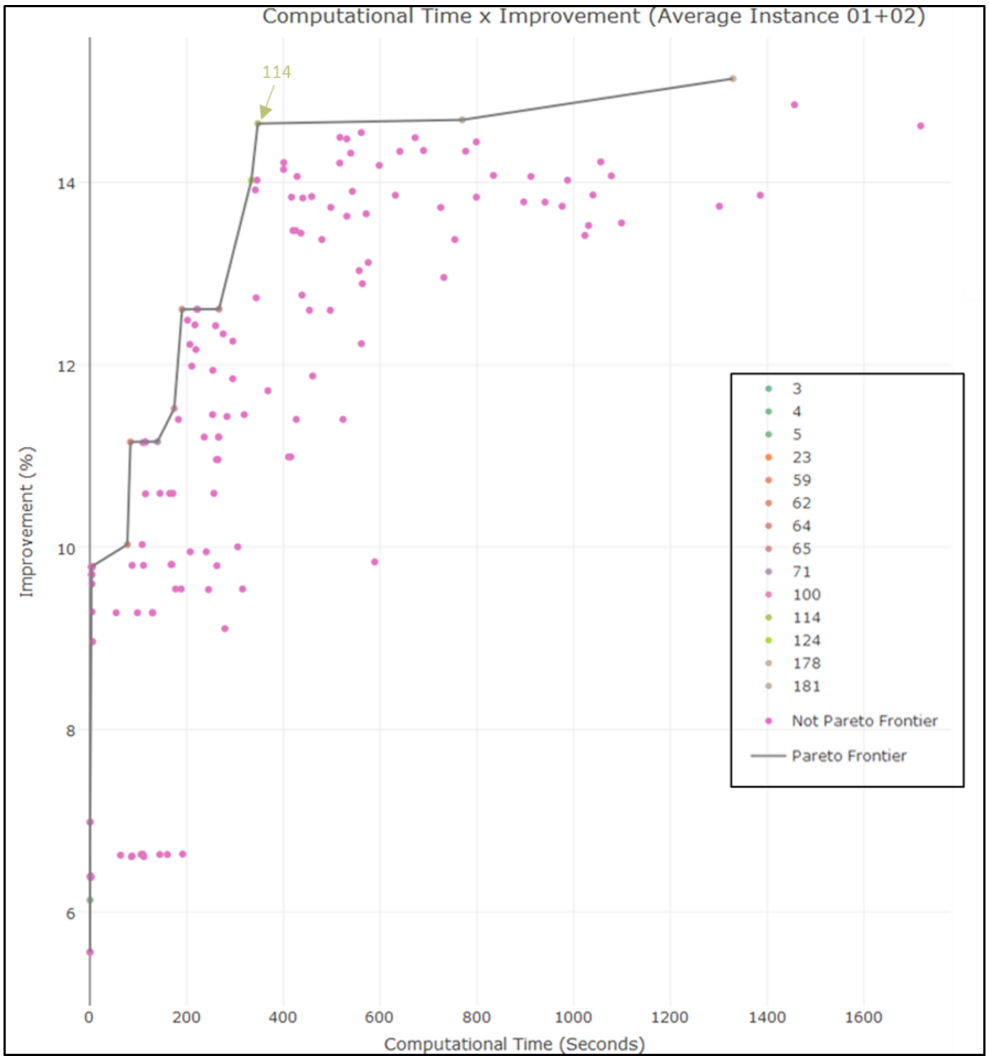

Chart 3. Pareto frontier chart for the parametrization results (Source: the author).

Our goal is to reduce the costs, *i.e.*, to improve the objective function, with minimal computational effort, resulting in a set of just 14 approaches that aren't outperformed by any the solutions found by any other heuristic, as shown in Chart 3. For the studied problem, where the budget is in scale of billions, a slight reduction in the objective function results in great gain for the company. With this in mind, it is possible to say that solution found by the method #114, the *VND* using *insert with dates change Anticipation(After–In)–Postponement(After–From)* neighborhood structure with *best-improvement* search strategy, has the best trade-off between executing time and cost reduction. Several others tests were performed in the instances 01 and 02 aiming to verify the best fitted heuristics and are presented in the Appendix V, which also analyzes the performance of the selected VND.

In order to verify the matheuristic approach and to compare it with the linear programming models, the selected heuristic (*VND* #114) was tested in instances 01 and 02 using the initial solutions from the mathematical models were able to find one (presented in Section 6.1.1. and 6.2.1.). These results are shown in Table 17. The final schedules are also presented in Appendix VI.

Table 17. Matheuristics results for instance 01 and 02 (Source: the author).

Instance	Mathematical programming			Matheuristic using *VND #114*			
	Math. Models	Budget ($mu \times 10^6$)	Comp. time (sec.)	Improve (%)	Total time (sec.)	Final budget solution ($mu \times 10^6$)	Dev. from BKS(%)
Small instance 01 Best Known Solution: $1{,}028.4 \times 10^6$ *m.u.*	1 (days)	1,108.6	136	6.9	163	1,032.1	0.4
	$1 \rightarrow 2$ (days)	1,028.4	26,468	0.0	26,486	1,028.4	0.0
	1 (weeks)	1,108.9	7	6.7	36	1,034.2	0.6
	2 (weeks)	1,030.1	137,547	0.0	137,547	1,030.1	0.2
	$1 \rightarrow 2$ (weeks)	1,029.2	28,892	0.0	28,910	1,029.2	0.1
Large instance 02 Best Known Solution: $\$1{,}593.5 \times 10^6$ *m.u*	1 (days)	2,164.6	10,701	26.4	10,957	1,593.5	0.0
	$1 \rightarrow 2$ (days)	2,164.6	101,857	26.4	102,113	1,593.5	0.0
	1 (weeks)	2,153.9	395	22.6	1,060	1,668.2	4.5
	$1 \rightarrow 2$ (weeks)	1,741.9	57,278	3.9	57,686	1,674.1	4.8

The results from Table 17 support the capacity of the matheuristics to find schedules with a low budget. In addition, the best known solutions for the instance 01 was found through the matheuristic procedures. Different from the others combinations, the procedure mixing the mathematical model 1 with the local search algorithm proved to be an efficient method, finding near optimal solutions with seconds or minutes, depending on the instance’s size. Even though, the best results were using the mathematical models in days, the small budget difference between the exact models in days and in weeks does not compensate the computational effort necessary for executing the mathematical model in days. It is possible to observe that the local search was able to improve solutions far away from optimal, but, when using an initial solution extremely near from the optimal, it has shown a slight difficult of leaving the local optimum and improve it.

Another possibility is to use initial solutions provided by the constructive heuristics as input to the local search instead of the mathematical model. This purely heuristic approach was also tested using the greedy algorithms described in Section 5.6 and the selected VND #114. The initial solutions were taken from the results in Table 11 and Table 15, while the best known solutions (BKS) are the best results from Table 17. Bellow, Table 18 presents the results of the combination the constructive heuristics and variable neighborhood descent #114.

Table 18. Constructive heuristic with local search results (Source: the author).

Instance	Constructive Heuristic				Results after the *VND #114*			
	Const. Heur.	Sort. Criterion	Budget ($mu \times 10^6$)	Comp. time (sec.)	Improve (%)	Total time (sec.)	Final budget solution ($mu \times 10^6$)	Dev. from BKS(%)
Small instance01 Best known solution: $1{,}028.4 \times 10^6 m.u.$	1	1	1,515.6	0.003	19.3	116.7	1,223.9	16.0
		2	1,558.6	0.001	19.0	98.7	1,262.6	18.5
		3	1,181.9	0.001	8.1	61.6	1,085.9	5.3
		4	1,151.8	0.001	6.8	69.4	1,073.0	4.2
	2	1	1,657.6	0.201	25.9	145.4	1,228.4	16.3
		2	1,687.2	0.204	27.5	105.7	1,224.0	16.0
		3	1,161.0	0.137	6.4	69.8	1,086.7	5.4
		4	1,132,4	0.144	7.3	72.6	1,049.6	2.0
Large instance02 Best known solution: $1{,}593.5 \times 10^6 m.u.$	1	1	2,968.0	0.005	29.7	767.1	2,085.5	23.6
		2	3,070.8	0.003	35.1	964.9	1,991.7	20.0
		3	2,169.5	0.002	17.6	1,007.2	1,786.9	10.8
		4	2,139.8	0.002	16.1	620.2	1,796.3	11.3
	2	1	3,064.0	1.039	24.3	1,016.9	2,319.7	31.3
		2	2,836.7	0.904	25.9	796.7	2,103.3	24.2
		3	1,949.3	1.115	7.9	561.6	1,795.4	11.2
		4	1,923.9	1.311	6.1	1,094.2	1,807.5	11.8

Analyzing the results from Table 18, it is clear that the combination of the greedy constructive heuristics with the local search is a viable possibility to generate reliable and efficient solutions, if the proper heuristic and sorting criteria was selected. The best results were obtained with the sorting criteria 4 and, usually with the constructive algorithm 2. Thus, when combining the constructive heuristic and the local search, the quality of the final solution strongly relies in the quality of the initial solution provided by the greedy algorithm. Regardless the constructive method used, the computation effort seems to be very similar and low, needing around 2 minutes in the small instance and 10 minutes in large one. However, despite the extremely low computational effort, the solutions quality found by the combination of these two heuristics are still a bit far away from the best solutions. In the small instance the matheuristic combination of the Model 1 in weeks with the VND outperforms the purely heuristic approach, as much as in time or in solution quality. This situation is repeated in the large instance, in which the constructive and the local search can provide faster solutions, but with quality still poor if compared with the ones found by the matheuristic and, therefore, does not justify using it instead of the matheuristic. Next, Table 19 compares the overall results of the hybrid and purely heuristic methods, analyzing just best results of each one.

Table 19. Average results for best heuristic and matheuristic methods (Source: the author).

Instance	Constructive Heuristic + VND #114			
	Method	Final budget solution ($mu \times 10^6$)	Dev. from BKS (%)	Time (sec.)
Small instance01 Best known solution: $1{,}028.4 \times 10^6 m.u.$	Constructive (Sort. 3 or 4) + VND	1,073.8	4.2	68.35
	Matheuristic (Model 1 +VND)	1,033.2	0.5	99.5
Large instance02 Best known solution: $1{,}593.5 \times 10^6 m.u.$	Constructive (Sort. 3 or 4) + VND	1,796.5	11.3	820.8
	Matheuristic (Model 1 +VND)	1,630.9	2.3	6,008.5

Even though the use of the constructive as initial solution requires less computational effort, the solutions are still extremely far from the best known solutions, with average deviation of 4.2% in the small instance and over 10% in the large instance. It is possible to obtain much better solutions using the linear programing models as initial solution, with deviation of 0.5% in the small instance and 2.3% in the large instance and just a little bit more of executing time.

Therefore, the recommended solution method is the matheuristic using the mathematical model 1 (weeks) with the variable neighborhood descent #114. This combination is capable of efficiently finding strong solutions with a very low computational time, needing just some seconds (small instances) or minutes (large instances). As the rigs costs sum up billions of dollars, the proposed matheuristic approach has potential to reduce millions (or even billions) of dollars for the studied company, which will result in an important competitive advantage.

7
Conclusion and future research

The Oil & Gas companies are extremely important to the nation’s development and economy, being the supplier for most of the world’s energy and for several industries. One of the most critical, complex, expensive and risky phases in their production chain is the Exploration & Production of Oil & Gas, which relies mainly in the operations of rigs. However, the rigs are a scarce and costly resource. Furthermore, the rigs must perform several tasks, such as drilling, evaluation, completion and workover, immersed in an environment full of uncertainties, especially in offshore wells. Therefore, these companies require to properly plan and schedule theirs fleet of rigs to ensure that the right rigs will be available in the right place at the right time with the lowest cost possible. Due to its importance, the problem of determining the rigs schedule, known as Rig Scheduling Problem, has received a lot of attention in the literature.

Nonetheless, as seen in the literature review from Section 3.1, most of these researches focus on less complex problems for planning workover tasks in onshore wells. Few studies apply to others important tasks such as drilling and completion and none of them consider realistic functions and constraints to be applied in real companies. Aiming to fill this gap in the literature and support companies in their decision making process, this dissertation has approached a real rig scheduling problem in a major multinational of the Oil & Gas sector that needs to contract a large number of oil rigs to perform several of the offshore E&P activities, such as drilling, evaluation and completion.

In short, this case study is an offshore rig scheduling problem in which the goal is to find an optimal rigs fleet size and schedule that minimizes the company’s budget (a cost function that takes into account the rigs fleet size, idleness and its use) considering some precedence constraints, time windows and others particularity of the studied company. This problem can be represented as a $P_m \left| \bar{r}_j \bar{d}_j prec \right| nonreg$ scheduling problem, *i.e.*, the scheduling of identical machines in parallel with strict release and due dates, as well as precedence rules, whose objective is to minimize the budget, a non-regular objective function equal to the weighted sum of idleness time, number of machines and activities durations.

For the purpose of solving this problem, two mathematical models – Model 1 (that minimizes the rigs fleet) and Model 2 (that minimizes the rigs budget) – and some local search algorithms were formulated. The proposed solution method was to apply the combination of the mathematical models with the heuristics in a matheuristic algorithm, a new approach for the rig scheduling problem. A purely exact approach was also tested, but the results suggest a large difficult of using it in practical and large instances.

In order to deal with the complexity of the problem, the mathematical models were tested with two possible time horizon unit: days and weeks. The mathematical models using days represented the real schedule and, therefore, they required much more computational effort, as verified in the computational experiments. On the other hand, the models in weeks were a representation of an approximate and relaxed scenario and, as a result, they required much less executing time. In the small instance, the computational complexity was not enough to comprise the results and, consequently, the models in days achieved better results than the models in weeks. However, in the larger instance, the model in days needed much more computational effort and resulted in worse solutions and weaker bounds. Also to reduce the problem's complexity, a mathematical modeling procedure was developed combining Model 1 and Model 2 aiming to reduce the search space of the problem. After the computational experiments, this mathematical programing procedure using the Model 1 to provide an initial solution for Model 2 allowed to reduce the problem complexity if compared of the use of the budget minimization model 2 purely, but it has still presented a difficult in large instances and using daily time units.

When comparing the results from the mathematical model 1 alone and the procedure involving the mathematical model 1 and 2, the mathematical model 1 required much less computational effort and as it minimizes the rigs fleet size, it minimizes the sum of the rigs contracts, which is an element of the budget function. Furthermore, the model tries to fit the maximum of tasks in the minimal fleet of rigs. As a result, the mathematical model 1 is able to find relatively good solutions in a short executing time and can generate strong initial solution for others methods. In contrast, the procedure for the mathematical model 2 requires much more computational effort and, in large instances, might not find feasible solutions due to the computer memory.

Several local searches were developed based on three different neighborhood structures of *insert* movement: *insert with fixed dates*, *insert with dates anticipation* and *insert with dates postponement*. The first movement type removes a set of tasks from a position in a rig and inserts them at the end of another existing rig or in a new rig, while keeping the same start time of each task being moved. The second tries to anticipate a set of task in a rig and then insert these tasks in another rig (an existing one or even a new one) or insert a block of tasks of another rig in the end of this rig whose tasks were anticipated. And the last one tries to postpone a block of task in a rig and then insert these postponed tasks in another rig (an existing one or even a new one) or insert a set of tasks of another rig in the end of this rig whose tasks were postponed. Variations of these movements changing their settings, search strategies and combining multiple methods in a variable neighborhood descent search were generated. A total of 156 variations of local search heuristics was tested for two instances using an initial solution from the mathematical model 1 with times horizon periods in weeks. All the developed algorithms succeeded to reduce the budget in millions of dollars, having most of them outperformed the mathematical models 1 and 2 in the large instance.

