Pontifícia Universidade Católica
do Rio de Janeiro

Victor Abu-Marrul Carneiro da Cunha

Reprogramação de embarcações de apoio à exploração de petróleo através de uma abordagem metaheurística

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia de Produção do Departamento de Engenharia Industrial da PUC-Rio.

Orientador: Prof. Silvio Hamacher
Co-orientador: Prof. Luciana de Souza Pessôa

Rio de Janeiro
Março de 2017

Victor Abu-Marrul Carneiro da Cunha

Reprogramação de embarcações de apoio à exploração de petróleo através de uma abordagem metaheurística

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia de Produção do Departamento de Engenharia Industrial da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Silvio Hamacher
Orientador
Departamento de Engenharia Industrial - PUC-Rio

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

Prof. Paulo Cesar Ribas
Petrobras

Prof. Simone de Lima Martins
UFF

Prof. Marcio da Silveira Carvalho
Coordenador (a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 13 de março de 2017.

Victor Abu-Marrul Carneiro da Cunha

Graduou-se em Engenharia de Produção pela Universidade Veiga de Almeida do Rio de Janeiro – UVA em 2013. Durante a graduação estagiou em empresa do ramo energético, atuando na área de gestão da logística de medição de energia.

Ficha Catalográfica

Cunha, Victor Abu-Marrul Carneiro da

Reprogramação de embarcações de apoio à exploração de petróleo através de uma abordagem metaheurística / Victor Abu-Marrul Carneiro da Cunha; orientador: Silvio Hamacher; co-orientador: Luciana de Souza Pessôa. – 2017.
109 f. : il. color. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2017.
Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Programação de navios. 3. Logística petrolífera. 4. PLSV. 5. Programação de máquinas paralelas idênticas. 6. Metaheuristíca ILS. I. Hamacher, Silvio. II. Pessôa, Luciana de Souza. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

Aos meus pais, Claudiano e Fádua e aos meus irmãos Michele, Diego e Vyrna.

Agradecimentos

À minha família pela educação, princípios e valores que me foram transmitidos e todo o apoio e carinho incondicionais a mim concedido.

Aos orientadores Luciana Pessôa e Silvio Hamacher, pela atenção, ensinamentos e ótima orientação durante todo o mestrado e desenvolvimento da dissertação.

Aos professores e funcionários do Departamento de Engenharia Industrial da PUC-Rio por todo o conhecimento passado e suporte dado ao longo do mestrado.

À Aline pela paciência e apoio incondicional, acreditando e me fazendo acreditar no meu potencial, antes e durante o mestrado.

Ao João Gabriel por me convencer e acreditar na minha capacidade, me motivando a ingressar no mestrado e a todos os meus amigos pelo suporte e momentos de descontração.

Ao Iuri pelo apoio e tempo despendido ao longo do desenvolvimento do trabalho.

À Janaina, Leonardo, Pedro, Iuri, Tiago, Gabriela, Danuza e demais colegas de trabalho, pela receptividade e colaboração no ambiente de trabalho.

À Leila, Jessica, Janaina, Nathália, Leonardo, Dimas, Igor e demais amigos do mestrado, pela parceria nos estudos e pelos momentos de descontração.

À CAPES, ao TECGRAF e à PUC-Rio, pelos auxílios concedidos e pelo ótimo ambiente de estudo sem os quais este trabalho não teria sido possível.

A todos aqueles que, de alguma forma, contribuíram para este trabalho, os meus sinceros agradecimentos.

Resumo

Cunha, Victor Abu-Marrul Carneiro da; Hamacher, Silvio (Orientador); Pessôa, Luciana de Souza (Co-Orientador). **Reprogramação de embarcações de apoio à exploração de petróleo através de uma abordagem metaheurística.** Rio de Janeiro, 2017. 109p. Dissertação de Mestrado - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

A dissertação aborda um problema real de reprogramação de uma frota de embarcações do tipo PLSV (*Pipe Laying Support Vessel*), responsáveis pelas interligações de poços petrolíferos submarinos. O cronograma de curto prazo dessas embarcações está sujeito à inúmeras incertezas inerentes às operações realizadas, acarretando em ociosidade nas embarcações ou postergações na produção de petróleo, que podem resultar em prejuízo de milhões de reais. Uma metaheurística ILS (*Iterated Local Search*) é proposta para atender a frequente demanda por reprogramações dos PLSVs. O método é composto de uma fase inicial de viabilização, para tratar potenciais inconsistências nas programações. Na sequência, iterativamente, são realizadas perturbações na solução por meio de movimentos de *swap* e aplicada uma busca local baseada na vizinhança *insert*, a fim de fugir de ótimos locais e encontrar soluções que aprimorem o cronograma. Foram feitos experimentos com diferentes parâmetros e critérios do ILS, sendo definidas duas abordagens aplicadas a dez instâncias oriundas de uma programação real de PLSVs. A partir de uma função de avaliação, capaz de medir o impacto operacional na programação, o ILS proporcionou uma melhoria média nos cronogramas acima de 91%, quando comparados aos cronogramas originais. As soluções foram obtidas em um tempo computacional médio de 30 minutos, aderente ao processo da companhia. Em função dos resultados alcançados, o método provou ser uma boa base para uma ferramenta de apoio à decisão para a reprogramação dos PLSVs.

Palavras-chave

Programação de Navios; Logística Petrolífera; PLSV; Programação de Máquinas Paralelas Idênticas; Metaheuristíca ILS; Reprogramação.

Abstract

Cunha, Victor Abu-Marrul Carneiro da; Hamacher, Silvio (Advisor); Pessôa, Luciana de Souza (Co-Advisor). **Rescheduling of oil exploration support vessels within a metaheuristic approach.** Rio de Janeiro, 2017. 109p. Dissertação de Mestrado - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation addresses a real-life rescheduling problem of a Pipe Laying Support Vessels (PLSVs) fleet, in charge of subsea oil wells interconnections. The short-term schedule of these vessels is subject to uncertainties inherent to its operations, resulting in ships idleness or delays in oil production, which may lead to losses of millions of Brazilian *Reais*. A method based on the ILS (Iterated Local Search) metaheuristic is proposed to meet the frequent demand of PLSVs rescheduling. The first step of this method aims to find a feasible initial solution from an incoming schedule with potencial inconsistencies. The following steps consists in, iteratively, performing a perturbation on a solution through swap movements and applying a local search based on the insertion neighborhood, in order to escape from local optimal and find better solutions. Extensive preliminary experiments were conducted considering different ILS parameters setups. The two most performing setups were selected and applied to ten instances of a real PLSV schedule. Taking into account an objective function that measures the operational impact on schedules, the ILS provided an average improvement above 91% in schedules when compared to the original planning. These solutions were obtained in an average computational time of 30 minutes, which fits in the company process. The obtained results showed that the proposed method might be a basis for a decision support tool for the PLSVs rescheduling problem.

Keywords

Ship Scheduling; Offshore Logistics; PLSV; Pipe Laying Support Vessel; Identical Parallel Machine Scheduling; Metaheuristic Iterated Local Search; Rescheduling.

Sumário

Lista de figuras

Lista de tabelas

Lista de quadros

1
Introdução

A exploração e produção de petróleo *offshore* se caracteriza por possuir grande complexidade técnica em sua execução, acarretada por condições climáticas no ambiente marinho e grandes distâncias das plataformas aos poços produtores e à costa marítima. Esses fatores são agravados na exploração do pré-sal, por operar com lâminas d´água superiores a 2.000 metros (Morais, 2013).

Nesse tipo exploratório, a produção de óleo e gás é escoada para as plataformas na superfície por meio de linhas ou dutos de produção interligados aos poços submarinos. Além das linhas de produção, linhas de controle e de injeção de fluidos também são conectadas aos poços (Speight, 2014).

A tarefa de interligar os poços é realizada por embarcações do tipo PLSV (*Pipe Laying Support Vessel*), responsáveis tanto pelo transporte e lançamento dos dutos no leito submarino, quanto pela conexão dos mesmos aos poços e às plataformas (Serpa, 2012).

Em um contexto com alta demanda de interligações a serem realizadas, impactadas por limitações operacionais ou por regras de priorização de atividades, é imprescindível que a frota PLSV disponível seja programada de forma eficiente, objetivando atender essa demanda da melhor forma possível. A programação consiste no desenvolvimento de cronogramas de execução das atividades, baseando-se em regras operacionais e diretrizes gerenciais para sua construção, indicando a embarcação que realizará cada uma das atividades e em que ordem se dará essa execução. Esse cronograma de atendimento serve como base para a previsão do início de produção de cada um dos poços submarinos.

A programação da frota de navios PLSV equivale a um problema de programação de máquinas paralelas, que, de acordo com Mokotoff (2001), consiste em programar n atividades ($1 \leq j \leq n$) em m máquinas ($2 \leq i \leq m$), de modo que cada atividade seja realizada apenas uma vez em uma das máquinas disponíveis. No presente trabalho, os navios equivalem às máquinas e as atividades às interligações dos poços submarinos. As interligações se destacam como função

preponderante do navio, uma vez que o tempo de deslocamento é pequeno quando comparado ao de execução das atividades a ele alocadas, distanciando o problema de uma abordagem de roteamento de veículos. Vernalha *et al*. (2008), Queiroz e Mendes (2012) e Moura (2012) abordaram o problema de programação de embarcações PLSV com base em modelos de programação de máquinas paralelas.

Pinedo (2008) destaca a importância das operações que envolvem programação de recursos em uma companhia, principalmente quando lida com recursos críticos. Este é o caso dos navios PLSV devido ao impacto que essa programação acarreta em inúmeras operações que dela dependem, direta ou indiretamente, além de afetar os objetivos traçados pela empresa.

Esta pesquisa aborda o problema apresentado por uma empresa do ramo de energia que lida com exploração e produção de óleo e gás *offshore* e precisa programar sua frota limitada de navios PLSV a fim de maximizar a sua produção. Os navios são afretados através de contratos de longo prazo com empresas especializadas nas operações realizadas pelas embarcações. O contrato de cada embarcação possui valores diários que podem chegar a US$ 300.000,00 (Sinaval, 2013; Offshore Energy Today, 2013). As interligações se dão na região do pré-sal, onde um poço típico produz cerca de 20 mil barris de petróleo por dia, gerando receita diária de milhões de reais. Esses fatores são determinantes para evidenciar a criticidade desses recursos e a necessidade de que sejam programados de forma eficiente, a fim de evitar atrasos no início da produção.

A programação dos navios PLSV impacta diretamente na produção prevista pela companhia, evidenciando assim o ganho que uma boa programação é capaz de gerar. Atualmente, a programação é feita de forma manual, sem o auxilio de ferramental matemático capaz de auxiliar na proposição de cronogramas, sendo função do programador, por meio das regras de negócio e do conhecimento tácito do processo, encontrar uma programação que satisfaça os critérios estabelecidos e que esteja de acordo com as metas propostas pela empresa.

Semanalmente, é realizado o processo de ajuste do cronograma de curto prazo que corresponde à programação dos três meses subsequentes à semana corrente. Devido aos fatores de incerteza inerentes ao problema, muito do que foi planejado não acontece conforme previsto, trazendo a necessidade de que o cronograma seja refeito para se adaptar a nova realidade, além de atender novas diretrizes impostas pela direção da empresa.

Dessa forma, a presente pesquisa aparece como um importante suporte para o desenvolvimento de uma ferramenta de apoio a decisão que seja capaz de propor ajustes nos cronogramas de curto prazo, com foco na eficiência da utilização dos PLSVs e no atendimento das datas previamente estipuladas. O estudo estende a contribuição na literatura com a aplicação de modelos baseados em máquinas paralelas em casos práticos e na programação de frotas marítimas, divergindo do tratamento mais comumente encontrando, feito com base em roteamento de veículos.

A proposta consiste em identificar as regras aplicadas na etapa do processo interno de reprogramação das embarcações PLSV da companhia estudada, a fim de entender quais as necessidades reais no problema, e, com o auxílio de uma metaheurística ILS (*Iterated Local Search*), propor melhores soluções quando comparadas às que são desenvolvidas atualmente. O intuito é minimizar o impacto causado pelas mudanças indesejadas no cronograma proposto, que afetam não só o curto prazo analisado, mas refletem também em todos os meses posteriores programados.

Os objetivos específicos do estudo podem ser destacados como:

- Mapeamento das regras de negócio aplicadas ao processo de reprogramação da frota marítima disponível;
- Coleta de dados e identificação de padrões em programações anteriores;
- Aplicação do ILS na reprogramação do cronograma, atendendo as regras de negócio previstas, visando minimizar o impacto operacional nas atividades a serem executadas;
- Testes computacionais para validação do método proposto com base em dados reais fornecidos pela empresa estudada.

O presente trabalho está organizado em 7 capítulos, levando em consideração este capítulo de introdução. O Capítulo 2 tem como objetivo contextualizar o leitor sobre a produção e exploração de pétroleo, destacando o papel do navio PLSV, além de detalhar o caso estudado com informações relevantes sobre o processo interno de programação das embarcações realizado pela companhia e os objetivos propostos pelo estudo. No Capítulo 3 é apresentado o problema de programação de navios conforme abordado na literatura, além de uma comparação entre suas aplicações e abordagens de programação de máquinas,

caracterizando o problema dos PLSVs como um caso de programação de máquinas paralelas idênticas. Na sequência são apresentados trabalhos com aplicações práticas e teóricas em programação e reprogramação de máquinas paralelas, incluindo alguns que tratam a programação de PLSVs, além de trabalhos que aplicam a metaheurística ILS em problemas de programação de máquinas, enriquecendo a contextualização acerca do tema. O Capítulo 4 apresenta a metodologia do estudo, indicando a classificação da pesquisa e evidenciando as etapas a serem seguidas no desenvolvimento do trabalho. O Capítulo 5 aborda os métodos propostos para a reprogramação dos cronogramas dos PLSVs de forma detalhada, sendo estes aplicados e avaliados no Capítulo 6. O Capítulo 7 apresenta as considerações finais do estudo.

2
Descrição do Problema

Neste capítulo será apresentado o ambiente no qual a operação dos navios PLSV ocorre, descrevendo características do processo de produção de óleo e gás onde essas embarcações atuam, destacando sua importância na operação como um todo e nos resultados de produção previstos pela empresa onde o estudo foi realizado. Será também detalhado o processo da companhia para o desenvolvimento da programação dessas embarcações, indicando as principais regras de negócio e características do problema a serem consideradas durante a elaboração de seus cronogramas de execução de atividades.

2.1
Estudo de Caso

O presente estudo foi realizado em uma empresa brasileira do ramo energético, atuante na exploração e produção de óleo e gás *offshore*, abordando o seu processo interno de desenvolvimento de cronogramas de execução de atividades, referentes à interligação de poços petrolíferos submarinos na região do pré-sal. A operação de interligação desses poços é realizada por embarcações do tipo PLSV, e destaca-se como uma das mais importantes dentre aquelas que circundam a exploração e produção de óleo e gás *offshore*, por se tratar da última etapa a ser realizada antes que um determinado poço entre em operação. Um exemplo de uma embarcação PLSV é mostrado na Figura 1.

Figura 1 - Embarcação PLSV. Fonte: Barboza (2015).

Os navios do tipo PLSV se caracterizam por serem altamente especializados, possuindo em seu convés equipamentos complexos e tecnológicos utilizados em suas operações, como: cestas e carretéis de armazenamento de linhas, guindaste de movimentação de carga, rampa de lançamento de linhas, veículos de operação remota, entre outros. Devido à complexidade e ao elevado número de tarefas a serem realizadas, essas operações demandam grande tripulação a bordo capacitada a executá-las (Wollner e Rosa, 2015). Todos esses fatores evidenciam o alto custo envolvido em uma operação de interligação realizada por um navio PLSV, expondo assim a importância no uso eficiente dos recursos disponíveis.

Os poços submarinos podem ser produtores ou injetores. Em geral cada poço produtor possui três atividades a ele associadas, que são: interligação da linha de produção, interligação da linha de controle e interligação da linha de injeção. Nos poços injetores a linha de produção não se faz necessária, tendo assim por padrão duas atividades a eles associadas. Porém, em alguns casos essas atividades podem ser desmembradas por questões gerenciais ou operacionais, resultando em um maior número de atividades vinculadas a um determinado poço. De acordo com Labanca (2005) e Bai e Bai (2010), as linhas de produção são responsáveis por escoar o fluido produzido nos poços produtores, podendo ser dos tipos flexíveis ou rígidas. Os mesmos tipos de linhas são utilizados para a injeção de fluidos, que tem como objetivo aumentar a pressão nos reservatórios, seja no próprio poço produtor onde a injeção é feita ou em poços adjacentes quanto se trata de um poço exclusivamente injetor, facilitando o fluxo da produção. O controle é feito pela linha "Umbilical", responsável pelo acionamento de válvulas, aquisição de dados e transporte de químicos que otimizam o escoamento da produção. Alguns carretéis com linhas flexíveis são exibidos na Figura 2.

Figura 2 - Linhas Flexíveis. Fonte: National Oilwell Varco (2012).

As interligações são realizadas entre os poços e a UEP (Unidade Estacionária de Produção), unidade responsável pelo recebimento do fluido produzido nos poços submarinos, tratamento desse fluido e exportação para o meio que transportará o material gerado (Labanca, 2005).