After a Pareto frontier analysis and the evaluation of budget progression through the executing time, the variable neighborhood search (*VND*) #114 that uses *insert with dates change Anticipation(After–In)–Postponement(After–From)* neighborhood structures with *best-improvement* search strategy was selected due to its great potential to efficiently improve the budget, having obtained powerful results for the two instance. This *VND* searches solutions in two neighborhood structures, one of *insert with dates anticipation* and other of *insert with dates postponement*, movements that together allow the search to perform unique schedules.

In addition, the proposed local search #114 was combined with the others mathematical models to verify the most suitable matheuristic approach to the problem. These results supported the capacity of the matheuristics to find schedules with a low budget. Once again, the mathematical models using daily time units found better results but needed much more computational effort. In the larger instances, the best-known solution were found by the matheuristics. The combination of mathematical model 1 minimizing the rigs fleet with the VND minimizing the budget resulted in an efficient method, finding near optimal solutions in a relatively low execution time, especially using the mathematical model #1 with the time unit in weeks. Due to its high performance and low complexity, this matheuristic using the mathematical model 1 in weeks to find an initial solution for the variable neighborhood search #114 is suggested for this rig scheduling problem as it is capable of finding strong solutions with a very low computational time, needing just some seconds (small instances) or minutes (large instances).

Some constructive greedy algorithms were also developed as an alternative method to generate initial solutions for the local search instead of the mathematical models and tested with the proposed variable neighborhood descent #114. Despite the extremely low computational efforts, the solution quality were still far away from those found by those using the matheuristic. Therefore, results indicate that this purely heuristic approach is not preferable to the proposed matheuristic method.

In short, the proposed matheuristic is a fast and realistic decision support tool that has potential to reduce millions (or even billions) of dollars for oil & gas companies, capable of find near optimal schedules with few computational efforts even for large instances where most exact methods are too complex and slow.

The success of the proposed matheuristic release new opportunities for futures applications of matheuristics and metaheuristics in rig scheduling problems. More computational experiments are required in others instances in order to determine the patterns in the different heuristics tested in this study. In addition, it is possible to test others more advanced algorithms in the heuristic stage, such as Variable Neighborhood Search (VNS), Iterated Local Search (ILS), Iterated Greedy Search (IGS), Tabu Search (TS) and Genetic Algorithms (GA). Another possibility is to try others neighborhood structures, constructive heuristics and local search variations to enhance the methods tested here.

The complexity of the problem forces to consider relaxed assumptions for the problem and hinders the effectiveness of the mathematical models. The matheuristic approach has emerged to fulfill this gap. Nonetheless, its results relies on the quality of the initial solution of the mathematical model. This dissertation results proved that the matheuristics results using the exact models in days were slightly better than the other models in weeks, but the computational effort was too much to justify its use. Therefore, the application of advantages linear programming techniques to reduce the mathematical models complexity trends to generates great gains for future researches.

The reduction of the problem's complexity through advanced mathematical programming or metaheuristics allows approaching more realistic scenarios and rig scheduling problems. Currently, there is a demand for new formulations considering heterogeneous fleets of rigs, machine eligibility, fleets availability constraints and variable costs over the planning horizon.

Furthermore, as the Oil & Gas companies are immersed in an environment full of risks, there is the need of rig scheduling models taking into account the uncertainties in tasks processing time, rigs rates, wells productions and tasks release dates.

8
References

ACCIOLY, R.; MARCELLINO, F. J. M.; KOBAYASHI, H. Uma aplicação da programação por restrições no escalonamento de atividades em poços de petróleo. In: *XXXIV SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Rio de Janeiro, RJ, Brazil, 2002.

AL GHARBI, S. H. *Drilling rig schedule optimization*. 2011. 97p. Thesis (Master of Science in Computer Science) – King Fahd University of Petroleum & Minerals, Saudi Arabia, 2011.

ALOISE, D. J.; ALOISE, D.; ROCHA, C. T. M.; RIBEIRO, C. C.; FILHO, J. C. R.; MOURA, L. S. S. Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, v. 154, n. 5, p. 695-702, 2006.

ALOISE, D.; NORONHA, T. F.; MAIA, R. S.; BITTENCOURT, V. G.; ALOISE, D. J. Heurísticas de colônia de formigas com path-relinking para o problema de otimização da alocação de sondas de produção terrestre–SPT. In: *XXXIV SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Rio de Janeiro, RJ, Brazil, 2002.

ALVES, V. R. F. M.; FERREIRA FILHO, V. J. M. Proposta de algoritmo genético para a solução do problema de roteamento e sequenciamento de sondas de manutenção. In: *XXXVIII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Goiânia, GO, Brazil, 2006, p. 1837-1848.

ARONOFSKY, J. S.; WILLIAMS, A. C. The use of linear programming and mathematical models in under-ground oil production. *Management Science,* v. 8, n. 4, p. 394-407, 1962.

BAKER, R. A primer of oilwell drilling: a basic text of oil and gas drilling, *Petroleum Extension Service,* v. 184, 1996.

BARNES, J. W.; BRENNAN, J. J.; KNAPP, R. M. Scheduling a backlog of oil well workovers. *Journal of Petroleum Technology*, v. 29, n. 12, p. 1651–1653, 1977.

BASSI, H. V.; FERREIRA FILHO, V. J. M.; BAHIENSE, L. Planning and scheduling a fleet of rigs using simulation-optimization. *Computers & Industrial Engineering,* v. 63, n. 4, p. 1074-1088, 2012.

BISSOLI, D. C. *Uma abordagem heurística para o problema de roteamento de sondas de intervenção bi-objetivo*. 2014. 108p. Dissertation (Master in Energy Engineering) – Universidade Federal do Espírito Santo (UFES), São Mateus, ES, Brazil, 2014.

BISSOLI, D. C.; CHAVES, G. L. D.; RIBEIRO, G. M. Drivers to the workover rig problem. *Journal of Petroleum Science and Engineering*, v. 139, p. 13-22, 2016.

BP, *BP statistical review of world energy*. London, United Kingdom, 2017a. Available at: <https://www.bp.com/content/dam/bp/en/corporate/pdf/energy-economics/statistical-review-2017/bp-statistical-review-of-world-energy-2017-full-report.pdf>. Accessed in: 2th February, 2018.

BP, *BP energy outlook*. London, United Kingdom, 2017b. Available at: <https://www.bp.com/en/global/corporate/energy-economics/energy-outlook.html>. Accessed in: 2th February, 2018.

CARILHO, L. M.; VILLAS BOAS, M. A. C. *Core: decision support system for oil rig scheduling*. 2016. Final project (Bachelor in Industrial Engineering) – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, RJ, Brazil, 2016.

COSTA, L. R. *Soluções para o problema de otimização de itinerário de sondas*. 2005. 117p. Dissertation (Master in Industrial Engineering) – Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brazil, 2005.

COSTA, L. R.; FERREIRA FILHO, V. J. M. Uma heurística para o problema do planejamento de itinerários de sondas em intervenções de poços de petróleo. In: *XXXVI SBPO - Simpósio Brasileiro De Pesquisa Operacional*, São João del Rei, MG, Brazil, 2004, p. 1844-1853.

COSTA, L. R.; FERREIRA FILHO, V. J. M. Uma heurística de montagem dinâmica para o problema de otimização de itinerários de sondas. In: *XXXVII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Gramado, RS, Brazil, 2005.

CULVER, G. Chapter 6: Drilling and well construction. In: CULVER, G. *Geothermal direct use engineering and design guidebook*. Geo-Heat Center, Oregon Institute of Technology. 1998. p. 129-164p

CURRIE, J. C.; NOVOTNAK, J. F.; AASBOEE, B. T.; KENNEDY, C. J. Optimized Reservoir Management Using Mixed Linear Programming. In: *Hydrocarbon Economics and Evaluation Symposium*, Dallas, TX, United States of America, 1997a, p. 235-241.

CURRIE, J. C.; NOVOTNAK, J. F; AASBOEE, B. T.; KENNEDY, C. J. Optimized reservoir management using mixed linear programming. *SPE Computer Applications*, v. 9, n. 4, p. 103-106, 1997b.

DE ANDRADE FILHO, A. C. B. Optimal scheduling of development in an oil field. 1994 112p. Dissertation (Master in Petroleum Engineering) - Stanford University, California, United States of America, 1994.

DELL'AMICO, M.; IORI, M.; MARTELLO, S.; MONACI, M. Heuristic and exact algorithms for the identical parallel machine scheduling problem. *INFORMS Journal on Computing*, v. 20, n. 3, p. 333-344, 2008.

DEVOLD, H. *Oil and gas production handbook: an introduction to oil and gas production, transport, refining and petrochemical industry*. 3. ed. Oslo: ABB Oil and Gas, 2013.

DONG, X.; HUANG, H.; CHEN, P. An iterated local search algorithm for the permutation flowshop problem with flowtime criterion. *Computers & Operations Research*, v. 36, n.5, p. 1664-1669, 2009.

DOURO, R. F.; LORENZONI, L. L. A Um algoritmo genético-2opt aplicado ao problema de otimização de itinerário de sondas de produção terrestre. In: *XLI SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Porto Seguro, BA, Brazil, 2009. p. 2121-2132

DUHAMEL, C.; SANTOS, A. C.; GUEDES, L. M. Models and hybrid methods for the onshore wells maintenance problem. *Computers & Operations Research*, v. 39, n. 12, p. 2944-2953, 2012

EAGLE, K. Using simulated annealing to schedule oil field drilling rigs. *Interfaces*, v. 26, n. 6, p. 35-43, 1996.

FALEX, A. *Planejamento da frota de sondas para atendimento de uma campanha de perfuração de um campo*, 2009, 60p. Thesis (Doctoral in Petroleum Engineering) – Universidade Federal Do Rio De Janeiro (UFRJ), Rio de Janeiro, RJ, Brazil, 2009.

FERREIRA FILHO, V.; HAMACHER, S. *Aplicações de pesquisa operacional na indústria internacional de petróleo:* modelagem e solução para problemas da exploração a distribuição. Rio de Janeiro, Campus, Elsevier Brasil, 2015. 432p.

FLAGER, F. A method to optimize onshore drilling rig fleet size and schedule considering both reservoir management and operational objectives. *Journal of Project Production Management. Project Production Institute*, v. 1, 2016.

GONÇALVES, R. K. *Otimização de alocação de sondas para exploração off-shore de hidrocarbonetos por algoritmos genéticos*, 2009, 42p. Dissertation (Specialization in Business Intelligence) – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, RJ, Brazil, 2009.

GOUVÊA, E. F; GOLDBARG, M. C.; COSTA, W. E. Algoritmos evolucionários na solução do problema de otimização do emprego de sondas de produção em poços de petróleo In: *XXXIV SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Rio de Janeiro, RJ, Brazil, 2002.

GRAHAM, R. L; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, v. 5, p. 287-326, 1979.

GUPTA, V.; GROSSMANN, I. E. An efficient multiperiod MINLP model for optimal planning of offshore oil and gas field infrastructure. *Industrial & Engineering Chemistry Research*, v. 51, n. 19, p. 6823-6840, 2012.

HASLE, G.; HAUT, R.; JOHANSEN, B.; ØLBERG, T. Well activity scheduling - an application of constraint reasoning. *Artificial Intelligence in the Petroleum Industry: Symbolic and Computational Applications*, v. 2 p. 209-228, 1996

IFP SCHOOL. *What are the Main Steps of an Oil or Gas Field Development Project?*. IFP School, The Graduate School for Energy and Transportation Professions. 2015. Available at: <http://www.ifp-school.com/upload/docs/application/pdf/2015-2/3_main_steps_oil_gas_field_development.pdf> Accessed in: 2th February, 2018.

IHS MARKIT. *Petrodata Offshore Rig Day Rate Trends*. 2017. Available at: <https://www.ihs.com/products/oil-gas-drilling-rigs-offshore-day-rates.html> Accessed in: 2th February, 2018.

IOM3. *An Introduction to Oil & Gas Drilling and Well Operations*. Educational Material from the IOM3 Oil and Division. IOM3, The Institute of Material, Minerals and Mining United Kingdom, 2015. Available at: < http://www.iom3.org/sites/default/files/IOM3%20Introduction%20to%20Oil%20%20Gas%20Drilling%20and%20Well%20Operations%20Nov%202016.pdf> Accessed in: 2th February, 2018.

IRGENS, M.; LAVENUE, W. L. Use of advanced optimization techniques to manage a complex drilling schedule. In: *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, Anaheim, CA, United States of America, 2007.

IRGENS, M.; GUZMAN, R. P.; STAMATOPOULOS, J. G.; JACKSON, K. Optimization for operational decision support: the rig management case. In: *SPE Annual Technical Conference and Exhibition*, Denver, Colorado, USA, 2008.

IYER, R. R.; GROSSMANN, I. E.; VASANTHARAJAN, S.; CULLICK, A. S. Optimal planning and scheduling of offshore oil field infrastructure investment and operations. *Industrial & Engineering Chemistry Research*, v. 37, n. 4, p. 1380-1397, 1998.

KAISER, M. J.; SNYDER, B. The five offshore drilling rig markets. *Marine Police*, v. 39, p. 201-214, 2013.

LASRADO, V. Workover Rig Scheduling Using Reservoir Simulation. In: *Intelligent Energy Conference and Exhibition*. Society of Petroleum Engineers, Amsterdam, Netherlands, 2008.

LITVAK, M. L.; GANE, B. R; WILLIAMS, G.; MANSFIELD, M.; ANGERT, P. F.; MACDONALD, C. J.; WALKER, G. J. Field Development Optimization Technology. In: *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, Houston, TX, United States of America, 2007.

LORENZONI, L. L.; POLYCARPO, W. M. Scatter search aplicado ao problema de otimização de itinerários de sondas de produção terrestre. In: *XXX ENEGEP - National Production Engineering Meeting,* São Carlos, SP, Brazil, 2010.

LOTOV, A. V.; MIETTINEN, K. Visualizing the Pareto frontier. In: BRANKE, J, DEB, K.; MIETTINEN, K.; SŁOWIŃSKI, R. (Eds.), *Multiobjective Optimization,* Lecture Notes in Computer Science, Springer, Berlin, v. 5252, p. 213–243, 2008.

MAIA, R. S.; DE MEDEIROS GONZAGA, C. S.; DE LIMA JÚNIOR, F. C.; BITTENCOURT, V. G. Optimization of interventions in oil wells by onshore rigs. In: *XXXIV SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Rio de Janeiro, RJ, Brazil, 2002.

MANNING, M. Offshore oil production in deepwater and ultra-deepwater is increasing. *Today in Energy*. U.S. Energy Information Administration – Independent Statistics & Analysis, 2016. Available at: <https://www.eia.gov/todayinenergy/detail.php?id=28552> Accessed in: 2th February, 2018.

MARQUES, L. C.; DE PAULA MACHADO, F. A. P.; OLIVEIRA, F.; HAMACHER, S. Sizing and scheduling resources: a decision support system applied to oil rigs scheduling. In: *XLVI SBPO - Simpósio Brasileiro De Pesquisa Operacional,* Salvador, BA, Brazil, 2014. p. 2538-2547.

MAURI, G. R.; LORENA, L. A. N. Customers' satisfaction in a dial-a-ride problem. *IEEE Intelligent Transportation Systems Magazine*, v. 1, n. 3, p. 6–14, 2009.

MAZZINI, F. F.; DE MELO, R.; ACCIOLY, S.; VASCONCELOS, R. V. J. Dimensionamento de equipamentos críticos da cadeia de suprimento da perfuração e completação de poços. In: *XLII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Bento Gonçalves, RS, Brazil, 2010.

MONEMI, R. N.; DANACH, K.; KHALIL, W.; GELAREH, S.; LIMA JR, F. C.; ALOISE, D. J. Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. *Expert Systems with Applications*, v. 4, n. 9, p. 4493-4505, 2015.