Além das atividades de interligação dos poços, o PLSV também realiza a instalação de *manifolds* submarinos, que de acordo com Bai e Bai (2010), são alojados no leito marítimo com o intuito de combinar a produção ou injeção de fluidos de diversos poços, simplificando o arranjo submarino e consequentemente diminuindo a utilização de dutos e linhas.

A Figura 3 apresenta um exemplo de um sistema submarino de produção de óleo e gás, com poços e *manifolds* interligados a uma UEP.

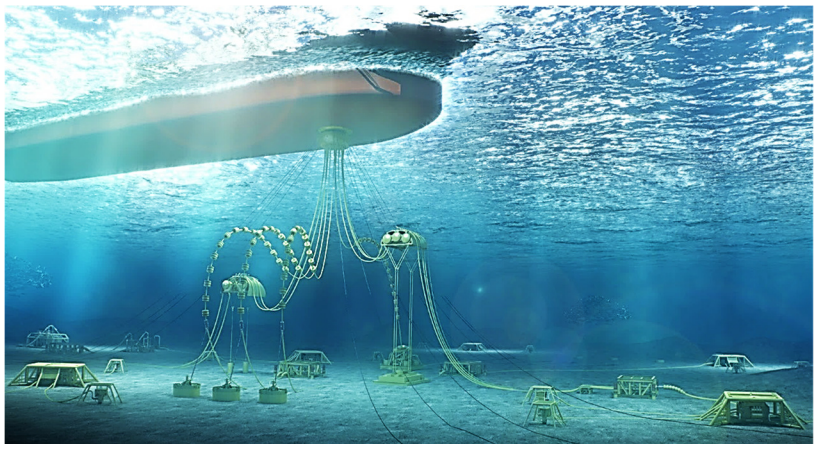


Figura 3 - Sistema submarino. Fonte: MG Vowgas (2016).

A entrada de um novo poço incrementa a produção da empresa, impactando positivamente nas metas propostas para cada ano, que se baseiam dentre outros fatores, nas datas estipuladas previamente para a finalização de cada poço. Por se tratar da última etapa no processo de entrada em produção de um novo poço não explorado, a interligação suscita maior interesse gerencial, estando esta operação sempre no foco das atenções.

Desta forma, programar as embarcações disponíveis de forma eficiente, maximiza a precisão quanto à previsão de produção anual da empresa, aumentando consequentemente o seu retorno financeiro.

A partir de reuniões realizadas com a área responsável pela programação das embarcações PLSV foram delineadas as características e regras aplicadas durante a programação da frota. Esses conceitos são expostos a seguir nas Seções 2.1.1 e 2.1.2 e dizem respeito à forma de abordar o problema por parte da empresa estudada.

2.1.1
Características da Programação de Navios PLSV

A ocupação do convés do navio é o principal fator limitante do número de operações em sequência que podem ser realizadas por uma mesma embarcação PLSV. Isso ocorre pelo fato de que, antes de qualquer operação ou grupo de operações serem realizadas, o material referente a cada atividade, sejam linhas ou *manifold*, precisa ser carregado na embarcação, ocupando seu espaço interno limitado, fazendo com que a mesma tenha que retornar à base onde o material é armazenado após a realização de um conjunto de operações, para realizar um próximo carregamento. Surge assim como característica do problema, o conceito de "viagem" realizada pelos navios. Uma viagem completa de um navio consiste em: carregar as linhas na base, navegar para o local de execução das atividades, executar as atividades a ele alocadas e navegar de volta à base. O tempo de navegação inicial, juntamente com o tempo de carregamento das linhas podem ser caracterizados como tempo de *setup*. A duração da navegação de retorno para a base utilizada como padrão nas programações é de um dia, sendo definida assim com base em dados históricos. Além disso, os tempos de navegação entre atividades não são considerados, devido às curtas distâncias entre os poços na região do pré-sal. A Figura 4 exibe uma viagem com sua composição padrão com: Carregamento (C), Navegação (N) e as n atividades a realizar.

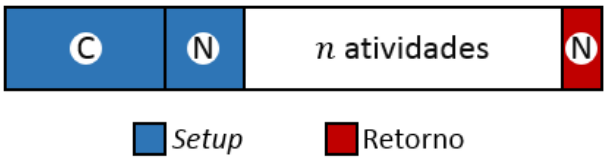


Figura 4 - Exemplo de Viagem. Fonte: Elaboração própria.

No caso dos PLSVs o tempo de *setup* não depende da sequência das atividades e nem do navio que realizará as atividades, sendo estas durações estipuladas pelos tipos de atividades a serem realizadas em cada viagem do navio, e definidas como:

- 4 dias: para viagens contendo apenas atividades de instalação de *manifolds.*
- 6 dias: para viagens contendo apenas atividades de interligação de poço.
- 9 dias: para viagens com atividades combinadas de instalação de *manifold* e interligação de poço.

Os tempos de *setup* independem do número de atividades programadas em cada viagem e têm a duração especificada a partir de uma estimativa feita pela companhia com base em tempos médios históricos das etapas de carregamento e navegação. As durações são diferentes por existirem duas bases distintas para o carregamento dos materiais. Em uma delas é feito o carregamento de *manifolds* enquanto na outra são carregadas as linhas de interligação. Considera-se também, caso uma mesma viagem possua atividades dos dois tipos, o tempo de navegação entre as duas bases, sendo este tempo incluído na duração de *setup* especificada.

As atividades possuem datas de liberação, baseadas em previsões de entrega dos materiais por parte dos fornecedores, caracterizando assim o *setup* como do tipo não antecipatório, ou seja, o mesmo não pode ocorrer antes da data de liberação das atividades, dado que o início do *setup* nesse caso representa a etapa de carregamento e para que seja realizado depende da disponibilidade das linhas na base. Dessa forma, o *setup* só pode ser programado para iniciar em um instante de tempo no qual todas as atividades programadas em uma mesma viagem estejam liberadas.

Devido à baixa frequência de ocorrência das instalações de *manifold* e da necessidade de adaptações no navio para torná-lo apto a realizá-las, apenas uma das embarcações da frota, que foi submetida ao processo de ajuste necessário, é capaz de atender a essa demanda.

Em uma mesma viagem um PLSV pode realizar atividades de diferentes poços. Para que um poço seja considerado como finalizado e que consequentemente possa iniciar a produção de óleo e gás, todas as atividades referentes a esse mesmo poço devem ser finalizadas, de modo que, o instante de término da última dessas atividades representa o instante de conclusão do poço.

Além das atividades de interligação de poços e de instalação de *manifold*, paradas preventivas para manutenção também são programadas nos PLSVs. As paradas são programadas como se fossem atividades, possuindo datas limites de início e fim para sua realização, porém, não seguem a regra da viagem, dado que, a mesma não necessita de carregamento de material e nem de navegação de deslocamento.

A execução das atividades pelos PLSVs apresenta particularidades importantes que restringem a programação. Dentre as restrições a serem observadas, se destacam: datas de liberação das atividades, janela de disponibilidade de cada embarcação, ocupação do navio, elegibilidade de cada

embarcação em realizar cada uma das atividades e datas limite para finalização de cada poço.

A liberação das atividades, como destacado anteriormente, restringe o problema quanto à data mínima para início do *setup* de uma atividade ou de um grupo de atividades em uma mesma viagem. A janela de disponibilidade de cada embarcação indica ao programador em quais instantes de tempo o navio pode ter atividades a ele alocadas ou não. A ocupação do navio, também abordada anteriormente nesse capítulo, determina o limite de atividades que podem ser realizadas em uma mesma viagem, tendo como forma de controle a taxa de ocupação de cada atividade dada em valores percentuais, de forma que, o somatório das taxas de todas as atividades selecionadas em uma mesma viagem não pode ultrapassar os 100% de ocupação disponível em cada navio. A elegibilidade da embarcação indica se um navio está ou não habilitado por questões técnicas a executar uma determinada atividade, fazendo com que cada atividade possa então ser atendida apenas por um subconjunto de navios da frota disponível. Com exceção da data limite para finalização dos poços, as demais restrições citadas são restrições rígidas do processo que não podem ser violadas, tornando o cronograma proposto inviável em caso de descumprimento de qualquer uma delas. Com relação às datas limites, estas servem como diretrizes para a elaboração do cronograma que deve tentar se aproximar ao máximo dessas datas estipuladas pela alta direção como datas ótimas de entrada em produção de cada poço considerado.

A frota disponível é fixa, oriunda de contratos firmados com empresas prestadoras desse tipo de serviço e detentoras das embarcações. Os contratos são firmados por longos períodos, de forma que a embarcação fica disponível para a contratante, empresa objeto desse estudo, por tempo integral enquanto o contrato encontra-se vigente, ficando indisponível apenas quando manutenções preventivas estão sendo realizadas pela contratada.

Desta forma, a programação consiste em sequenciar viagens para cada um dos PLSVs, cumprindo todas as restrições e objetivos acima destacados. Atualmente, uma área específica dentro da empresa é responsável pela elaboração dos cronogramas por navio, feitos manualmente pelos programadores, com o auxílio apenas de um sistema para a montagem visual do cronograma, sem ferramental matemático ou inteligência computacional aplicada que auxilie na decisão de onde alocar cada atividade e em como sequenciá-las. Na Figura 5 é

demonstrado um exemplo de cronograma com 3 navios PLSV, contendo 11 atividades distribuídas em 6 viagens programadas.

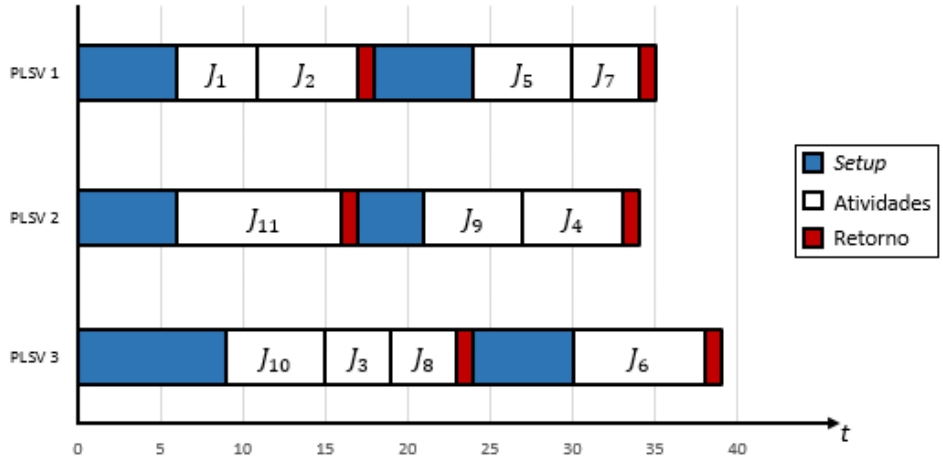


Figura 5 - Cronograma de viagens em 3 PLSVs. Fonte: Elaboração própria.

2.1.2 Desenvolvimento da Programação

A área responsável pela programação da frota PLSV na região do pré-sal subdivide o processo em duas macro etapas que são: desenvolvimento do cronograma anual e manutenção semanal do cronograma em execução. O desenvolvimento do cronograma anual é realizado no último trimestre de cada ano vigente, onde são traçadas as metas e objetivos da companhia para o ano seguinte. As diretrizes gerenciais servem como base para o planejamento das embarcações, programando-as de forma a atender ao máximo as datas propostas para finalização de cada poço, respeitando a capacidade operacional disponível.

A programação final define as metas a serem seguidas pela empresa ao longo do ano, tendo em vista que o término das etapas de interligação habilita os poços para que entrem em operação e, consequentemente, gerem retorno financeiro. A versão final do cronograma serve como base para os ajustes a serem realizados na programação ao longo do ano, na fase de manutenção.

A manutenção do cronograma é feita semanalmente, na qual, os três meses subsequentes à semana avaliada são ajustados para se adaptarem as mudanças ocasionadas pelas incertezas inerentes ao processo. Na última semana de cada mês, além dos três meses do ajuste padrão, o restante do cronograma até o final do ano também é revisado, adequando-o às modificações propostas. As principais incertezas que distanciam o cronograma executado do cronograma planejado se originam tanto de problemas relacionados aos navios quanto às atividades. Aquelas

que se referem aos navios são: paradas não programadas por quebra, atrasos na documentação de liberação para a realização das operações e condições meteorológicas que impeçam o navio de realizar alguma atividade a ele alocada. Com relação às atividades, as incertezas existentes são: variação nas durações das interligações dos poços e mudança em datas de liberação de determinadas atividades.

Atualmente, as durações das atividades são estimadas pela companhia com base em dados históricos, porém, existem características particulares que afetam os tempos reais de execução necessários, seja uma variação causada pelo navio ou tripulação utilizada na operação ou por variações naturais existentes em qualquer tipo de operação. Essas durações podem ter seu tempo total incrementado ou decrementado. Com exceção da quebra de navio e da mudança nas datas de liberação das atividades, o impacto ocasionado pelos demais fatores acarreta no deslocamento do que foi planejado, antecipando ou atrasando as atividades posteriores às que tiveram seus tempos de execução alterados. O sistema utilizado internamente pela companhia, no qual o cronograma é desenvolvido, atualiza essas durações e desloca as atividades afetadas por essas mudanças automaticamente.

A documentação de liberação do navio é a permissão emitida pelos órgãos competentes que fiscalizam a operação, para que uma viagem possa ser executada. Quando o navio fica inativo por questões meteorológicas, demora na liberação de documentação ou qualquer outro motivo, esse tempo extra é acrescido na tarefa em execução, seja carregamento, navegação, interligação ou instalação de *manifold*. Sendo assim, o ajuste feito pelo sistema interno é o mesmo, já que o que é de fato alterado é a duração das tarefas.

As alterações inesperadas impactam nas restrições do problema e nos objetivos traçados no seu planejamento inicial, modificando as datas de início de produção de cada poço e descumprindo datas de liberação das atividades.

Desta forma, este estudo pretende servir de base para o desenvolvimento de uma ferramenta de apoio à decisão que seja capaz de reprogramar os três meses futuros do cronograma de atividades dos navios PLSV afetados por alterações indesejadas, focando nas datas originalmente propostas para o término das interligações de cada poço e cumprindo as restrições do problema. A ideia é viabilizar o novo cronograma a ser seguido, tentando afetar minimamente o que foi planejado. Tendo isso em vista, o ajuste deve, além de atender as datas de liberação

e ajustar o término dos poços, se preocupar com o impacto causado nas operações futuras.

Como as atividades avaliadas são realizadas em curtíssimo prazo e dependem de documentação que libere um determinado navio para executá-las, os navios passam então a ser agrupados em famílias, onde cada família representa a empresa detentora da embarcação, facilitando que a documentação seja agilizada sem maiores problemas. Dessa forma, uma atividade só pode ser movimentada para algum navio de mesma família daquele no qual ela encontra-se alocada. O termo "família" foi convencionado durante o desenvolvimento desse trabalho e não corresponde a um termo comercial utilizado pela companhia.

Outro ponto importante é identificar o quão distante do final original o trecho de cronograma avaliado se encontra, ou seja, quantos dias devem ser acrescidos em cada navio para que o mesmo grupo de atividades seja programado, dado que, quanto mais dias forem incrementados, maior será o impacto nos meses subsequentes, não avaliados no ajuste que está sendo realizado. Esses fatores devem ser levados em conta na hora de desenvolver um novo cronograma operacional que fique o mais próximo do original possível.

Tendo em vista o alto custo operacional envolvido e a constante cobrança gerencial quanto às operações da frota PLSV, períodos de ociosidade entre as viagens são indesejados, dado que custos contratuais são incorridos mesmo que um navio esteja ocioso, sem atividades a realizar.

Desta forma, o problema de reprogramação do cronograma de PLSV a ser seguido pelo presente trabalho tem como objetivo reprogramar *n* atividades de interligação de poços petrolíferos submarinos alocadas a *m* navios do tipo PLSV disponíveis, tendo como função multicritério de avaliação três componentes a serem minimizados, que são: atraso no término dos poços no comparativo entre o cronograma atual e o original, número de dias excedentes utilizados nos navios para a realização das atividades reprogramadas e a ociosidade total dos navios entre as viagens a serem realizadas. As principais características do problema tratado são:

- Uma atividade só pode ser executada em um único navio, uma única vez;
- Um navio não pode executar mais de uma atividade por vez;
- Não existem restrições de precedência entre as atividades;

- Uma vez iniciada uma atividade, esta não pode ser interrompida para continuar o processamento de seu tempo remanescente em um instante futuro;
- As durações de cada atividade são conhecidas e independem da programação definida;
- Cada atividade, com exceção das paradas preventivas e instalação de *manifolds*, está vinculada a um poço;
- Para que o poço seja considerado como finalizado, todas as atividades a ele vinculadas devem ser finalizadas;
- Nem todas as embarcações podem executar todas as atividades;
- Um tempo de *setup* deve ser definido para um conjunto de atividades;
- O *setup* é do tipo não antecipatório, ou seja, só pode ser realizado quando a atividade já se encontra liberada;
- Cada navio possui um intervalo de tempo de disponibilidade;
- Não são consideradas probabilidades de quebra de navios, podendo ser utilizado todo o horizonte dentro da janela de disponibilidade de cada embarcação;
- O número de atividades é fixo;
- O número de navios é fixo;
- As datas de liberação de todas as atividades são conhecidas;
- O tempo de deslocamento (navegação) entre os poços é desconsiderado;
- O tempo de navegação de retorno para a base é fixo, com um dia de duração;
- As datas de compromisso para conclusão de cada poço são conhecidas;