NEVES, T. A. *Heurísticas com memória adaptativa aplicadas ao problema de roteamento e scheduling de sondas de manutenção*. 2007, 81p. Dissertation (Master in Computer Science) – Universidade Federal Fluminense Federal (UFF), Niterói, RJ, Brazil, 2007.

NEVES, T. A.; OCHI, L. S. GRASP com memória adaptativa aplicado ao problema de roteamento e scheduling de sondas de manutenção. *XXVII Congresso da SBC – Sociedade Brasileira de Computação*, Belém do Pará, PA, Brazil. 2007.

NEVES, M. L.; LAZZARINI, A. L.; BRANDÃO, A. L. S.; DE CASTRO, J. F. T. Alocação de equipamentos críticos em projetos de produção de petróleo offshore. In: *Annals of the XVI CLAIO - XLIV SBPO - Workshop LIA-SGT*, Rio de Janeiro, RJ, Brazil, 2012.

NUMMI, E. *Energy outlook may influence unconventional oil exploration*. ThermoFischer Scientific, 2017. Available at: <https://www.thermofisher.com/blog/mining/energy-outlook-may-influence-unconventional-oil-exploration/> Accessed in: 2th February, 2018.

OFFSHORE CENTER DANMARK. *OffshoreBook: An Introduction to the Offshore Industry*, 2010. Available at: < http://www.offshorecenter.dk/log/bibliotek/OffshoreBook2010.pdf > Accessed in: 2th February, 2018.

OLIVEIRA, E. D.; PAGOTO, F. B.; SILVA, F. T.; LORENZON, L. L. Scatter search aplicado ao problema de otimização da alocação de sondas de produção em poços de petróleo. In: *XXVII ENEGEP – Encontro Nacional de Estudantes de Engenharia de Produção*, Foz do Iguaçu, PR, Brazil, 2007.

ONWUNALU, J. E.; LITVAK, M. L.; DURLOFSKY, L. J.; AZIZ, K. Application of Statistical Proxies to Speed Up Field Development Optimization Procedures. In: *Proceedings of the Abu Dhabi International Petroleum Exhibition and Conference*. Society of Petroleum Engineers, Abu Dhabi, United Arab Emirates, 2008.

OSMUNDSEN, P.; ROLL, K. H.; TVETERÅS, R. Exploration Drilling Productivity at the Norwegian Shelf. *Journal of Petroleum Science and Engineering*, v. 73, p. 122-128, 2010.

PACHECO, A. V. F. *Métodos de solução para o problema da alocação de sondas a poços de petróleo*. 2011, 68p. Dissertation (Bachelor in Industrial Engineering) – Universidade Federal do Espírito Santo (UFES), São Mateus, ES, Brazil, 2011.

PACHECO, A. V. F.; DIAS FILHO, A. C. T.; RIBEIRO, G. M. Uma heurística para o problema da alocação de sondas de produção em poços de petróleo. In: *XXIX ENEGEP – Encontro Nacional de Estudantes de Engenharia de Produção*, Salvador, BA, Brazil, 2009.

PACHECO, A. V. F.; RIBEIRO, G. M.; MAURI, G. R. Novas soluções para o problema da alocação de sondas de produção a poços de petróleo com um Grasp+path-relinking. In: *XLII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Bento Gonçalves, RS, Brazil, 2010, p. 3129-3138.

PACHECO, A. V. F; RIBEIRO, G. M.; MAURI, G. R. A GRASP with Path-Relinking for the Workover Rig Scheduling Problem. *Nature-Inspired Computing Design, Development, and Applications*, p. 90-103, 2012.

PAIVA, R. O. *Otimização do itinerario de sondas de intervenção*. 1997, 115p. Dissertation (Master in Petroleum Engineering) – Universidade Estadual de Campinas (Unicamp), Campinas, SP, Brazil, 1997.

PAIVA, R. O.; SCHIOZER, D. J.; BORDALO, S. N. Optimizing the itinerary of workover rigs. In: *16th World Petroleum Congress*, Calgary, Canada, p. 103-109, 2000.

PAMPLOMA, N. Sondas de perfuração paradas custaram R$ 1,1 bilhão à Petrobras, *Folha de São Paulo*, may, 2016. Available at: <http://www1.folha.uol.com.br/mercado/2016/05/1771018-sondas-de-perfuracao-paradas-custaram-r-11-bilhao-a-petrobras.shtml>. Accessed in: 5th February, 2018.

PELIZARO, C. *Dimensionamento de frota em uma empresa de petróleo integrada.* 2008. 95p. Dissertation (Master in Industrial Engineering) – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, RJ, Brazil, 2008.

PEREIRA, R. A. *Scheduling of development activities of oil wells: GRASP*. 2005. 92p. Dissertation (Master of Science in Computer Science) – Instituto de Computação, Universidade Estadual de Campinas (Unicamp), Campinas, São Paulo, Brazil, 2005.

PÉREZ, M.; OLIVEIRA, F.; HAMACHER, S. A New Mathematical Model for the Workover Rig Scheduling Problem. *Pesquisa Operacional*, v. 36, n. 2, p. 241-257, 2016.

PETROBRAS *Comparativo Entre Os Diferentes Tipos De Plataformas*. 2014. Available at: <http://www.petrobras.com.br/infograficos/tipos-de-platformas/desktop/index.html>. Accessed in: 5th February, 2018.

PINEDO, M. *Scheduling*. New York: Springer, 2008, 671p.

PITTMAN, J. Computer Speeds Offshore Well Planning. *Rig Scheduling. Oil & Gas Journal*, v. 83, n. 48, p. 84, 1985.

RAIDL, G. R.; PUCHINGER, J. Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization. *Hybrid metaheuristics*, v. 114, p. 31-62, 2008.

REID, D.; YOST, T.; RUSSELL, I.; CHEUNG, T. O. Avoiding the Money Pit: The Industrialization of Delivering Complex Drilling Facilities. In: *Drilling Conference and Exhibition*, Society of Petroleum Engineers, Fort Worth, TX, United States of America, 2016.

RIBEIRO, G. M.; DESAULNIERS, G.; DESROSIERS, J. A Branch-Price-and-Cut Algorithm for the Workover Rig Routing Problem. *Computers & Operations Research*, v. 39, n. 12, p. 3305-3315, 2012b.

RIBEIRO, G. M.; DESAULNIERS, G.; DESROSIERS, J.; VIDAL, T.; VIEIRA, B. S. Efficient Heuristics for the Workover Rig Routing Problem with a Heterogeneous Fleet and a Finite Horizon. *Journal of Heuristics*, v. 20, n. 6, p. 677-708, 2014.

RIBEIRO, G. M.; LAPORTE, G.; MAURI, G. R. A Comparison of Three Metaheuristics for the Workover Rig Routing Problem. *European Journal of Operational Research*, v. 220, n. 1, p. 28-36, 2012a.

RIBEIRO, G. M.; MAURI, G. R.; LORENA, L. A. N. A Simple and Robust Simulated Annealing Algorithm for Scheduling Workover Rigs on Onshore Oil Fields. *Computers & Industrial Engineering*, v. 60, n. 4, p. 519-526, 2011.

ROCHA, C. T. M.; ALOISE, D.; ALOISE, D. J.; MELO, J. D. Heurísticas paralelas de busca em vizinhança variável para o problema de otimização do emprego de sondas de produção terrestre. In: *XXXV SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Natal, RN, Brazil, 2003.

SANTOS, I. M.; CARRILHO, L. M.; OLIVEIRA, F. L. C.; ANDRADE, T.; RIBAS, G. Offshore oil rig scheduling simulation: a multi-perspective approach. In: *XLIX SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Blumenau, SC, Brazil, 2017.

SERRA, T.; NISHIOKA, G.; MARCELLINO, F. J. A constraint-based scheduling of offshore well development activities with inventory management. In: *XLIII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Ubatuba, SP, Brazil, 2011

SERRA, T.; NISHIOKA, G.; MARCELLINO, F. J. Integrated Project Selection and Resource Scheduling of Offshore Oil Well Developments: An Evaluation of CP Models and Heuristic Assumptions. In: *Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, 22th International Conference on Automated Planning and Scheduling, Atibaia, SP, Brazil, 2012a, p. 51-58.

SERRA, T.; NISHIOKA, G.; MARCELLINO, F. J. The Offshore Resources Scheduling Problem: Detailing a Constraint Programming Approach. In: Milano M. (Eds) *Principles and Practice of Constraint Programming*. Lecture Notes in Computer Science, v. 7514. Springer, Berlin, Heidelberg, 2012b, p. 823-839.

SHAKHSI-NIAEI, M.; IRANMANESH, S. H; TORABI, S. A. A Review of Mathematical Optimization Applications in Oil-and-Gas Upstream & Midstream Management. *International Journal of Energy and Statistics*, v. 1, n. 2, p. 143-154, 2013.

SILVA, L. M. R.; SANTOS, A.M.P.; GUEDES SOARES, C. A Mixed Integer Formulation for the Offshore Rig Scheduling Problem. In: *3rd International*

Conference on Maritime Technology and Engineering (MARTECH), Lisbon, 2016, p. 1005-1011.

SUSLICK, S. B.; SCHIOZER, D. J. Risk analysis applied to petroleum exploration and production: an overview. *Journal of Petroleum Science and Engineering*, v. 44, n. 1, p. 1-9, 2004.

SUSLICK, S. B.; SCHIOZER, D. J.; RODRIGUEZ M. R. Uncertainty and risk analysis in petroleum exploration and production. *Terræ*, v. 6, n. 1, p. 30-41, 2009.

TALBI, E. F. *Metaheuristics:* from design to implementation. New York, John Wiley & Sons, 2009, v. 74.

TARHAN, B.; GROSSMANN, I. E.; GOEL, V. Stochastic Programming Approach for the Planning of Offshore Oil or Gas Field Infrastructure Under Decision-dependent Uncertainty. *Ind. Eng. Chem. Res*; v. 48, n. 6, p. 3078–3097, 2009.

TRINDADE, V. A.; OCHI, L. S. Proposta e avaliação experimental de heurísticas GRASP para um problema de escalonamento de veículos. In: *XXXVI SBPO - Simpósio Brasileiro De Pesquisa Operacional*, São João Del Ray, MG, Brazil, 2004.

TRINDADE, V. A.; OCHI, L. S. Hybrid adaptive memory programming using GRASP and path relinking for the scheduling workover rigs for onshore oil production. In: *Proceedings of the 5th International Conference on Hybrid Intelligent Systems*, Rio de Janeiro, RJ, Brazil, 2005.

VASCONCELLOS, R.; FERREIRA FILHO, V. J. M. F. Algoritmo genético para o problema de scheduling de projetos com restrição de recurso: uma aplicação em operações em poços de petróleo. In: *XXXVIII SBPO - Simpósio Brasileiro De Pesquisa Operacional*, Goiânia, GO, Brazil, 2006.

Appendix I: Neighborhood Structure Algorithms

The procedure to search for these neighbor solutions using the *insert with fixed dates in an existing rig* movement is simplified in Algorithm 3.

Algorithm 3. *Insert with fixed dates in an existing rig* neighborhood search

```
1:  $s, s^* \leftarrow s_0$; //current solution//
2:  **for each** $i$ **in** $M$ **do**:
3:      **for each** $i'$ **in** $M\backslash\{i\}$ **do**:
4:          **for each** $j$ **in** $blocksRig(i)$ **do**:
5:              $tempList \leftarrow i.getTasksFrom(j)$;
6:              $i'.receiveTasks(tempList)$;
7:              **if** $viableSolution(s)$ **then**:
8:                  $s^* \leftarrow acceptanceCriterion(s)$;
9:              **endif**;
10:             $s \leftarrow s_0$;
11:         **endfor**;
12:     **endfor**;
13: **endfor**;
14: **return** $s^*$;
```

In order to help the understanding of this neighborhood structure, some steps were simplified, or even omitted, in Algorithm 3. When searching solutions in the *insert with fixed dates in an existing rig* neighborhood, the heuristic tests for each machine (rig) i (loop from line 2 to 13) if there is another machine i' (loop from line 3 to 12) in which some block of tasks (composed by any block of activity from j to end of the machine i list – loop from line 4 to 11) could be reallocated in the end of the other machine i' (line 6), respecting the constraints of the rig scheduling problem (line 7) and without changing any allocation date. Any viable solution found is tested using an acceptance criterion (line 8) that selects the new neighbor solution to be used as current solution (line 14). In short, this neighborhood is composed by any viable solution moving a set of tasks from the end of a rig to another rig's end keeping the same allocation dates.

Next, the main steps of the procedure to search for neighbor solutions using the *insert with fixed dates in a new rig* movement are illustrated in Algorithm 4.

Algorithm 4. *Insert with fixed dates in a new rig* neighborhood search

```
1:   $s, s^* \leftarrow s_0$; //current solution//
2:   for each $i$ in $M$ do:
3:       for each $j$ in $blocksRig(i)$ do:
4:           $tempList \leftarrow i.getTasksFrom(j)$;
5:           $M \leftarrow M \cup i'$; //adds a new rig into the fleet//
6:           $i'.receiveTasks(tempList)$;
7:           $s^* \leftarrow acceptanceCriterion(s)$;
8:           $s \leftarrow s_0$;
9:       endfor;
10:  endfor;
11:  return $s^*$;
```

In the *insert with fixed dates in a new rig* neighborhood structure, the heuristic tests for each machine (rig) i (loop from line 2 to 10) the impact on the objective function after moving any set of scheduled tasks (loop from line 3 to 9) to a new rig. In these loops, such blocks are composed by all the tasks from block j in the end of the machine i list (line 4). A new rig i' is added to the fleet at line 5 and the tasks selected ($tempList$) are reallocated in i' (line 6), without changing any allocation date. As this new rig was initially empty, there is no need to check the viability of the solution, differently from the previous method. All the neighbor solutions found are tested using an acceptance criterion (line 7) that selects the new neighbor solution to be used as current solution (line 11). In short, this neighborhood is composed by any solution moving a set of tasks from the end of a rig to a new rig maintaining the same allocation dates. As to *insert with dates anticipation* movement, they can be described in two stages: *anticipation* (*after* or *before*) followed by *insert* (*in* or *from*). Algorithm 5 presents the neighborhood search using any type of *insert with dates anticipation* movement.

Algorithm 5. *Insert with dates anticipation* neighborhood search

```
1:   $s, s^* \leftarrow s_0$; //current solution receives initial solution//
2:   $strategy \leftarrow defineNeighborSelectionStrategy()$;
3:   for each $i$ in $M$ do:
4:       for each $j$ in $blocksRig(i)$ do:
5:           $i.anticipationStep(j)$;
6:           $s.insertStep(i,\ strategy)$;
7:           $s^* \leftarrow acceptanceCriterion(s)$;
8:           $s \leftarrow s_0$;
9:       endfor;
10:  endfor;
11:  return $s^*$;
```

In Algorithm 5, the heuristic tests for each machine (rig) i (loop from line 3 to 10) the impact on the objective function after performing an anticipation of tasks in the machine i followed by a instertion movement according to a block of task j from rig i (loop from line 4 to 9). As mentioned earlier, the anticipation step performed in machine i at line 5 can be on the tasks *before* or *after* a position j and is followed by the insertion of tasks *in* or *from* the machine i (line 6). Furthermore, it is important to notice that the $insertStep$ (line 6) searches some tasks *from* the anticipated rig to be inserted in the end of another rig (*insert from*) or some others tasks from the end of another rig to be inserted *in* the end of the anticipated rig (*insert in*) with no obligation to perform an insert move, determined according to a neighbor selection strategy (line 2). If the neighbor solution satisfies an acceptance criterion, it is store as the new best solution so far (line 7). At the end of each iteration of the loop, the current solution is restored (line 8). Finally, at line 11, the neighborhood search returns the best overall solution found (line 11).