3
Referencial teórico

Neste capítulo é apresentada inicialmente uma visão geral sobre a programação de frotas marítimas, destacando suas modalidades operacionais existentes. Em seguida, é exposto um comparativo entre problemas de programação de máquinas e problemas de roteamento de veículos, com trabalhos que apontam as suas diferenças e semelhanças, sendo então apresentados alguns trabalhos de programação de embarcações PLSV feitos com base em modelos de programação de máquinas paralelas. Na segunda parte do capítulo a programação de máquinas é abordada, com ênfase em modelos com máquinas idênticas, tema principal do presente trabalho. Nesse contexto serão apresentados diversos trabalhos com aplicações teóricas e práticas na resolução de problemas de programação e problemas de reprogramação em máquinas paralelas idênticas, deste modo, serão identificadas as principais técnicas aplicadas e os segmentos estudados sobre o tema. Ainda no contexto de problemas de programação de máquinas, são apresentados na sequência do capítulo, trabalhos que aplicaram a metaheurística *Iterated Local Search*, método utilizado nesse trabalho, para a resolução desse tipo de problema. O capítulo é finalizado com uma análise sobre os trabalhos existentes na literatura, classificando o problema abordado nesse trabalho, destacando suas particularidades.

3.1
Programação de Navios

As frotas marítimas são responsáveis por transportar aproximadamente 80% do que é comercializado internacionalmente, despontando como principal modal para o comércio em escala global. Apesar de apresentar queda na taxa de crescimento nos últimos três anos, no início de 2015 a frota mundial era composta de mais de 89.000 navios com capacidade total de carga de aproximadamente 1,75 bilhões de toneladas (UNCTAD, 2015).

Apesar do aumento da demanda por transporte, a aquisição de um navio requer um alto investimento inicial, além dos altos custos diários de operação. Fica evidente a importância de uma programação adequada desses recursos, de forma a otimizar a sua

utilização e consequentemente aumentar a sua produtividade e sua capacidade de gerar lucros (Christiansen *et al.,* 2007).

Lawrence (1972) subdivide os problemas de programação de navios em três modalidades de operação: *liner*, *tramp* e *industrial*.

Na modalidade *liner*, as rotas são fixas e previamente estipuladas, de modo similar a um sistema de linhas públicas de ônibus (Christiansen *et al.,* 2013). O lucro dos operadores dessa modalidade é influenciado pelas rotas propostas, que impactam diretamente na qualidade do serviço prestado no que diz respeito à frequência dos navios, tempos de deslocamento, confiabilidade, entre outros fatores. (Ronen, 1983).

Empresas que operam na modalidade *tramp*, trabalham com demandas de transporte, similares a um serviço de táxi. Em geral essas empresas possuem uma gama de contratos firmados que precisam ser cumpridos e visam aumentar o lucro com transportes avulsos (Christiansen *et al.,* 2004).

Os operadores da modalidade *industrial* possuem a carga a ser transportada e a frota marítima, focando assim na minimização de seus custos operacionais com transporte e no atendimento da demanda proposta. Nos casos em que a capacidade da frota é insuficiente para atender a demanda, esses operadores precisam fretar navios adicionais (Ronen, 1993).

Algumas importantes revisões sobre o tema são encontradas na literatura. Com intervalos de aproximadamente dez anos entre cada publicação, a primeira delas foi publicada por Ronen (1983). Dando continuidade ao trabalho, Ronen (1993) publicou a seguinte, abordando os anos entre 1982 e 1992. Christiansen *et al*. (2004) analisaram a década seguinte e no último dos trabalhos de revisão publicados sobre o tema até então, Christiansen *et al.* (2013) destacam o crescimento do assunto na literatura, que praticamente dobrou em quantidade de publicações década após década.

Apesar da programação de embarcações PLSV, tema abordado nesta pesquisa, não tratar de transporte de carga e sim de apoio às operações *offshore*, Mendes (2007) salienta que este tipo de problema pode ser enquadrado na modalidade *industrial*, tendo em vista que trata-se de uma frota fixa de embarcações pertencente a uma empresa, que precisa programar seus recursos disponíveis de forma eficiente, para que atendam a uma demanda de atividades previamente definidas, a fim de minimizar seus custos operacionais, maximizando sua produtividade.

Desta forma, para um melhor entendimento dos conceitos específicos da programação de recursos, uma visão geral acerca do tema pode ser encontrada no

Apêndice I do presente trabalho, onde são identificados os principais ambientes de programação. Na literatura, os recursos são generalizados como máquinas e tratados com modelos específicos para problemas de programação de máquinas. Estes modelos lidam com a alocação e o sequenciamento de atividades a esses recursos de forma que sejam otimizadas as programações com base em um objetivo específico definido previamente. Em contrapartida, os problemas de programação de navios são abordados de forma geral através de modelos de roteamento de veículos e fluxo em redes, com o foco dos navios sendo o de transporte de carga, visando-se assim a minimização com os custos de deslocamento.

Os problemas de programação de máquinas e o roteamento de veículos guardam importantes semelhanças entre si, com comparativos abordados em alguns trabalhos. Além de analisar as similitudes existentes, aplicações práticas envolvendo transformações entre os modelos são propostas. A seguir são expostos alguns trabalhos que realizam comparativos entre os modelos, sendo apresentados na Seção 3.1.2 desse capítulo trabalhos que abordam o problema de programação de PLSVs através de modelos de programação de máquinas, mais especificamente modelos com máquinas em paralelo, abordagem também utilizada no presente trabalho, identificada durante o desenvolvimento do trabalho como a que melhor se adapta ao problema estudado.

3.1.1 Problemas de Programação de máquinas e roteamento de veículos

Beck *et al.* (2002) propõem duas reformulações para modelos distintos de programação de máquinas. A primeira transforma um modelo de roteamento de veículos em um problema de *open shop* e a segunda transforma no sentido oposto um modelo *job shop* de programação de máquinas em um modelo de roteamento, a fim de analisar a qualidade da solução transformada com relação ao uso do roteamento de veículos clássico.

Em Beck *et al.* (2003) é apresentado um estudo sobre as diferenças entre modelos *job shop* de programação em máquinas e modelos de roteamento de veículos, caracterizando cada método e analisando o impacto na qualidade da solução quando um problema de roteamento é tratado como *job shop.* A transformação na modelagem de um tipo de problema para outro é demonstrada e aplicada a instâncias propostas pelo autor, e a qualidade das soluções geradas por essas técnicas são comparadas em função de variações no tempo de execução computacional.

Kouki *et al*. (2007) traçam um paralelo entre os problemas de roteamento de veículo e modelos *flexible job shop*, com o intuito de propor *lower bounds* para problemas de roteamento de veículos, através da aplicação de modelos de *flexible job shop*. De forma análoga, ambos os problemas tentam propor uma alocação e sequenciamento adequados, seja para a realização das atividades ou para a visita dos clientes no caso do roteamento, de forma a minimizar o término da operação.

De acordo com Lam e Xing (1997), modelos de programação de máquinas paralelas idênticas, com *setup* dependentes da sequência na qual as atividades são realizadas, se equivalem a modelos de roteamento de veículos, quando o número de veículos é fixo e conhecido. Cada cidade i a ser visitada é análoga a uma atividade a ser programada nas máquinas, enquanto que os tempos de *setup* s_{ij} entre uma atividade e outra, acrescidos do tempo de processamento p_i da atividade i, correspondem a distância entre essas cidades.

3.1.2 Programação de PLSV

Apesar das adaptações existentes e similaridades entre abordagens de roteamento e alguns problemas específicos de programação de máquinas, a programação de PLSV no presente trabalho apresenta uma maior similaridade com os problemas de programação de máquinas paralelas, onde a atividade preponderante do navio é a realização das atividades de interligação dos poços submarinos a ele alocadas e não a de transporte, tarefa principal na maioria dos trabalhos sobre programação de navios encontrados na literatura. As durações de deslocamento são pequenas quando comparadas aos tempos de execução das atividades. Além disso, devido à proximidade dos campos, os tempos de deslocamento entre uma atividade e outra são desconsiderados. Esses fatores corroboram para que o problema seja tratado como um caso de programação de máquinas paralelas. Para o problema especificamente abordado nesse trabalho, os navios são considerados idênticos, tendo em vista que, de acordo com a literatura, esse tipo de configuração corresponde ao fato de não existir variação nas durações das atividades entre os navios, ou seja, a duração é uma característica inerente a própria atividade.