Last, the *insert with dates postponement* movements can be described in two stages: *postponement* (*after* or *before*) followed by *insert* (*in* or *from*). Algorithm 6 presents the neighborhood search using any type of *insert with dates postponement* movement.

Algorithm 6. *Insert with dates postponement* neighborhood search

```
 1:  $s, s^* \leftarrow s_0$; //current solution receives initial solution//
 2:  $strategy \leftarrow defineNeighborSelectionStrategy()$;
 3:  for each $i$ in $M$ do:
 4:      for each $j$ in $blocksRig(i)$ do:
 5:          $i.postponementStep(j)$;
 6:          $s.insertStep(i,\ strategy)$;
 7:          $s^* \leftarrow acceptanceCriterion(s)$;
 8:          $s \leftarrow s_0$;
 9:      endfor;
10:  endfor;
11:  return $s^*$;
```

When searching solutions in the *insert with dates postponement* neighborhoods, the heuristic tests for each machine (rig) i (loop from line 3 to 10) the impact on the objective function after delaying tasks in the machine i followed by a instertion movement according to a block of task j from rig i (loop from line 4 to 9). As mentioned earlier, the *postponement* step performed in machine i at line 5 can be on the tasks *before* or *after* a position j and is followed by the insertion of tasks in or $from$ the machine i (line 6). Furthermore, it is important to notice that the $insertStep$ (line 6) searches some tasks *from* the delayed rig to be inserted in the end of another rig (*insert from*) or some others tasks from the end of another rig to be inserted *in* the end of the delayed rig (*insert in*) with no obligation to perform an insert move, determined according to a neighbor selection strategy (line 2). If the neighbor solution satisfies an acceptance criterion, it is store as the new optimal solution (line 7). At end of each iteration of the loop, the current solution is restored (line 8). Finally, at line 11, the neighborhood search returns the best solution found (line 11).

Appendix II: Instances Description

The following table contains important details about the small instance:

Table 20. Description of the Small Instance (Source: the author).

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Succesors List
1	9	33	3	17	8555	8751		2;3;
2	9	33	3	82	8572	8833	1;	3;
3	9	33	3	45	8654	8878	1;2;	
4	61	34	3	17	8778	8974		5;6;
5	61	34	3	72	8795	9046	4;	6;
6	61	34	3	45	8867	9091	4;5;	
7	10	35	3	17	8699	8895		8;9;
8	10	35	3	72	8716	8967	7;	9;
9	10	35	3	45	8788	9012	7;8;	
10	12	36	3	17	9012	9208		11;12;
11	12	36	3	79	9029	9287	10;	12;
12	12	36	3	46	9108	9333	10;11;	
13	55	37	3	62	7975	8216		
14	58	38	3	56	8395	8630		
15	56	39	3	17	8037	8233		16;17;
16	56	39	3	116	8054	8349	15;	17;
17	56	39	3	68	8170	8417	15;16;	
18	7	40	3	17	8184	8380		19;20;
19	7	40	3	107	8201	8487	18;	20;
20	7	40	3	67	8308	8554	18;19;	
21	8	41	3	17	8375	8571		22;23;
22	8	41	3	96	8392	8667	21;	23;
23	8	41	3	67	8488	8734	21;22;	
24	60	42	3	17	8599	8795		25;26;
25	60	42	3	105	8616	8900	24;	26;
26	60	42	3	57	8721	8957	24;25;	
27	11	43	3	17	8833	9029		28;29;
28	11	43	3	105	8850	9134	27;	29;
29	11	43	3	57	8955	9191	27;28;	
30	62	44	3	17	8912	9108		31;32;
31	62	44	3	105	8929	9213	30;	32;
32	62	44	3	57	9034	9270	30;31;	
33	23	45	4	63	8171	8413		
34	24	46	4	17	8234	8430		35;36;
35	24	46	4	93	8251	8523	34;	36;
36	24	46	4	55	8344	8578	34;35;	
37	39	47	4	17	8453	8649		38;39;
38	39	47	4	93	8470	8742	37;	39;
39	39	47	4	55	8563	8797	37;38;	
40	26	48	4	17	8564	8760		41;42;
41	26	48	4	93	8581	8853	40;	42;
42	26	48	4	55	8674	8908	40;41;	
43	41	49	4	17	8783	8979		44;45;
44	41	49	4	93	8800	9072	43;	45;
45	41	49	4	55	8893	9127	43;44;	
46	28	50	4	17	8894	9090		47;48;
47	28	50	4	93	8911	9183	46;	48;
48	28	50	4	55	9004	9238	46;47;	
49	42	51	4	17	8948	9144		50;51;
50	42	51	4	93	8965	9237	49;	51;
51	42	51	4	55	9058	9292	49;50;	
52	30	52	4	17	9224	9420		53;54;
53	30	52	4	93	9241	9513	52;	54;
54	30	52	4	55	9334	9568	52;53;	
55	44	53	4	17	9278	9474		56;57;
56	44	53	4	93	9295	9567	55;	57;
57	44	53	4	55	9388	9622	55;56;	

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Succesors List
58	37	54	4	63	8225	8467		
59	38	55	4	17	8288	8484		60;61;
60	38	55	4	93	8305	8577	59;	61;
61	38	55	4	55	8398	8632	59;60;	
62	25	56	4	17	8399	8595		63;64;
63	25	56	4	93	8416	8688	62;	64;
64	25	56	4	55	8509	8743	62;63;	
65	27	57	4	17	8729	8925		66;67;
66	27	57	4	93	8746	9018	65;	67;
67	27	57	4	55	8839	9073	65;66;	
68	40	58	4	17	8618	8814		69;70;
69	40	58	4	93	8635	8907	68;	70;
70	40	58	4	55	8728	8962	68;69;	
71	29	59	4	17	9059	9255		72;73;
72	29	59	4	93	9076	9348	71;	73;
73	29	59	4	55	9169	9403	71;72;	
74	43	60	4	17	9113	9309		75;76;
75	43	60	4	93	9130	9402	74;	76;
76	43	60	4	55	9223	9457	74;75;	
77	1	3	1	17	7215	7411		78;79;
78	1	3	1	62	7232	7473	77;	79;
79	1	3	1	29	7294	7502	77;78;	
80	54	5	1	17	7754	7950		81;82;
81	54	5	1	50	7771	8000	80;	82;
82	54	5	1	23	7821	8023	80;81;	
83	50	6	1	53	7344	7576		
84	16	7	1	78	10148	10405		85;
85	16	7	1	38	10226	10443	84;	
86	5	8	1	13	7696	7888		87;88;
87	5	8	1	73	7709	7961	86;	88;
88	5	8	1	23	7782	7984	86;87;	
89	3	9	1	14	7459	7652		90;91;
90	3	9	1	86	7473	7738	89;	91;
91	3	9	1	26	7559	7764	89;90;	
92	48	10	1	31	7248	7458	93;94;	
93	48	10	1	17	7160	7356		94;92;
94	48	10	1	71	7177	7427	93;	92;
95	53	1	1	17	7644	7840		96;97;
96	53	1	1	70	7661	7910	95;	97;
97	53	1	1	23	7731	7933	95;96;	
98	15	14	1	68	10042	10289		99;
99	15	14	1	38	10110	10327	98;	
100	49	11	1	65	7279	7523		
101	52	61	1	17	7534	7730		102;103;
102	52	61	1	58	7551	7788	101;	103;
103	52	61	1	35	7609	7823	101;102;	
104	2	4	1	17	7323	7519		105;106;
105	2	4	1	66	7340	7585	104;	106;
106	2	4	1	53	7406	7638	104;105;	
107	51	12	1	17	7397	7593		108;109;
108	51	12	1	67	7414	7660	107;	109;
109	51	12	1	53	7481	7713	107;108;	
110	13	13	1	74	9787	10040		111;
111	13	13	1	53	9861	10093	110;	
112	4	2	1	13	7585	7777		113;114;
113	4	2	1	63	7598	7840	112;	114;
114	4	2	1	35	7661	7875	112;113;	
115	14	15	1	75	9914	10168		116;
116	14	15	1	53	9989	10221	115;	
117	18	16	2	57	7536	7772		
118	35	17	2	17	7938	8134		119;120;
119	35	17	2	73	7955	8207	118;	120;
120	35	17	2	45	8028	8252	118;119;	
121	32	18	2	17	7508	7704		122;123;
122	32	18	2	75	7525	7779	121;	123;
123	32	18	2	37	7600	7816	121;122;	
124	47	19	2	90	10286	10555		125;
125	47	19	2	45	10376	10600	124;	
126	33	20	2	17	7637	7833		127;128;
127	33	20	2	75	7654	7908	126;	128;
128	33	20	2	37	7729	7945	126;127;	
129	19	21	2	52	7593	7824		
130	36	22	2	17	8073	8269		131;132;

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Succesors List
131	36	22	2	73	8090	8342	130;	132;
132	36	22	2	45	8163	8387	130;131;	
133	22	23	2	17	8029	8225		134;135;
134	22	23	2	81	8046	8306	133;	135;
135	22	23	2	46	8127	8352	133;134;	
136	17	62	2	17	7363	7559		137;138;
137	17	62	2	100	7380	7659	136;	138;
138	17	62	2	56	7480	7715	136;137;	
139	31	24	2	17	7328	7524		140;141;
140	31	24	2	97	7345	7621	139;	141;
141	31	24	2	66	7442	7687	139;140;	
142	34	25	2	17	7766	7962		143;144;
143	34	25	2	97	7783	8059	142;	144;
144	34	25	2	58	7880	8117	142;143;	
145	20	26	2	17	7645	7841		146;147;
146	20	26	2	108	7662	7949	145;	147;
147	20	26	2	68	7770	8017	145;146;	
148	21	27	2	17	7838	8034		149;150;
149	21	27	2	106	7855	8140	148;	150;
150	21	27	2	68	7961	8208	148;149;	
151	46	28	2	123	10106	10408		152;
152	46	28	2	57	10229	10465	151;	
153	45	29	2	123	9926	10228		154;
154	45	29	2	57	10049	10285	153;	
155	6	30	3	17	8036	8232		156;157;
156	6	30	3	82	8053	8314	155;	157;
157	6	30	3	49	8135	8363	155;156;	
158	57	31	3	17	8238	8434		159;160;
159	57	31	3	90	8255	8524	158;	160;
160	57	31	3	50	8345	8574	158;159;	
161	59	32	3	17	8451	8647		162;163;
162	59	32	3	82	8468	8729	161;	163;
163	59	32	3	49	8550	8778	161;162;	

As to the large instance, the next table contains important details about its tasks:

Table 21 - Description of the Large Instance (Source: the author).

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
1	35	58	8	31	6437	6647		
2	149	3	3	47	6036	6262		
3	149	3	3	30	6083	6292		
4	1	4	3	42	6041	6262	100;	
5	1	4	3	38	6083	6300	100;	
6	2	5	3	29	6121	6329		3;
7	127	6	3	42	6126	6347	125;126;	
8	127	6	3	27	6168	6374	125;126;	
9	3	7	3	42	6150	6371	2;2;	
10	3	7	3	31	6192	6402	2;2;	
11	152	8	3	61	6198	6438	150;151;	
12	152	8	3	27	6259	6465	150;151;	
13	62	9	3	47	6228	6454	61;	
14	62	9	3	30	6275	6484	61;	
15	63	10	3	48	6305	6532		64;
16	63	10	3	17	6353	6549		64;
17	153	11	3	54	6320	6553		154;155;
18	153	11	3	34	6374	6587		154;155;
19	154	12	3	49	6408	6636	153;	155;
20	154	12	3	27	6457	6663	153;	155;
21	170	13	4	13	6025	6230	169;	171;
22	172	13	4	55	6156	6403		173;
23	172	13	4	39	6211	6442		173;
24	171	14	4	75	6038	6305	169;170;	
25	171	14	4	43	6113	6348	169;170;	
26	148	15	4	63	6057	6299	146;147;	
27	148	15	4	44	6120	6343	146;147;	
28	94	16	4	81	6178	6445		
29	94	16	4	48	6259	6493		

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
30	173	17	4	69	6250	6511	172;	
31	173	17	4	47	6319	6558	172;	
32	95	18	4	36	6307	6529		96;
33	96	19	4	43	6343	6572	95;	
34	174	20	4	38	6366	6596		175;
35	97	21	4	44	6386	6616		
36	98	22	4	73	6430	6689		
37	98	22	4	44	6503	6733		
38	99	23	4	66	6547	6799		
39	105	23	4	59	6757	6995	104;	
40	58	24	5	91	6041	6311	57;	
41	32	24	5	31	6277	6487		
42	7	25	5	62	6199	6440		8;
43	100	25	5	32	6483	6694		1;
44	33	26	5	25	6308	6512		
45	9	27	5	85	6350	6614		10;
46	179	27	5	30	6482	6691	178;	
47	132	28	5	22	6390	6591	#N/A	#N/A
48	10	29	5	65	6468	6712	9;	
49	10	29	5	26	6533	6738	9;	
50	180	30	5	86	6512	6777		181;
51	180	30	5	32	6598	6809		181;
52	102	31	5	22	6519	6720		103;
53	38	32	5	85	6540	6804		39;
54	158	32	5	30	6628	6837		159;
55	183	33	5	66	7094	7339	182;	
56	183	33	5	45	7160	7384	182;	
57	78	34	5	64	7437	7680		79;
58	78	34	5	31	7501	7711		79;
59	185	35	5	111	7484	7774	184;	
60	185	35	5	47	7595	7821	184;	
61	142	36	5	107	7555	7841	140;141;	
62	142	36	5	37	7662	7878	140;141;	
63	186	37	5	73	7642	7894		
64	186	37	5	42	7715	7936		
65	143	38	5	97	7699	7975		144;145;
66	143	38	5	29	7796	8004		144;145;
67	113	39	5	83	8028	8290	112;	
68	113	39	5	39	8111	8329	112;	
69	26	40	6	48	6045	6272		27;
70	26	40	6	6	6093	6278		27;
71	26	40	6	23	6099	6301		27;
72	181	41	6	71	6994	7244	180;	
73	4	42	7	49	6050	6278		5;
74	4	42	7	10	6099	6288		5;
75	151	43	7	61	6113	6353	150;	152;
76	151	43	7	24	6174	6377	150;	152;
77	5	44	7	35	6148	6362	4;	
78	131	45	7	43	6325	6547		
79	64	45	7	34	6384	6597	63;	
80	65	46	7	47	6418	6644		66;
81	65	46	7	12	6465	6656		66;
82	67	47	7	46	6511	6736		68;
83	67	47	7	16	6557	6752		68;
84	67	47	7	12	6573	6764		68;
85	160	48	7	62	6776	7017		
86	160	48	7	20	6838	7037		
87	166	49	7	56	7295	7532	165;165;	
88	166	49	7	28	7351	7560	165;165;	
89	141	50	7	48	7443	7671	140;	142;
90	141	50	7	22	7491	7693	140;	142;
91	123	51	8	92	6051	6323	122;	124;
92	177	51	8	31	6359	6569	176;	
93	28	52	8	15	6073	6267		29;
94	124	53	8	30	6143	6364	122;123;	
95	125	54	8	62	6173	6426		126;127;
96	157	54	8	31	6574	6784		
97	126	55	8	30	6235	6456	125;	127;
98	129	56	8	28	6262	6469	128;	130;
99	34	57	8	29	6319	6527		
100	35	58	8	69	6368	6616		
101	135	59	8	17	6581	6777	134;	136;
102	159	60	8	83	6688	6950	158;	