Alguns trabalhos abordaram o tema de programação de navios PLSV com modelos de programação de máquinas paralelas.

Vernalha *et al*. (2008) aplicaram métodos exatos com uso de formulações mistas inteiras para tratarem o problema de programação com foco na minimização do *total weighted tardiness* ($\sum w_j T_j$). Dois métodos foram utilizados, o primeiro com base em

modelos clássicos de programação de máquinas paralelas e o segundo uma versão simplificada com base em geração de colunas, onde o modelo decide qual rota usar para cada embarcação a partir de todas as combinações de rotas previamente geradas. A aplicação se deu em uma instância simplificada, atingindo o ótimo em ambos os métodos propostos.

Queiroz e Mendes (2012) trataram um problema com máquinas paralelas não relacionadas, representado por uma frota heterogênea de navios, com datas de liberação das atividades e tendo o *total weighted tardiness* ($\sum w_j T_j$) como critério de avaliação a ser minimizado. Foi aplicada uma metaheurística GRASP com *Path Relinking* para resolução do problema, onde as soluções obtidas foram comparadas com *lower bounds* gerados por meio de um modelo de programação inteira com base em geração de colunas.

Moura (2012) aplicou uma metaheurística VNS (*variable neighborhood search*) partindo de soluções construídas com base em lista de prioridade, para um problema com frota homogênea com o objetivo de minimizar o atraso na entrada em produção dos poços de petróleo. Como base de comparação para as soluções da metaheurística, *lower bounds* foram obtidos com a aplicação de relaxação lagrangena, liberando um número maior de navios, tornando assim possível que o atendimento de todas as atividades fosse realizado até a sua data limite.

3.2
Programação de Máquinas Paralelas Idênticas

Com o objetivo de entender melhor o problema de programação em máquinas paralelas idênticas, alguns trabalhos recentes acerca do tema foram levantados e são apresentados nesta seção, agrupados de acordo com o método de resolução utilizado. Diversas abordagens são propostas no desenvolvimento das programações, utilizando-se de métodos exatos, heurísticas e metaheurísticas, com aplicações que vão desde modelos puramente teóricos à casos práticos.

3.2.1
Métodos Exatos

Os métodos exatos são algoritmos computacionais capazes de encontrar a solução ótima para problemas combinatórios, cumprindo restrições inerentes ao problema, minimizando ou maximizando, de acordo com o objetivo proposto, o critério de avaliação utilizado (Nocedal e Wright, 2006).

Para problemas de programação inteira, como nos casos de programação de máquinas, a complexidade computacional exigida tende a tornar impossível seu

tratamento em um tempo viável, conforme o problema aumenta de porte. Dessa forma, se faz necessária a utilização de métodos que dividam o problema em partes, facilitando assim a sua resolução. O método *branch and bound* se destaca como um dos mais utilizados nesse contexto e consiste em tratar de forma iterativa o problema, resolvendo-o de forma linear, eliminando as soluções fracionárias encontradas e delimitando o espaço de soluções viáveis (Wolsey, 1998).

Mellouli *et al*. (2009) abordam o problema de indisponibilidade das máquinas devido a manutenções preventivas, onde um único intervalo de indisponibilidade dentro do horizonte de planejamento é considerado. Apesar da equivalência das máquinas, a duração da parada pode variar devido a particularidades inerentes à manutenção necessária em cada uma delas. São utilizados sete métodos para o tratamento do problema: quatro formulações de programação inteira mista, um modelo de programação dinâmica e dois modelos *branch and bound* com limites definidos por heurísticas construtivas baseadas na regra SPT (*Shortest Processing Time*), tendo como objetivo a minimização do *total completion time* ($\sum C_j$).

Su (2009) desenvolveu um algoritmo *branch and bound* para um problema com datas de compromisso (d_j) das atividades e elegibilidade (M_j) das máquinas, objetivando minimizar o *total completion time* ($\sum C_j$). O trabalho utiliza instâncias aleatórias geradas pelo próprio autor e propõe a elaboração de limites prévios a aplicação do *branch and bound*. O limite inferior proposto utiliza um modelo inteiro para o problema de alocação que desconsidera o sequenciamento. Para o limite superior, uma heurística baseada na heurística SPT (*Shortest Processing Time*) foi desenvolvida, que além de alimentar o modelo se mostrou eficaz em gerar boas soluções de forma rápida, quando comparadas às soluções do algoritmo *branch and bound*.

3.2.2 Heurísticas e Metaheurísticas

As heurísticas constituem métodos de obtenção de soluções para problemas combinatórios em um tempo computacional razoável, porém, sem garantir a otimalidade da solução obtida. A importância da utilização desse tipo de método se dá quando os métodos exatos não são capazes de resolver um determinado problema devido ao seu porte (Pearl, 1984; Reeves, 1993).

As metaheurísticas são métodos que integram procedimentos de buscas locais por melhores soluções e etapas construtivas, de forma a expandir a busca dentro do espaço de solução viável, visando fugir de soluções ótimas locais. O objetivo é que, a partir de

algoritmos robustos, problemas de grande porte possam ser resolvidos mais rapidamente, podendo estes, serem aplicados aos mais variados problemas (Talbi, 2009; Gendreau e Potvin, 2010).

A seguir são expostos alguns trabalhos com aplicações de heurísticas e metaheurísticas em problema de programação de máquinas paralelas idênticas, a fim de que se identifique algumas das estratégias utilizadas e métodos propostos para a resolução desse tipo de problema na literatura.

Biskup *et al*. (2008) propõem quatro heurísticas baseadas em listas, considerando a natureza paralela do problema, de forma que no instante de alocação é avaliado o conjunto de *m* atividades prioritárias a designar para as *m* máquinas disponíveis, para tratar o problema de minimização do *total tardiness* ($\sum T_j$). Os autores usam o método congruente aleatório, proposto por Lehmer (1951) e os resultados obtidos foram comparados com a resolução de um modelo de programação inteira mista proposto pelos autores para instâncias pequenas e também com as três melhores heurísticas da literatura na época do estudo, para instâncias maiores.

Xu *et al*. (2009) tratam a indisponibilidade de máquinas em um problema com duas máquinas, para os casos *off-line* e *online*. Apenas uma das máquinas possui períodos de indisponibilidade, enquanto a outra permanece disponível durante todo o horizonte. O objetivo é a minimização do *makespam* (C_{max}), sendo testadas duas heurísticas construtivas da literatura, com bons resultados para problemas similares, a fim de avaliar o rendimento delas para esse problema que até a publicação do artigo, de acordo com os autores, não havia sido abordado.

Zhao *et al*. (2011) abordam o mesmo problema, porém, apenas um período de indisponibilidade em uma das máquinas é considerado. O objetivo é minimizar o *total weighted completion time* ($\sum w_j C_j$), considerando que uma atividade interrompida pela indisponibilidade deve ser reiniciada, não permitindo assim que a mesma continue de onde parou. O problema é resolvido com uma heurística proposta pelos autores, que indicam que o mesmo pode ser facilmente adaptado para problemas com mais de duas máquinas bem como para o problema de máquinas paralelas uniformes.

Liao *et al*. (2012) estudam o problema com *setup* por família, no qual a duração do *setup* é definida de acordo com a família da qual cada atividade faz parte. As atividades são processadas em lotes, de forma que quando um lote sucede outro de mesma família, não é incorrido tempo de *setup*. O objetivo é minimizar o *total completion time* ($\sum C_j$) e

o problema é tratado com uma heurística própria baseada na regra WSPT (*Weighted Shortest Processing Time*) que visa diminuir a necessidade de que *setups* sejam executados, enquanto busca balancear os términos das atividades de acordo com seus pesos. Os resultados obtidos são comparados com um método construtivo da literatura e limites inferiores também da literatura, obtendo bons resultados quanto à qualidade da solução e tempos computacionais.

Gokhale e Mathirajan (2012) apresentam uma aplicação em uma etapa crítica da produção de marchas para veículos, na qual máquinas idênticas em paralelo foram instaladas a fim de diminuir o impacto causado pela operação gargalo. Cada atividade requer um tempo de *setup* (s_{ij}) que depende da sequência programada, além de possuírem um peso (w_j) e uma data de liberação (r_j) previamente estipulados. O problema lida com elegibilidade das máquinas (M_j) onde nem todas podem realizar todas as atividades e tem como objetivo a minimização do *total weighted flowtime* ($\sum w_j F_j$). Foram desenvolvidas cinco heurísticas baseadas em listas ordenadas de acordo com as características inerentes às atividades e um modelo matemático. Para instâncias de porte menor os valores das heurísticas foram avaliados com relação a solução ótima encontrada, enquanto nos problemas de maior porte foram utilizadas estimativas de valores ótimos.