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
103	159	60	8	23	6771	6973	158;	
104	140	61	8	64	7358	7601		141;142;
105	140	61	8	25	7422	7626		141;142;
106	144	62	8	56	7790	8025	143;	145;
107	144	62	8	25	7846	8050	143;	145;
108	145	63	8	56	7871	8106	143;144;	
109	145	63	8	24	7927	8130	143;144;	
110	17	64	8	47	8150	8376		18;
111	17	64	8	18	8197	8394		18;
112	18	65	8	46	8215	8440	17;	
113	18	65	8	27	8261	8467	17;	
114	115	66	8	46	8244	8469	114;	
115	115	66	8	18	8290	8487	114;	
116	19	67	8	55	8288	8522		20;
117	19	67	8	23	8343	8545		20;
118	93	68	9	71	6062	6312		
119	93	68	9	28	6133	6340		
120	155	69	9	27	6456	6662	153;154;	
121	101	70	9	27	6484	6690		
122	104	71	9	79	6646	6904		105;
123	104	71	9	5	6725	6909		105;
124	104	71	9	21	6730	6930		105;
125	69	72	9	72	6669	6920		70;71;
126	69	72	9	5	6741	6925		70;71;
127	69	72	9	18	6746	6943		70;71;
128	70	73	9	75	6764	7018	69;	71;
129	70	73	9	5	6839	7023	69;	71;
130	70	73	9	20	6844	7043	69;	71;
131	71	74	9	78	6864	7121	69;70;	
132	71	74	9	5	6942	7126	69;70;	
133	71	74	9	20	6947	7146	69;70;	
134	162	75	9	72	6884	7135	161;	
135	162	75	9	5	6956	7140	161;	
136	162	75	9	18	6961	7158	161;	
137	72	76	9	63	6967	7209		
138	72	76	9	5	7030	7214		
139	72	76	9	13	7035	7227		
140	73	77	9	67	7048	7294		74;
141	73	77	9	5	7115	7299		74;
142	73	77	9	17	7120	7316		74;
143	164	78	9	67	7122	7370	163;	
144	164	78	9	5	7189	7375	163;	
145	164	78	9	17	7194	7392	163;	
146	74	79	9	58	7137	7374	73;	
147	74	79	9	5	7195	7379	73;	
148	74	79	9	18	7200	7397	73;	
149	75	80	9	14	7218	7411		76;
150	76	81	9	68	7232	7479	75;	
151	76	81	9	5	7300	7484	75;	
152	76	81	9	17	7305	7501	75;	
153	77	82	9	62	7322	7563		
154	77	82	9	5	7384	7568		
155	77	82	9	11	7389	7579		
156	27	83	10	29	6070	6278	26;	
157	31	84	10	30	6228	6437	30;	
158	178	85	10	83	6403	6665		179;
159	37	85	10	36	6501	6716	36;	
160	36	86	10	30	6459	6668		37;
161	134	87	10	75	6477	6731		135;136;
162	134	87	10	29	6552	6760		135;136;
163	11	88	10	73	6522	6774		12;
164	11	88	10	33	6595	6807		12;
165	103	89	10	80	6541	6800	102;	
166	103	89	10	20	6621	6820	102;	
167	40	90	10	50	6741	6970		41;
168	40	90	10	19	6791	6989		41;
169	14	91	10	84	6880	7143	13;	
170	14	91	10	30	6964	7173	13;	
171	14	91	10	16	6988	7188	13;	
172	167	92	10	65	7642	7886		168;
173	167	92	10	24	7707	7910		168;
174	168	93	10	65	7731	7975	167;	
175	168	93	10	17	7796	7992	167;	

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
176	169	94	10	75	7813	8067		170;171;
177	169	94	10	25	7888	8092		170;171;
178	16	95	10	57	7927	8165	15;	
179	16	95	10	25	7984	8190	15;	
180	85	96	10	54	8081	8314		86;
181	85	96	10	18	8135	8332		86;
182	114	97	10	75	8153	8407		115;
183	114	97	10	16	8228	8423		115;
184	20	98	10	47	8366	8592	19;	
185	20	98	10	26	8413	8618	19;	
186	150	99	11	34	6079	6292		151;152;
187	59	100	11	45	6177	6407		60;
188	59	100	11	27	6222	6434		60;
189	60	101	11	44	6249	6478	59;	
190	60	101	11	26	6293	6504	59;	
191	8	102	11	44	6260	6483	7;	
192	8	102	11	30	6304	6513	7;	
193	109	103	11	69	7640	7888	108;	
194	109	103	11	35	7709	7923	108;	
195	81	104	11	61	7702	7942	80;	
196	81	104	11	16	7763	7958	80;	
197	81	104	11	19	7779	7977	80;	
198	82	105	11	51	7798	8028		83;84;
199	82	105	11	34	7849	8062		83;84;
200	128	106	12	85	6171	6435		129;130;
201	137	106	12	24	6658	6861		138;139;
202	130	107	12	29	6290	6498	128;129;	
203	66	108	12	30	6489	6698	65;	
204	39	109	12	59	6628	6866	38;	
205	39	109	12	31	6687	6897	38;	
206	106	110	12	58	6810	7047		107;
207	106	110	12	17	6868	7064		107;
208	107	111	12	75	6885	7139	106;	
209	107	111	12	32	6960	7171	106;	
210	42	112	12	64	6972	7215		43;
211	42	112	12	16	7036	7231		43;
212	139	113	12	57	7121	7360	137;138;	
213	139	113	12	26	7178	7386	137;138;	
214	15	114	12	46	7857	8084		16;
215	15	114	12	24	7903	8108		16;
216	86	115	12	48	8153	8380	85;	
217	86	115	12	17	8201	8397	85;	
218	87	116	12	56	8218	8453		88;
219	87	116	12	22	8274	8475		88;
220	88	117	12	56	8296	8531	87;	
221	88	117	12	25	8352	8556	87;	
222	89	118	12	47	8377	8603		90;
223	89	118	12	26	8424	8629		90;
224	90	119	12	62	8450	8691	89;	
225	90	119	12	24	8512	8715	89;	
226	91	120	12	46	8536	8761		92;
227	91	120	12	17	8582	8778		92;
228	92	121	12	63	8599	8841	91;	
229	92	121	12	17	8662	8858	91;	
230	29	122	13	38	6110	6327	28;	
231	30	123	14	95	6142	6416		31;
232	13	123	14	28	6852	7059		14;
233	175	124	14	27	6256	6462	174;	
234	176	125	14	89	6283	6551		177;
235	108	125	14	26	6992	7197		109;
236	136	126	14	25	6630	6834	134;135;	
237	138	127	14	26	6682	6887	137;	139;
238	12	128	14	77	6775	7031	11;	
239	41	129	14	32	6794	7005	40;	
240	161	130	14	26	6858	7063		162;
241	163	131	14	72	6979	7230		164;164;
242	163	131	14	32	7051	7262		164;164;
243	182	132	14	27	7067	7273		183;
244	165	133	14	64	7211	7456		166;
245	165	133	14	20	7275	7476		166;
246	110	134	14	80	7744	8003		111;
247	110	134	14	26	7824	8029		111;
248	111	135	14	53	7858	8090	110;	

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
249	111	135	14	17	7911	8107	110;	
250	112	136	14	71	7928	8178		113;
251	112	136	14	29	7999	8207		113;
252	52	137	14	75	8173	8427		
253	52	137	14	17	8248	8444		
254	61	138	15	64	6144	6387		62;
255	61	138	15	20	6208	6407		62;
256	43	139	15	29	7159	7367	42;	
257	44	140	15	30	7188	7397		
258	45	141	15	76	7218	7473		46;
259	45	141	15	34	7294	7507		46;
260	46	142	15	29	7328	7536	45;	
261	47	143	15	76	7357	7612		
262	47	143	15	27	7433	7639		
263	48	144	15	51	7460	7690		49;
264	48	144	15	18	7511	7708		49;
265	49	145	15	68	7529	7776	48;	
266	49	145	15	27	7597	7803	48;	
267	50	146	15	25	7624	7828		51;
268	51	147	15	51	7649	7879	50;	
269	51	147	15	25	7700	7904	50;	
270	116	148	15	73	8345	8601		117;
271	116	148	15	27	8418	8628		117;
272	117	149	15	50	8445	8678	116;	
273	117	149	15	17	8495	8695	116;	
274	118	150	15	66	8512	8761		119;
275	118	150	15	27	8578	8788		119;
276	119	151	15	73	8605	8861	118;	
277	119	151	15	17	8678	8878	118;	
278	6	152	16	39	6161	6379		
279	6	152	16	15	6200	6394		
280	184	153	16	46	7400	7625		185;
281	184	153	16	27	7446	7652		185;
282	80	154	16	61	7593	7833		81;
283	80	154	16	48	7654	7881		81;
284	83	155	16	61	7883	8123	82;	84;
285	83	155	16	40	7944	8163	82;	84;
286	133	156	17	51	6397	6627	131;	
287	133	156	17	13	6448	6640	131;	
288	133	156	17	10	6461	6650	131;	
289	156	157	18	56	6502	6737		
290	156	157	18	23	6558	6760		
291	68	158	19	46	6585	6810	67;	
292	68	158	19	12	6631	6822	67;	
293	68	158	19	14	6643	6836	67;	
294	79	159	20	52	7513	7744	78;	
295	79	159	20	12	7565	7756	78;	
296	79	159	20	16	7577	7772	78;	
297	84	160	20	59	7984	8222	82;83;	
298	84	160	20	28	8043	8250	82;83;	
299	146	2	2	75	8071	8325		147;148;
300	146	2	2	67	8146	8392		147;148;
301	147	1	1	98	8213	8490	146;	148;
302	147	1	1	74	8311	8564	146;	148;
303	120	161	21	108	8706	8993		
304	120	161	21	31	8814	9024		
305	121	162	21	114	8845	9138		
306	121	162	21	29	8959	9167		
307	122	163	21	70	8988	9237		123;124;
308	122	163	21	33	9058	9270		123;124;
309	53	164	21	99	9009	9287		54;
310	53	164	21	23	9108	9310		54;
311	54	165	21	78	9131	9388	53;	
312	54	165	21	32	9209	9420	53;	
313	55	166	21	90	9241	9510		56;
314	55	166	21	48	9331	9558		56;
315	56	167	21	83	9379	9641	55;	
316	56	167	21	23	9462	9664	55;	
317	24	168	21	76	9383	9638		25;
318	24	168	21	34	9459	9672		25;
319	57	169	21	63	9485	9727		58;
320	57	169	21	25	9548	9752		58;
321	25	170	21	103	9493	9775	24;	

Task ID	Block ID	Well ID	Project ID	Processing Time	Release Date	Due Date	Precedence List	Sucessors List
322	25	170	21	31	9596	9806	24;	
323	21	171	22	14	9233	9426		22;23;
324	22	172	23	36	9247	9462	21;	23;
325	23	173	23	74	9283	9536	21;22;	
326	23	173	23	26	9357	9562	21;22;	

Appendix III: Mathematical Models Solutions

This appendix provides the schedules found by the mathematical models. Figures 21, 22 and 23 are the final solutions of the models using instance 01.

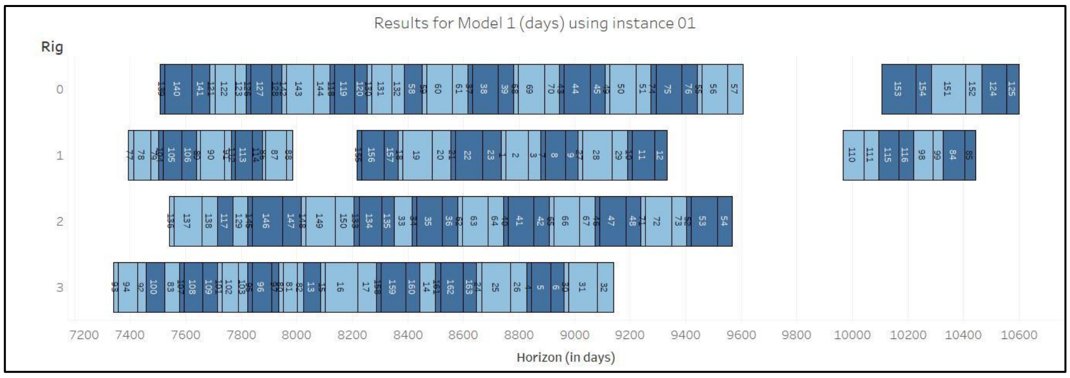

Figure 21. Mathematical model 1 in days result for instance 01 (Source: the author).

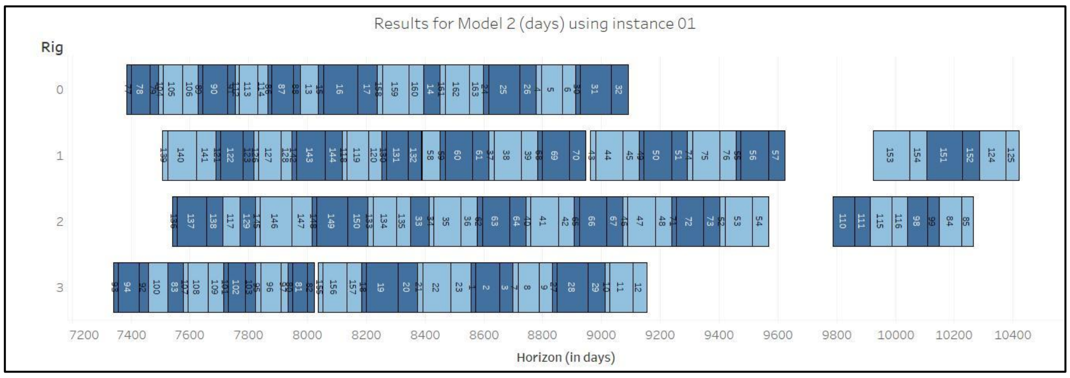

Figure 22. Mathematical model 2 in days results for instance 01 (Source: the author).

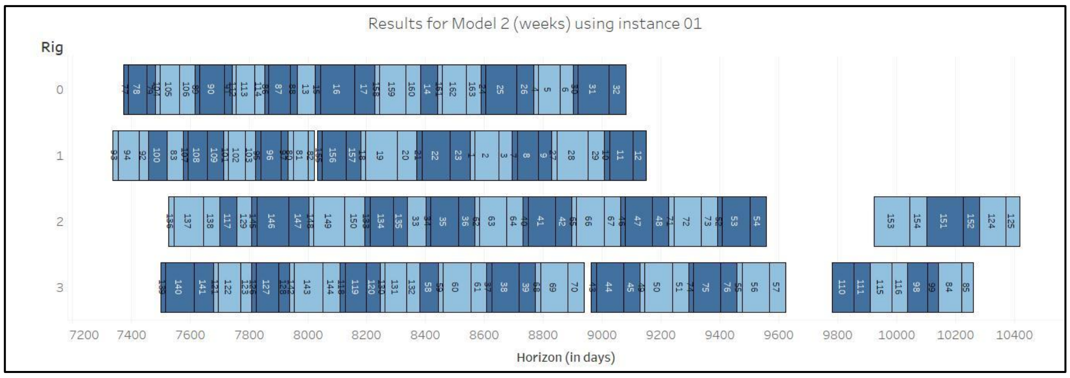

Figure 23. Mathematical model 2 in weeks results for instance 01 (Source: the author)

Meanwhile, Figures 24 and 25 are related with the linear programming solutions of instance 02.