Labbi *et al*. (2014) desenvolveram duas heurísticas baseadas em listas de prioridade para tratar os problemas com restrição de recursos na fase de preparação. A fase de preparação ou *setup* ocorre antes da execução da atividade e necessita a utilização de recursos que são disponibilizados de forma limitada. Os autores realizaram testes com mais de 20.000 instâncias geradas por eles mesmos, com o objetivo de minimizar o *makespam* (C_{max}).

Wang e Cheng (2015) tratam um problema real de uma fábrica que precisa atender sua demanda subcontratando uma empresa similar quando necessário, onde cada fábrica (própria e contratada) se equivale a uma máquina. A restrição de disponibilidade é imposta no caso da subcontratada, dado o tempo limitado que a empresa disponibiliza para atender a terceiros. O modelo foi tratado com um método heurístico desenvolvido pelos autores a partir de heurísticas existentes utilizadas em problemas de máquinas paralelas sem restrição de disponibilidade. Os resultados obtidos foram superiores quando comparados aos resultados dos métodos originais na qual a heurística se baseou. O objetivo é reduzir o total de pedidos atrasados ($\sum U_j$), alocando os pedidos em cada uma das fábricas ao longo do horizonte de programação.

Rajkanth *et al*. (2016) aplicam uma heurística própria em um problema real de uma fábrica de amortecedores automotivos, tendo como objetivo a minimização da variância do *total completion time* (VTCT). O estudo aborda a operação gargalo da fábrica que conta com duas máquinas idênticas em paralelo e considera que cada pedido de um produto específico com a sua quantidade demandada é uma atividade a ser programada. A heurística se mostrou aplicável, com resultados satisfatórios quando comparados com outras usadas na literatura.

Sabouni *et al*. (2010) aplicam um algoritmo genético para resolver um problema bi-critério, visando minimizar o *total completion time* ($\sum C_j$) e o *maximum lateness* (L_{max}) de forma simultânea em um problema com *setup* para as atividades, que devem ser processadas em batelada ou lote, de forma que o término de cada atividade se dá apenas quando a batelada da qual a mesma faz parte é finalizada. Para gerar a solução inicial os autores utilizaram uma abordagem construtiva baseada no *ranking* de Pareto. Duas variantes de algoritmos genéticos foram comparadas utilizando-se instâncias desenvolvidas de forma aleatória.

Bathrinath *et al*. (2013) tratam um problema multicritério, no qual se deseja minimizar de forma conjunta o *makespam* (C_{max}) e o *number of tardy Jobs* ($\sum U_j$). Os autores criaram as instâncias de forma aleatória e trataram o problema com a metaheurística *Simulated Anneling* (SA) e um algoritmo genético, com vantagem quanto às soluções obtidas pelo algoritmo genético, porém demandando maior tempo computacional.

Bathrinath *et al*. (2015) abordam o mesmo problema, dessa vez com o intuito de comparar as metaheurísticas *Variable Neighboohood search* (VNS) e *Simulated Anneling* (SA), apontando vantagem na qualidade de solução para a metaheurística VNS.

Kaid *et al*. (2015) propuseram a aplicação das metaheurísticas Busca Tabu e *Simulated Anneling* (SA), para o problema de minimização do *mean tardiness* ($\sum T_j \,/n$). Os autores utilizaram a heurística construtiva EDD (*Earliest Due Date*) para gerar uma solução inicial para as metaheurísticas e também como base de comparação para a qualidade das soluções obtidas. A Busca Tabu obteve vantagem quanto a qualidade geral das soluções, porém com tempo computacional muito superior ao *Simulated Anneling* (SA).

Lalla-Ruiz e Voß (2015) utilizam uma metaheurística VNS (*Variable Neighbohood Search*) aliada a um fator de memória para tratar o problema de

minimização do *Total Tardiness* ($\sum T_j$). O fator de memória é aplicado nas soluções iniciais que são geradas de forma aleatória, onde a cada iteração as soluções anteriores são utilizadas como base probabilística para o sorteio das novas, garantindo boa qualidade nas soluções de entrada da metaheurística. A aplicação se deu em instâncias da literatura com soluções ótimas conhecidas e que também foram alcançadas pelo método proposto. Dessa forma a validação do método se deu com base nos tempos computacionais, a fim de comparar a performance da metaheurística proposta com relação a outras metaheurísticas e métodos exatos testados.

3.2.3 Reprogramação de máquinas paralelas idênticas

Em muitos problemas de aplicações reais, a execução das programações propostas não ocorre de acordo com o planejado devido a perturbações no processo. Essas perturbações atingem tanto as atividades como as máquinas programadas e trazem consigo a necessidade de que o cronograma seja refeito para se adequar à nova realidade. Alguns trabalhos abordam esse tipo de problema em máquinas paralelas idênticas.

Azizoglu e Alagöz (2005) abordam o problema de quebra de máquinas, no qual uma das máquinas fica temporariamente indisponível, demandando a reprogramação das atividades da máquina indisponível nas demais. Os autores propõem um algoritmo com base na regra SPT (*Shortest Processing Time*) para uma função bi-critério que visa minimizar o *total completion time* ($\sum C_j$), juntamente com o número de atividades que mudam de máquina na comparação entre os cronogramas. Os testes são realizados tanto para o problema com hierarquia entre os objetivos quanto para o tratamento de forma simultânea.

Liu e Zhou (2013) tratam o problema do retrabalho de atividades, no qual atividades já executadas retornam para o sistema, causando uma perturbação na programação prevista, previamente otimizada com base no *total completion time* ($\sum C_j$). O objetivo tratado utiliza dois critérios, visando minimizar *o total completion time* ($\sum C_j$) do novo cronograma e também o número de atividades que mudam de máquina do planejamento original para o modificado. O trabalho desenvolve, por meio de heurísticas próprias, os cronogramas para variações hierárquicas dentre os elementos da função objetivo, bem como para o objetivo bi-critério simultâneo dos elementos a serem minimizados.

Em Hamzadayi e Yildiz (2016) é estudado um problema de reprogramação *online*, no qual a programação é refeita por completo quando novas atividades entram no sistema,

máquinas quebram ou voltam a funcionar ou quando a carga de trabalho está abaixo da mínima exigida no chão de fábrica. São considerados tempos de *setup* dependentes da sequência e servidor único, fazendo com que apenas um *setup* possa ser executado por vez. Os autores utilizam duas metaheurísticas para reconstruir o cronograma a cada vez que um dos eventos citados ocorre em um modelo de simulação, a fim de validar como os métodos funcionariam na prática, visando minimizar o *makespam* (C_{max}).

Yin *et al*. (2016) visam minimizar o impacto causado pela quebra de uma máquina em uma programação previamente otimizada com foco na minimização do *total completion time* ($\sum C_j$). O objetivo da nova programação passa a ser bi-critério levando em consideração não só o *total completion time* ($\sum C_j$) do novo planejamento, mas também a discrepância do cronograma atual para com o original, medido pela diferença do novo término de cada atividade com relação ao término planejado anteriormente. Os autores propõem duas heurísticas para a resolução do problema, comparando a qualidade da solução dos dois métodos para diferentes instâncias aleatórias propostas.

Em Vieira *et al*. (2003) é apresentado um framework sobre as estratégias e métodos na área de reprogramação de máquinas na manufatura, além de uma revisão sobre estudos e definições acerca do tema.

3.3 *Iterated Local Search* em programação de máquinas

A partir dos trabalhos exibidos na Seção 3.2, ficam evidentes os inúmeros métodos de resolução que podem ser aplicados aos problemas de programação de máquinas. Contudo, a fim de aprofundar o conhecimento relacionado a aplicações com *Iterated Local Search* (ILS), método proposto para o problema de reprogramação dos PLSVs, alguns trabalhos aplicando o ILS em diferentes problemas de programação de máquinas são expostos a seguir.

Santos *et al.* (2016) abordam um problema de máquinas paralelas não-relacionadas com fator de deterioração, no qual, a performance das máquinas piora conforme as atividades são executadas. Os experimentos foram realizados em instâncias da literatura como objetivo de minimizar o *makespam*. Um VND com duas vizinhanças foi utilizado na busca local e os resultados foram comparados com uma metaheurística *simulated annealing*, com melhores resultados médios obtidos para o ILS.

Fanjul-Peyro e Ruiz (2010) propõem um *iterated greedy local search*, método similar ao ILS, para o mesmo problema. Os autores aplicam uma perturbação viciada, de forma que, é maior a probabilidade de que seja selecionada a máquina com o maior tempo

de término e as atividades com menor tempo de execução em máquinas diferentes das quais encontram-se alocadas. São propostos sete diferentes métodos com variações nas perturbações e na construção da solução inicial, com um deles se destacando com melhores resultados na comparação com o método exato, em até 240 segundos de execução.