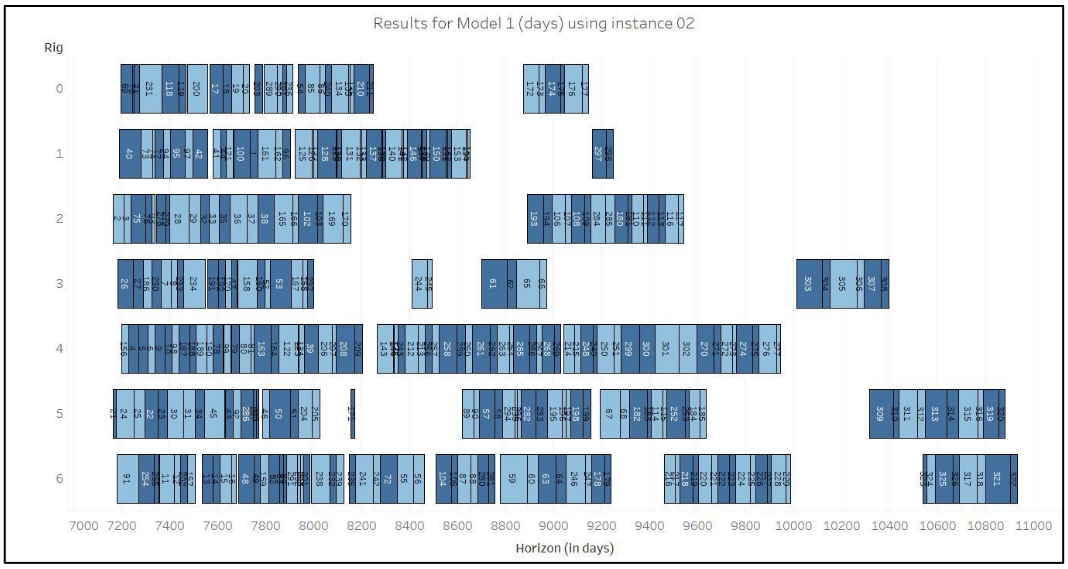

Figure 24. Mathematical model 1 in days results for instance 02 (Source: the author).

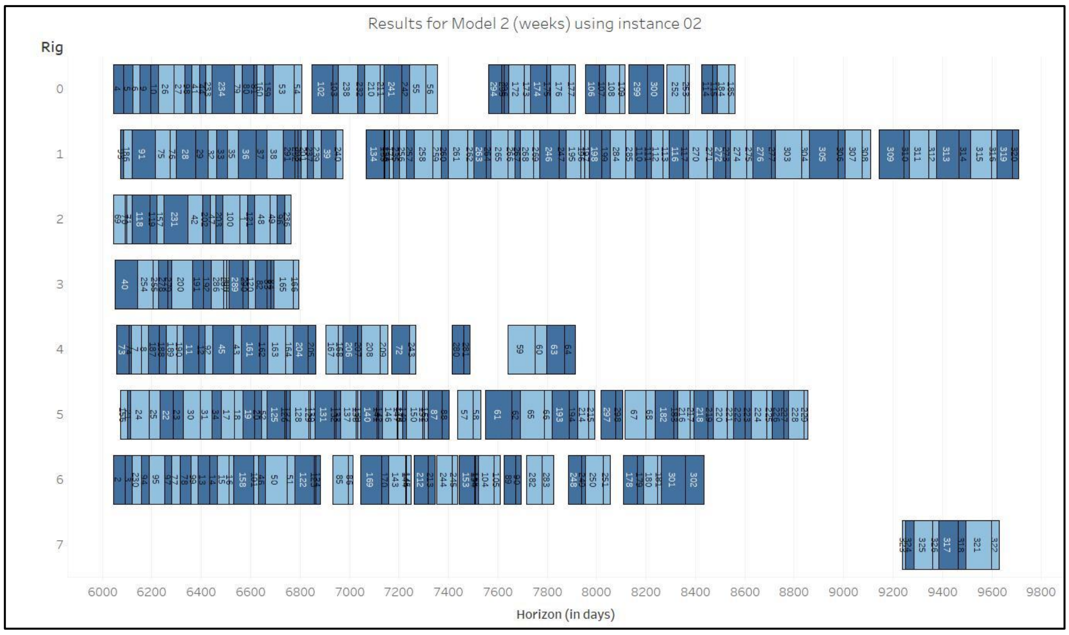

Figure 25. Mathematical model 2 in weeks results for instance 02 (Source: the author).

Appendix IV: Heuristic Results

The following table presents the results of all the heuristics tested for instances 01 and 02 using the initial solutions provided by mathematical model 1 in weeks:

Table 22. Heuristic results for instances 01 and 02 (Source: the author).

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
1	LS	Insert with Fixed Dates	New Rig	First	1,073,498,829	3.2%	0.181	1,983,145,847	7.9%	0.458
2	LS	Insert with Fixed Dates	New Rig	Best	1,073,498,829	3.2%	0.168	1,921,761,999	10.8%	0.732
3	LS	Insert with Fixed Dates	New Rig	Best-First	1,073,498,829	3.2%	0.166	1,921,761,999	10.8%	0.632
4	LS	Insert with Fixed Dates	New Rig	First-Best	1,073,498,829	3.2%	0.144	1,983,145,847	7.9%	0.389
5	LS	Insert with Fixed Dates	New Rig	Random	1,073,498,829	3.2%	0.145	1,958,592,308	9.1%	0.501
6	LS	Insert with Fixed Dates	Existing Rig	First	1,086,099,877	2.1%	0.127	1,922,748,946	10.7%	2.927
7	LS	Insert with Fixed Dates	Existing Rig	Best	1,086,099,877	2.1%	0.088	1,923,439,603	10.7%	2.924
8	LS	Insert with Fixed Dates	Existing Rig	Best-First	1,086,099,877	2.1%	0.153	1,922,748,946	10.7%	1.995
9	LS	Insert with Fixed Dates	Existing Rig	First-Best	1,086,099,877	2.1%	0.062	1,923,439,603	10.7%	2.816
10	LS	Insert with Fixed Dates	Existing Rig	Random	1,086,099,877	2.1%	0.104	1,922,887,077	10.7%	2.753
11	VND	Insert with Fixed Dates	New Rig - Existing	First	1,069,932,827	3.5%	0.433	1,843,478,284	14.4%	10.219
12	VND	Insert with Fixed Dates	New Rig - Existing	Best	1,069,932,827	3.5%	0.320	1,811,806,733	15.9%	7.498
13	VND	Insert with Fixed Dates	New Rig - Existing	Best-First	1,069,932,827	3.5%	0.368	1,808,057,453	16.1%	9.521
14	VND	Insert with Fixed Dates	New Rig - Existing	First-Best	1,069,932,827	3.5%	0.444	1,843,478,284	14.4%	8.892
15	VND	Insert with Fixed Dates	New Rig - Existing	Random	1,069,932,827	3.5%	0.391	1,829,379,017	15.1%	8.550
16	VND	Insert with Fixed Dates	Existing - Rig	First	1,069,932,827	3.5%	0.496	1,808,057,453	16.1%	8.455
17	VND	Insert with Fixed Dates	Existing - Rig	Best	1,069,932,827	3.5%	0.341	1,811,806,733	15.9%	6.762
18	VND	Insert with Fixed Dates	Existing - Rig	Best-First	1,069,932,827	3.5%	0.468	1,808,057,453	16.1%	6.611
19	VND	Insert with Fixed Dates	Existing - Rig	First-Best	1,069,932,827	3.5%	0.393	1,808,057,453	16.1%	8.097
20	VND	Insert with Fixed Dates	Existing - Rig	Random	1,069,932,827	3.5%	0.411	1,808,363,316	16.0%	7.249
21	VND	Insert with Fixed Dates	Random	First	1,069,932,827	3.5%	0.490	1,808,057,453	16.1%	8.365

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
22	VND	Insert with Fixed Dates	Random	Best	1,069,932,827	3.5%	0.348	1,811,806,733	15.9%	6.869
23	VND	Insert with Fixed Dates	Random	Best-First	1,069,932,827	3.5%	0.478	1,808,057,453	16.1%	6.465
24	VND	Insert with Fixed Dates	Random	First-Best	1,069,932,827	3.5%	0.375	1,808,057,453	16.1%	8.063
25	VND	Insert with Fixed Dates	Random	Random	1,069,932,827	3.5%	0.400	1,808,363,316	16.0%	7.273
26	LS	Insert with Fixed Dates	New Rig + Existing	Best	1,072,202,128	3.3%	0.334	1,811,806,733	15.9%	8.198
58	LS	Insert with Dates Change	Anticipation (After-In)	First	1,067,452,142	3.7%	11.038	1,834,640,863	14.8%	185.561
59	LS	Insert with Dates Change	Anticipation (After-In)	Best	1,034,201,947	6.7%	17.501	1,818,558,425	15.6%	150.282
60	LS	Insert with Dates Change	Anticipation (After-In)	Best-First	1,034,201,947	6.7%	17.880	1,818,854,421	15.6%	201.225
61	LS	Insert with Dates Change	Anticipation (After-In)	First-Best	1,067,452,142	3.7%	11.169	1,834,640,863	14.8%	97.408
62	LS	Insert with Dates Change	Anticipation (After-In)	Random	1,054,152,064	4.9%	13.843	1,828,267,087	15.1%	141.032
63	LS	Insert with Dates Change	Anticipation (After-From)	First	1,059,178,313	4.5%	30.237	1,794,328,928	16.7%	482.491
64	LS	Insert with Dates Change	Anticipation (After-From)	Best	1,034,201,947	6.7%	24.447	1,755,934,087	18.5%	356.752
65	LS	Insert with Dates Change	Anticipation (After-From)	Best-First	1,034,201,947	6.7%	24.450	1,755,835,421	18.5%	509.226
66	LS	Insert with Dates Change	Anticipation (After-From)	First-Best	1,051,961,695	5.1%	31.829	1,729,181,747	19.7%	487.739
67	LS	Insert with Dates Change	Anticipation (After-From)	Random	1,046,301,119	5.6%	28.637	1,751,422,273	18.7%	409.491
68	LS	Insert with Dates Change	Anticipation (Before-In)	First	1,084,323,902	2.2%	33.652	1,718,906,506	20.2%	439.109
69	LS	Insert with Dates Change	Anticipation (Before-In)	Best	1,084,323,902	2.2%	26.773	1,779,486,979	17.4%	147.802
70	LS	Insert with Dates Change	Anticipation (Before-In)	Best-First	1,084,323,902	2.2%	26.466	1,779,486,979	17.4%	194.812
71	LS	Insert with Dates Change	Anticipation (Before-In)	First-Best	1,084,323,902	2.2%	24.728	1,721,077,142	20.1%	254.681
72	LS	Insert with Dates Change	Anticipation (Before-In)	Random	1,084,323,902	2.2%	27.154	1,745,703,992	19.0%	202.639
73	LS	Insert with Dates Change	Anticipation (Before-From)	First	1,059,178,313	4.5%	64.046	1,723,670,545	20.0%	1058.205
74	LS	Insert with Dates Change	Anticipation (Before-From)	Best	1,059,178,313	4.5%	26.370	1,794,328,928	16.7%	263.441
75	LS	Insert with Dates Change	Anticipation (Before-From)	Best-First	1,059,178,313	4.5%	25.741	1,794,328,928	16.7%	317.222
76	LS	Insert with Dates Change	Anticipation (Before-From)	First-Best	1,059,178,313	4.5%	34.993	1,734,241,781	19.5%	386.299
77	LS	Insert with Dates Change	Anticipation (Before-From)	Random	1,060,153,693	4.4%	35.582	1,757,550,762	18.4%	330.313
78	LS	Insert with Dates Change	Postponement (After-In)	First	1,084,225,237	2.2%	13.261	1,916,237,038	11.0%	275.638
79	LS	Insert with Dates Change	Postponement (After-In)	Best	1,084,225,237	2.2%	17.649	1,916,237,038	11.0%	198.278
80	LS	Insert with Dates Change	Postponement (After-In)	Best-First	1,084,225,237	2.2%	17.689	1,916,237,038	11.0%	302.329
81	LS	Insert with Dates Change	Postponement (After-In)	First-Best	1,084,225,237	2.2%	13.338	1,916,237,038	11.0%	198.212
82	LS	Insert with Dates Change	Postponement (After-In)	Random	1,084,225,237	2.2%	15.099	1,916,237,038	11.0%	204.871
83	LS	Insert with Dates Change	Postponement (After-From)	First	1,072,498,123	3.3%	30.242	1,832,329,109	14.9%	527.865
84	LS	Insert with Dates Change	Postponement (After-From)	Best	1,071,610,136	3.4%	26.923	1,797,782,213	16.5%	453.891
85	LS	Insert with Dates Change	Postponement (After-From)	Best-First	1,071,610,136	3.4%	26.783	1,797,782,213	16.5%	387.782
86	LS	Insert with Dates Change	Postponement (After-From)	First-Best	1,072,498,123	3.3%	35.042	1,793,736,937	16.7%	575.933
87	LS	Insert with Dates Change	Postponement (After-From)	Random	1,072,142,928	3.3%	30.862	1,803,300,412	16.3%	493.862
88	LS	Insert with Dates Change	Postponement (Before-In)	First	1,084,027,907	2.2%	32.900	1,916,434,369	11.0%	350.668
89	LS	Insert with Dates Change	Postponement (Before-In)	Best	1,084,027,907	2.2%	17.780	1,917,618,352	11.0%	153.800
90	LS	Insert with Dates Change	Postponement (Before-In)	Best-First	1,084,027,907	2.2%	17.730	1,917,618,352	11.0%	205.123
91	LS	Insert with Dates Change	Postponement (Before-In)	First-Best	1,084,027,907	2.2%	24.723	1,916,927,695	11.0%	101.981

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
92	LS	Insert with Dates Change	Postponement (Before-In)	Random	1,084,027,907	2.2%	23.602	1,917,312,490	11.0%	150.852
93	LS	Insert with Dates Change	Postponement (Before-From)	First	1,071,412,806	3.4%	43.207	1,802,926,859	16.3%	1134.064
94	LS	Insert with Dates Change	Postponement (Before-From)	Best	1,071,412,806	3.4%	26.918	1,815,739,291	15.7%	350.341
95	LS	Insert with Dates Change	Postponement (Before-From)	Best-First	1,071,412,806	3.4%	26.797	1,815,739,291	15.7%	327.502
96	LS	Insert with Dates Change	Postponement (Before-From)	First-Best	1,071,412,806	3.4%	35.154	1,815,739,291	15.7%	595.840
97	LS	Insert with Dates Change	Postponement (Before-From)	Random	1,071,412,806	3.4%	33.397	1,815,995,821	15.7%	456.835
98	VND	Insert with Dates Change	Random	First	1,053,461,378	5.0%	36.588	1,730,104,925	19.7%	514.143
99	VND	Insert with Dates Change	Random	Best	1,047,924,851	5.5%	33.575	1,745,835,009	18.9%	379.469
100	VND	Insert with Dates Change	Random	Best-First	1,051,567,005	5.2%	30.298	1,769,034,023	17.9%	318.272
101	VND	Insert with Dates Change	Random	First-Best	1,047,762,771	5.5%	39.942	1,736,921,290	19.4%	394.159
102	VND	Insert with Dates Change	Random	Random	1,044,952,216	5.8%	33.697	1,740,199,817	19.2%	369.808
103	VND	Insert with Dates Change	Random (Anticipation-Postponement)	First	1,036,567,074	6.5%	81.800	1,676,686,174	22.2%	1471.458
104	VND	Insert with Dates Change	Random (Anticipation-Postponement)	Best	1,034,588,147	6.7%	55.664	1,674,703,003	22.2%	1006.117
105	VND	Insert with Dates Change	Random (Anticipation-Postponement)	Best-First	1,033,047,564	6.8%	63.971	1,677,124,547	22.1%	1280.344
106	VND	Insert with Dates Change	Random (Anticipation-Postponement)	First-Best	1,034,965,886	6.7%	72.375	1,673,235,730	22.3%	961.189
107	VND	Insert with Dates Change	Random (Anticipation-Postponement)	Random	1,046,215,131	5.7%	56.285	1,688,630,293	21.6%	1005.645
108	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-In)	First	1,067,452,142	3.7%	20.020	1,834,640,863	14.8%	238.889
109	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-In)	Best	1,034,201,947	6.7%	26.722	1,818,558,425	15.6%	202.274
110	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-In)	Best-First	1,034,201,947	6.7%	26.566	1,818,558,425	15.6%	202.494
111	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-In)	First-Best	1,067,452,142	3.7%	19.996	1,834,640,863	14.8%	238.674
112	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-In)	Random	1,054,152,064	4.9%	22.674	1,828,267,087	15.1%	193.521
113	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-From)	First	1,052,567,740	5.1%	47.805	1,708,166,045	20.7%	1078.258
114	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-From)	Best	1,034,201,947	6.7%	29.470	1,668,206,612	22.6%	664.749
115	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-From)	Best-First	1,034,201,947	6.7%	29.263	1,686,868,400	21.7%	1003.388
116	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-From)	First-Best	1,052,567,740	5.1%	52.758	1,684,289,051	21.8%	819.079

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
117	VND	Insert with Dates Change	Anticipation (After-In) Postponement (After-From)	Random	1,038,117,524	6.4%	43.396	1,679,049,925	22.0%	758.046
118	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-In)	First	1,032,129,977	6.9%	54.509	1,811,947,852	15.9%	991.620
119	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-In)	Best	1,032,129,977	6.9%	44.272	1,809,678,551	16.0%	462.662
120	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-In)	Best-First	1,032,129,977	6.9%	61.331	1,811,947,852	15.9%	791.646
121	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-In)	First-Best	1,032,129,977	6.9%	37.801	1,809,678,551	16.0%	600.127
122	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-In)	Random	1,032,129,977	6.9%	47.264	1,810,586,271	15.9%	519.406
123	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-From)	First	1,038,458,607	6.4%	61.994	1,698,990,175	21.1%	1890.072
124	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-From)	Best	1,032,129,977	6.9%	50.111	1,698,990,175	21.1%	617.401
125	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-From)	Best-First	1,032,129,977	6.9%	70.020	1,698,990,175	21.1%	1904.312
126	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-From)	First-Best	1,032,129,977	6.9%	43.517	1,698,990,175	21.1%	647.154
127	VND	Insert with Dates Change	Anticipation (After-In) Postponement (Before-From)	Random	1,033,395,703	6.8%	54.090	1,691,414,088	21.5%	746.466
128	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-In)	First	1,051,172,373	5.2%	71.498	1,792,651,619	16.8%	758.857
129	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-In)	Best	1,034,201,947	6.7%	33.271	1,755,934,087	18.5%	409.150
130	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-In)	Best-First	1,034,201,947	6.7%	33.649	1,755,934,087	18.5%	410.714
131	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-In)	First-Best	1,051,172,373	5.2%	58.578	1,792,651,619	16.8%	762.441
132	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-In)	Random	1,044,699,932	5.8%	42.825	1,750,583,619	18.7%	547.838
133	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-From)	First	1,059,178,313	4.5%	43.468	1,637,803,657	24.0%	2068.001
134	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-From)	Best	1,034,201,947	6.7%	36.721	1,718,737,281	20.2%	813.811
135	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-From)	Best-First	1,034,201,947	6.7%	35.976	1,718,737,281	20.2%	803.450

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
136	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-From)	First-Best	1,059,178,313	4.5%	43.666	1,692,379,602	21.4%	1419.234
137	VND	Insert with Dates Change	Anticipation (After-From) Postponement (After-From)	Random	1,046,301,119	5.6%	40.451	1,687,298,341	21.7%	1101.412
138	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-In)	First	1,032,129,977	6.9%	136.095	1,706,094,074	20.8%	2635.061
139	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-In)	Best	1,032,129,977	6.9%	54.027	1,706,982,062	20.8%	779.352
140	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-In)	Best-First	1,032,129,977	6.9%	73.953	1,706,982,062	20.8%	1523.639
141	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-In)	First-Best	1,032,129,977	6.9%	59.351	1,706,094,074	20.8%	1203.307
142	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-In)	Random	1,032,129,977	6.9%	66.275	1,704,222,245	20.9%	1018.472
143	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-From)	First	1,038,458,607	6.4%	120.576	1,698,990,175	21.1%	2481.049
144	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-From)	Best	1,032,129,977	6.9%	59.355	1,697,214,200	21.2%	796.939
145	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-From)	Best-First	1,032,129,977	6.9%	82.417	1,697,214,200	21.2%	1740.359
146	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-From)	First-Best	1,038,458,607	6.4%	70.433	1,672,660,602	22.3%	1308.533
147	VND	Insert with Dates Change	Anticipation (After-From) Postponement (Before-From)	Random	1,033,395,703	6.8%	74.085	1,689,509,848	21.6%	1121.886
148	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-In)	First	1,084,323,902	2.2%	42.257	1,718,906,506	20.2%	489.717
149	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-In)	Best	1,084,323,902	2.2%	35.516	1,779,092,318	17.4%	303.263
150	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-In)	Best-First	1,084,323,902	2.2%	35.304	1,779,092,318	17.4%	299.904
151	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-In)	First-Best	1,084,323,902	2.2%	42.115	1,718,906,506	20.2%	489.970
152	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-In)	Random	1,084,323,902	2.2%	36.012	1,745,546,128	19.0%	293.797
153	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-From)	First	1,069,439,500	3.6%	70.716	1,687,939,665	21.6%	836.779
154	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-From)	Best	1,069,439,500	3.6%	69.971	1,725,925,793	19.9%	665.802

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
155	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-From)	Best-First	1,069,439,500	3.6%	65.950	1,669,193,265	22.5%	1047.001
156	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-From)	First-Best	1,069,439,500	3.6%	75.773	1,687,939,665	21.6%	917.747
157	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (After-From)	Random	1,069,439,500	3.6%	68.820	1,680,763,890	22.0%	808.005
158	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-In)	First	1,049,001,737	5.4%	76.774	1,694,240,189	21.3%	1431.657
159	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-In)	Best	1,049,001,737	5.4%	52.861	1,756,004,646	18.5%	455.363
160	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-In)	Best-First	1,049,001,737	5.4%	69.800	1,758,668,608	18.4%	851.099
161	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-In)	First-Best	1,049,001,737	5.4%	60.352	1,694,240,189	21.3%	898.738
162	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-In)	Random	1,049,001,737	5.4%	58.914	1,721,757,932	20.1%	628.247
163	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-From)	First	1,055,330,367	4.8%	84.655	1,660,905,382	22.9%	1994.714
164	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-From)	Best	1,049,001,737	5.4%	58.686	1,670,785,962	22.4%	624.902
165	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-From)	Best-First	1,055,330,367	4.8%	77.307	1,675,310,511	22.2%	1984.101
166	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-From)	First-Best	1,049,001,737	5.4%	65.515	1,673,929,198	22.3%	851.145
167	VND	Insert with Dates Change	Anticipation (Before-In) Postponement (Before-From)	Random	1,050,900,326	5.2%	65.426	1,675,351,383	22.2%	930.242
168	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-In)	First	1,051,172,373	5.2%	103.495	1,672,350,553	22.4%	1777.616
169	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-In)	Best	1,051,172,373	5.2%	68.862	1,793,934,267	16.7%	460.745
170	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-In)	Best-First	1,051,172,373	5.2%	66.595	1,793,934,267	16.7%	456.639
171	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-In)	First-Best	1,051,172,373	5.2%	105.784	1,672,251,888	22.4%	1687.281
172	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-In)	Random	1,052,948,348	5.0%	73.890	1,752,321,503	18.6%	516.436
173	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-From)	First	1,059,178,313	4.5%	77.230	1,672,547,884	22.3%	1968.807

Heuristic\Instance					*Instance01* (Model 1 in Weeks) (Initial Objective Function: 1,108,595,557.84*m.u.*)			*Instance02* (Model 1 in Weeks) (Initial Objective Function: 2,153,921,664.30*m.u*)		
#	Heur.	Group	Neighborhood	Sear. Strat.	O.F. (*m.u.*)	Improvement (%)	Time (sec.)	O.F. (*m.u.*)	Improvement (%)	Time (sec.)
174	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-From)	Best	1,059,178,313	4.5%	39.528	1,685,289,756	21.8%	1111.087
175	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-From)	Best-First	1,059,178,313	4.5%	38.951	1,644,216,900	23.7%	1629.075
176	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-From)	First-Best	1,059,178,313	4.5%	77.844	1,666,627,968	22.6%	2119.591
177	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (After-From)	Random	1,060,153,693	4.4%	48.786	1,657,440,826	23.1%	1401.479
178	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-In)	First	1,032,129,977	6.9%	165.763	1,651,038,856	23.3%	2492.435
179	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-In)	Best	1,032,129,977	6.9%	68.619	1,707,376,723	20.7%	811.325
180	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-In)	Best-First	1,032,129,977	6.9%	130.820	1,696,918,205	21.2%	2024.452
181	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-In)	First-Best	1,032,129,977	6.9%	105.710	1,670,475,913	22.4%	1433.416
182	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-In)	Random	1,032,129,977	6.9%	85.051	1,685,357,446	21.8%	1196.122
183	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-From)	First	1,038,458,607	6.4%	151.056	1,651,038,856	23.3%	2761.467
184	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-From)	Best	1,038,458,607	6.4%	66.592	1,673,929,198	22.3%	1011.542
185	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-From)	Best-First	1,038,458,607	6.4%	115.749	1,661,018,101	22.9%	3318.343
186	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-From)	First-Best	1,038,458,607	6.4%	104.307	1,668,601,273	22.5%	1493.002
187	VND	Insert with Dates Change	Anticipation (Before-From) Postponement (Before-From)	Random	1,037,825,744	6.4%	85.509	1,665,428,468	22.7%	1036.264

Appendix V: Solution analysis

As there is the possibility that others approaches achieve similar results in a reduced time, but requiring a longer execution time to perform a complete run, it is important to analyze the objective function evolution over the time for each approach. The heuristics methods were analyzed for each instance comparing the objective function evolution over the execution time. The best approaches for instances 01 and 02 are presented in Chart 4 (left and right sides, respectively).

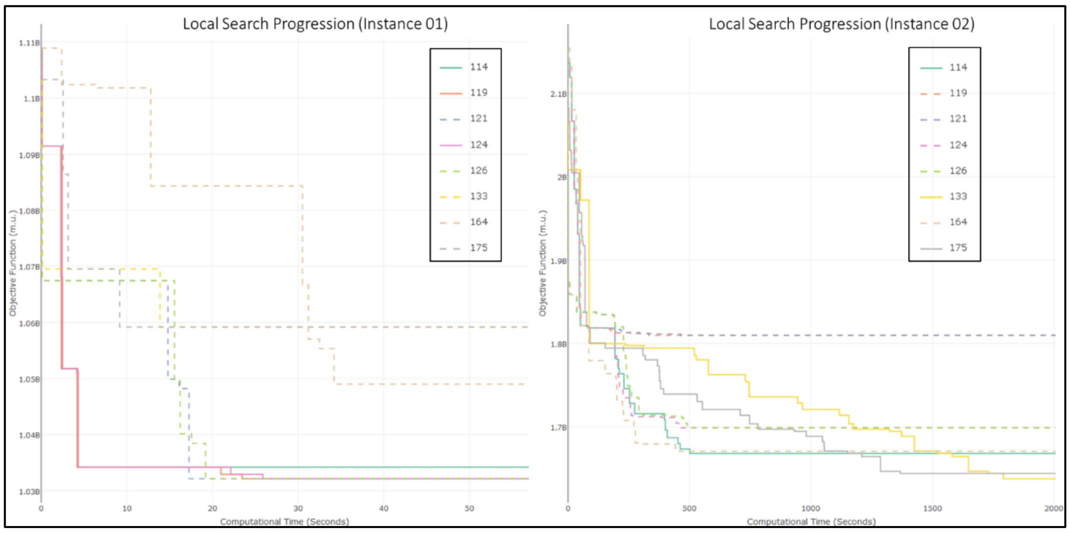

Chart 4. Progression for the best local searches, results for instance01 on the left and results for instance02 on the right (Source: the author).

The goal is to determine a local search approach that fits well to both instances. For *instance01*, the local searches #119, a *VND* using *insert with dates change Anticipation(After–In)–Postponement(Before–In)* neighborhood structure and *best-improvement* search strategy, and #124, a *VND* using *insert with dates change Anticipation(After–In)–Postponement(Before–From)* neighborhood structure and *best-improvement* search strategy, presented the most strong results, outperforming the others methods. On the other hand, these local searches did not produced powerful results in *instance02*, whose bests results were obtained by the approaches #133, a *VND* using *insert with dates change Anticipation(After–From)–Postponement(After–From)* with *first-improvement* search strategy, and #175, a *VND* using *insert with dates change Anticipation(Before–From)–*

Postponement(After–From) with *best-first* search strategy. It is important to notice that the results from heuristic #133 in instance are the best known results yet found. Despite not being the best approach possible in each instance, the local search #114, a *VND* using *insert with dates change Anticipation(After–In)–Postponement(After–From)* with *best-improvement*, found good results in both instances. These results emphasizes the conclusion of the Pareto frontier analysis, in which this variable neighborhood search was chosen as the most appropriated search method for the problem.

Next, with the goal of understanding the selected method, we analyze the movements performed by the local search and theirs impacts in the objective function, considering the results #6 (Model 1 followed by Model 2, both with a horizon plan in weeks) for instances 01 and 02 as initial solutions.

In the first instance, the *VND* #114 anticipates the last twenty-nine tasks from Rig 0, as shown in Figure 26, reducing the rig idleness costs, which results in an economy of 18 million fictional monetary unities.

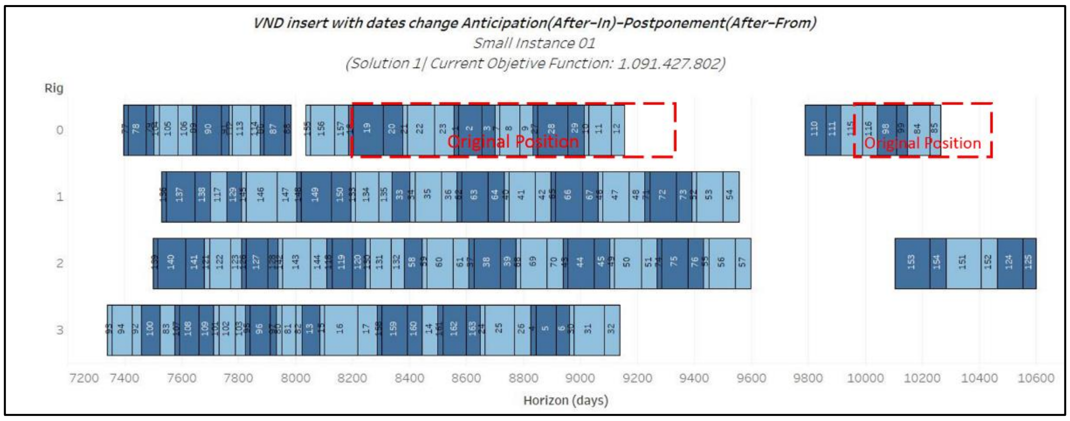

Figure 26. Results for the first iteration of the VND #114 using instance 01 (Source: the author).

Then, some of the last tasks from rig 0 are inserted in the end of the existing rig 1, as illustrated in Figure 27. Despite increasing the idleness cost in the rig 1, this movement radically reduces the idle cost in rig 0. Overall, the idle cost is reduce in more than 39 million of the fictional monetary unit.

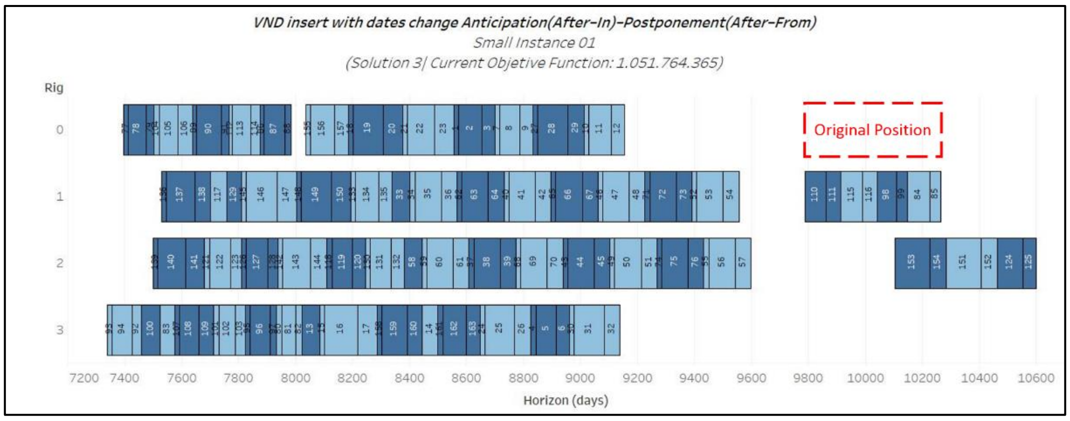

Figure 27. Results for the 2nd iteration of the VND #114 using instance 01 (Source: the author).

Last, the tasks at the end of rig 2 are anticipated, reduce the idleness cost in about 17 millions $u.m.$. The final rig schedule presented in Figure 28 has a budget of only $1.034.201.947, which represents an improvement of 6.7% in the original objective function with same rigs fleet size.

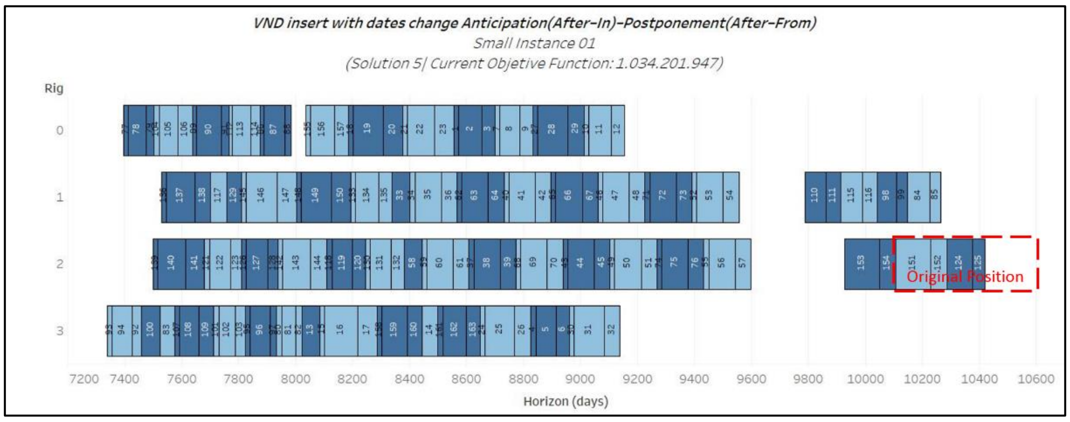

Figure 28. Results for the third iteration of the VND #114 using instance (Source: the author).

When tested with a large instance such as instance 02, the variable neighborhood search heuristic #114 was still able to improve the initial solution provided by the Model 1 (weeks) after performing an exhaustive search with more than twenty consecutive movements. Chart 5 summarizes the objective function evolution over each movement iteration and shows the heuristic’s improvement of 22.6% in the budget, after reducing the costs in almost half billion monetary unities and achieving a solution of 1,668,210,291 $u.m.$. Furthermore, it is possible to observe the reduction on rigs idle cost as the main source of improvement. In order to reduce the idle time, the algorithm allocates two new rigs to the fleet respectively at solution 16 and 27. Later, at the solution 30, the local search eliminates one of the rigs created, which initially seems to be superfluous combination of movement. However, moving some tasks to a temporary rig enables the heuristic to perform new insert movements and to interchange tasks.

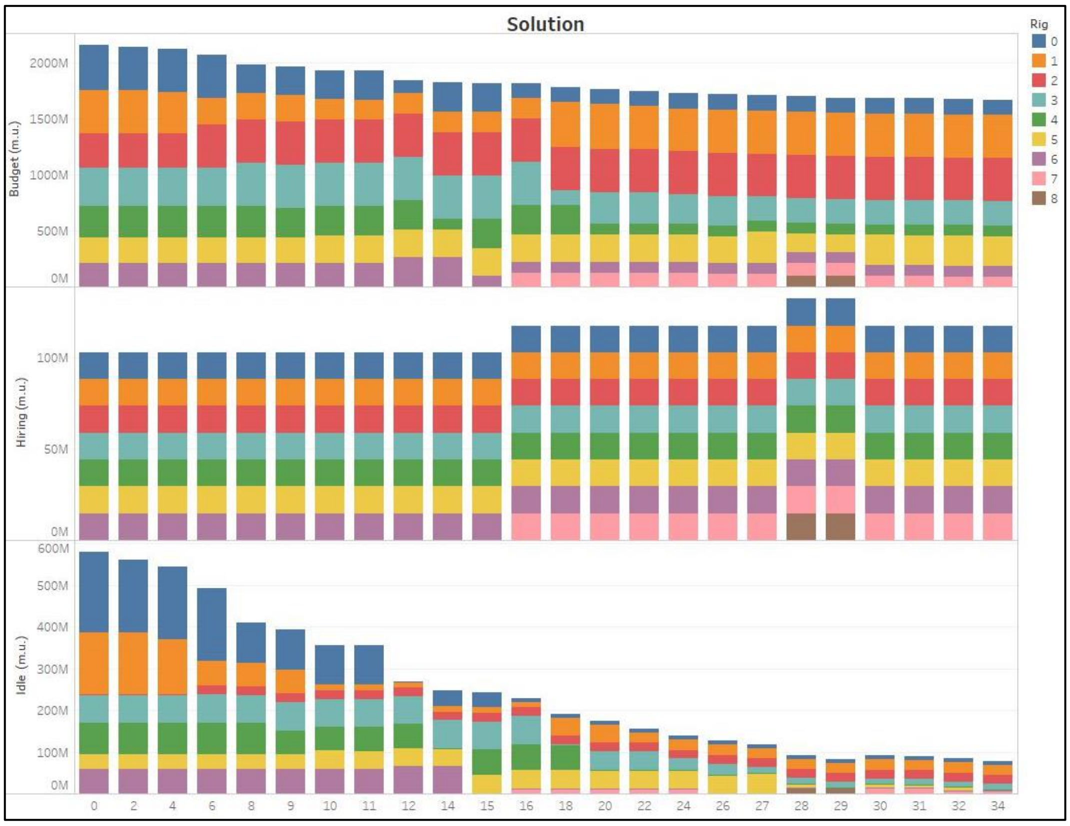

Chart 5. Budget for the solutions found by the VND #114 using instance 02 (Source: the author).

Clearly, the selected heuristic has great potential to efficient improve the budget, having obtained powerful results for the two instance, and it is definitely an appropriated local search method to be used in the matheuristic approach.

Appendix VI: Matheuristics Solutions

This appendix provides the schedules found by the matheuristics using the VND #114. Figures 29, 30 and 31 are the final solutions for instance 01.

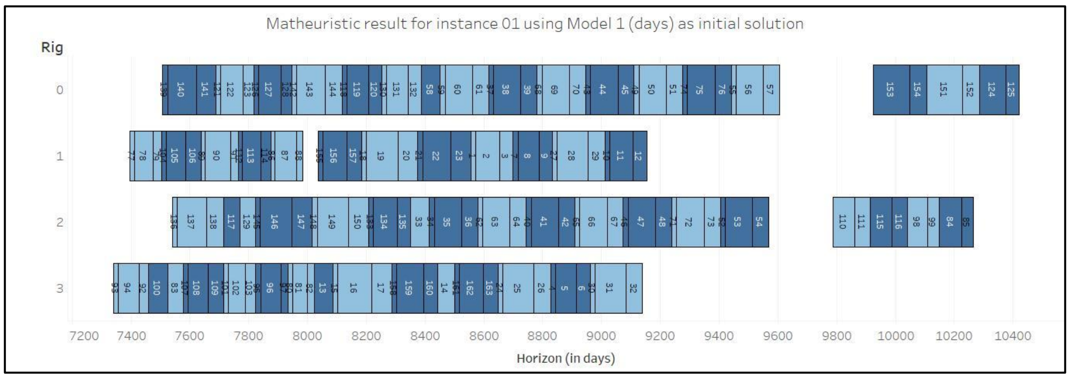

Figure 29. Matheuristic result for instance 01 using Model 1 in day (Source: the author).

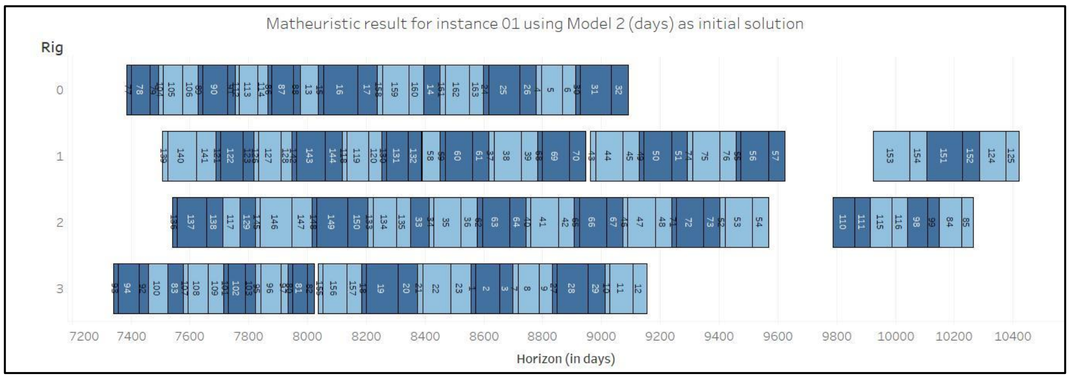

Figure 30. Matheuristic result for instance 01 using Model 2 in days (Source: the author).

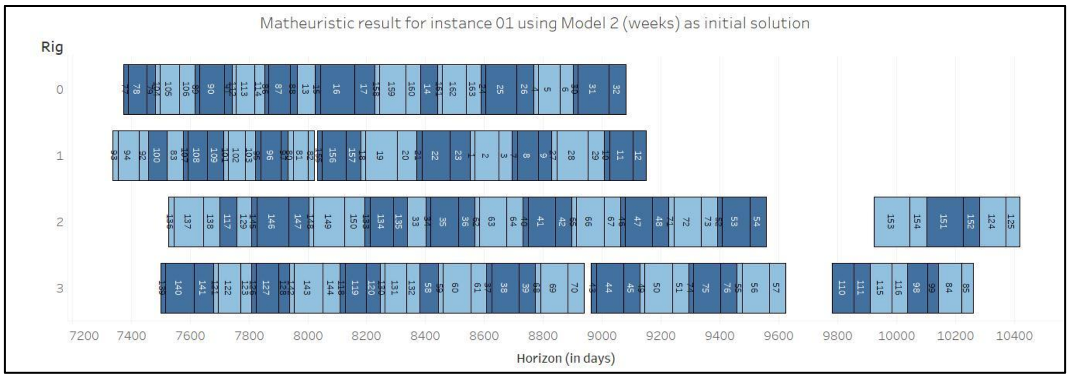

Figure 31. Matheuristic result for instance 01 using Model 2 in weeks (Source: the author)

Meanwhile, Figures 32 and 33 are related with instance 02 solutions.

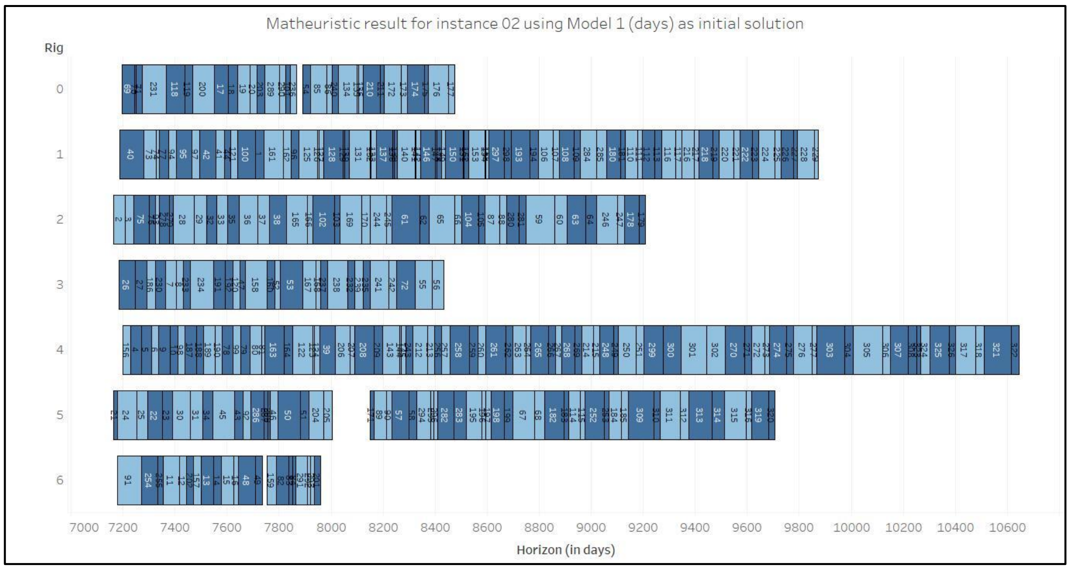

Figure 32. Matheuristic result for instance 02 using Model 1 in days (Source: the author).

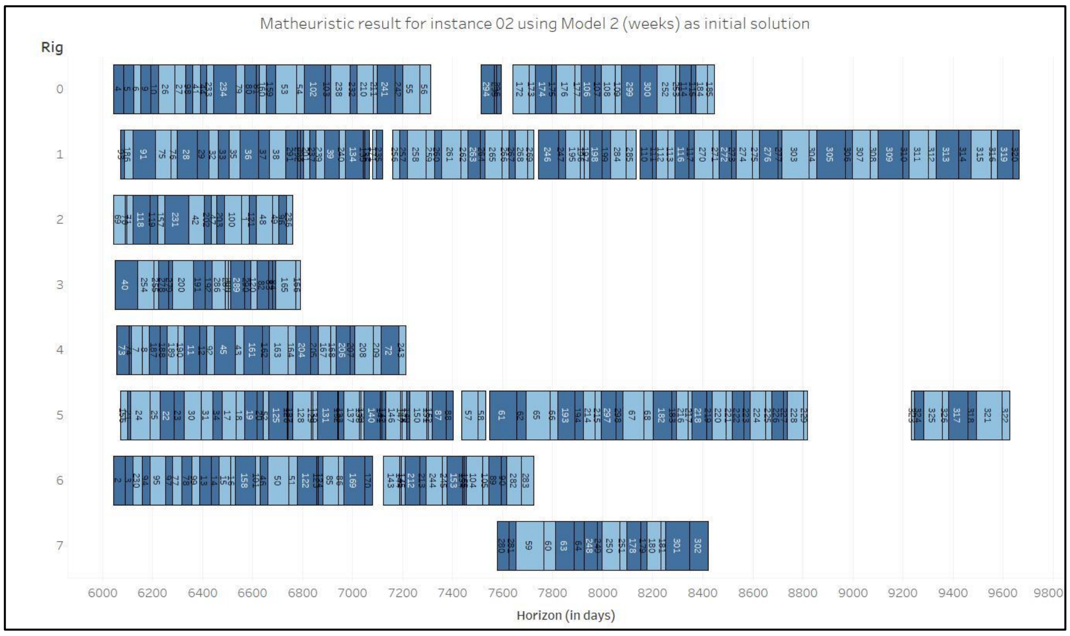

Figure 33. Matheuristic result for instance 02 using Model 2 in weeks (Source: the author).