Dong *et al.* (2009) aplicam o ILS em um *permutation flowshop* com o objetivo de minimizar o *total completion time* ($\sum C_j$). Para a perturbação é utilizada o *swap* de atividades adjacentes e para a busca local a vizinhança *insert*, tendo o método encontrado melhores resultados em mais da metade das instâncias testadas, no comparativo com outros sete métodos.

Ruiz e Stützle (2007) tratam o mesmo problema com um *iterated greedy algorithm*, também análogo ao ILS. Os autores usam um método que destrói as soluções, removendo elementos aleatóriamente, e as reconstrói a partir de inserções de atividades nas melhores posições possíveis a fim de minimizar o *makespam*. Duas abordagens são utilizadas, uma delas com a aplicação de uma busca local e a outra sem, usando um *simulated annealing* como critério de aceitação em ambas, sendo a abordagem com busca local a que obteve os melhores resultados em uma comparação com outros 12 métodos.

Juan *et al.* (2014) também abordam um *flowshop* com o objetivo de minimizar o *makespam*. Na construção da solução inicial o algoritmo é paralelizado, desenvolvendo simultaneamente várias soluções aleatórias. O método ficou entre os melhores no comparativo com outros cinco.

Cruz *et al.* (2013) abordam um *job shop* a fm de minimizar o *weighted tardiness* ($\sum w_j T_j$). Uma metaheurística GRASP é utilizada para a construção da solução inicial e um VND com quatro vizinhanças é proposto como busca local do ILS. Os resultados, para instâncias da literatura, foram comparados com quatro métodos distintos obtendo resultados médios similares aos melhores.

Xu *et al.* (2014) tratam de um problema de máquina única com *setup* dependente da sequência das atividades. O objetivo é minimizar o *weighted tardiness* ($\sum w_j T_j$) da programação e os autores propõem uma nova vizinhança intitulada *block move*, na qual, blocos de atividades são sorteados e alocados em uma posição diferente da programação, tanto na perturbação quanto na busca local. No comparativo com os melhores resultados para instâncias da literatura, o método foi superior em aproximadamente 28% dos testes.

Qin *et al.* (2015) também aborda um problema com máquina única, porém, com o objetivo de minimizar uma função bi-critério do *earliness* e do *tardiness* da programação. Três pertubações são propostas, uma utilizando busca tabu, outra GRASP e uma terceira aleatória. Uma busca local com duas vizinhanças nas quais as trocas são limitadas a uma distância máxima entre as atividades é proposta. Os resultados foram comparados com outros quatro métodos, com resultados equivalentes e até melhores em determinadas instâncias.

No Quadro 1 são listados os trabalhos apresentados nas seções 3.2 e 3.3. As principais informações foram consolidadas, indicando os trabalhos que são de programação e de reprogramação, identificando se o trabalho trata de um caso de aplicação a problemas reais ou um estudo teórico, as características particulares ao problema tratado, qual o objetivo a ser minimizado e, por fim, os métodos de resolução utilizados propostos pelos autores.

Quadro 1 - Síntese de trabalhos em programação de máquinas. Fonte: Elaboração própria.

Referência	Tipo	Aplicação	Características	Objetivo	Método
Biskup *et al*. (2008)	Programação	Teórica	-	*Total tardiness*	Programação Inteira e heurísticas
Sabouni *et al*. (2010)	Programação	Teórica	*Setup* e processamento em lotes	*Total completion time* e *maximum lateness* (Lmáx)	Algoritmo genético
Mellouli *et al*. (2009)	Programação	Teórica	Indisponibilidade de máquina	*Total completion time*	Métodos exatos e heurísticas
Su (2009)	Programação	Teórica	Elegibilidade de máquina	*Total completion time*	Branch and bound
Xu *et al*. (2009)	Programação	Teórica	Indisponibilidade de máquina	*Makespam*	Heurística
Zhao *et al*. (2011)	Programação	Teórica	Indisponibilidade de máquina	*Total weighted completion time*	Heurística
Liao *et al*. (2012)	Programação	Teórica	*Setup* por família	*Total completion time*	Heurística
Gokhale e Mathirajan (2012)	Programação	Prática	*Setup* dependente da sequência, prioridade das atividades e elegibilidade de máquina	*Total weighted flowtime*	Programação Inteira e heurísticas
Bathrinath *et al*. (2013)	Programação	Teórica	-	*Makespam* e *number of tardy Jobs*	*Simulated Annealing*
Bathrinath *et al*. (2015)	Programação	Teórica	-	*Makespam* e *number of tardy Jobs*	VNS e *Simulated Annealing*

Labbi *et al*. (2014)	Programação	Teórica	*Setup* e restrição de recursos	*Makespam*	Heurística
Wang e Cheng (2015)	Programação	Prática	Indisponibilidade de máquina	*Number of tardy Jobs*	Heurística
Kaid *et al*. (2015)	Programação	Teórica	-	*Mean tardiness*	Busca Tabu e *Simulated Annealing*
Lalla-Ruiz e Voß (2015)	Programação	Teórica	-	*Total tardiness*	VNS
Rajkanth *et al*. (2016)	Programação	Prática	-	Variância do *total completion time*	Heurística
Azizoglu e Alagöz (2005)	Reprogramação	Teórica	Indisponibilidade de máquina	*Total completion time* e número de atividades que mudam de máquina	Heurística
Liu e Zhou (2013)	Reprogramação	Teórica	Reprocessamento de atividades	*Total completion time* e número de atividades que mudam de máquina	Heurística
Hamzadayi e Yildiz (2016)	Reprogramação	Teórica	Online com indisponibilidade de máquina	*Makespam*	*Simulated Annealing*
Yin *et al*. (2016)	Reprogramação	Teórica	Indisponibilidade de máquina	*Total completion time* e discrepância do cronograma atual para com o original	Heurística
Santos *et al.* (2016)	Programação	Teórica	Deterioração de máquina	*Makespam*	ILS e RVND
Fanjul-Peyro e Ruiz (2010)	Programação	Teórica	-	*Makespam*	*Iterated greedy local search*
Dong *et al.* (2009)	Programação	Teórica	-	*Total completion time*	ILS

Ruiz e Stützle (2007)	Programação	Teórica	-	*Makespam*	*Iterated greedy algorithm*
Juan *et al.* (2014)	Programação	Teórica	-	*Makespam*	GRASP e ILS
Cruz *et al.* (2013)	Programação	Teórica	-	*Total weighted tardiness*	GRASP, VND e ILS
Xu *et al.* (2014)	Programação	Teórica	*Setup* dependente da sequência	*Total weighted tardiness*	ILS
Qin *et al.* (2015)	Programação	Teórica	-	*Earliness e tardiness*	ILS

3.4
Considerações sobre a revisão da literatura

O tratamento de problemas que envolvem a cadeia logística de apoio às operações *offshore*, como é o caso da programação de PLSVs, se destaca por lidar com especificidades e particularidades de difícil tratamento, contribuindo para a expansão da literatura em aplicações a problemas práticos.

O problema de reprogramação de PLSVs, como abordado nesse trabalho, traz uma nova metodologia de aplicação que avança no conceito de resolução desse tipo de problema por meio de modelos de máquinas paralelas. Trabalhos anteriores encontrados sobre o tema e que propuseram tal abordagem expandiram o horizonte na forma de tratar o problema em questão, contudo, fazendo uso de algumas hipóteses simplificadoras, a fim de aproximá-lo de uma abordagem mais clássica da literatura de máquinas paralelas. Estas simplificações distanciam a aplicação do problema real. A principal simplificação se dá no conceito de viagem, onde parte-se do pressuposto de que estas são construídas em uma etapa prévia à programação, fazendo com que cada viagem passe a ser considerada como uma única atividade a ser programada, não passível de alterações. Outro aspecto a destacar é que cada viagem atende a um único poço específico, sendo o término do poço definido a partir do término da viagem. As premissas seguidas no presente trabalho, permitindo viagens com atividades de diferentes poços, alocando atividades de diferentes poços em diferentes viagens e navios, permitindo alterações na composição das viagens durante a reprogramação e utilizando a data de compromisso por poço, consideram o problema vivenciado no dia a dia dos programadores.

A utilização da metodologia baseada em um problema de programação de máquinas incrementa o conteúdo em torno desse assunto que, apesar de diversificado e amplamente encontrado na literatura, possui uma parcela menor de trabalhos com aplicações reais, com grande parte das publicações com enfoque acadêmico. Além de ampliar ainda mais as publicações em planejamento de frotas marítimas com esse tipo de abordagem, ainda incipiente na literatura e em trabalhos de reprogramação.

No melhor do nosso conhecimento, alguns aspectos não foram encontrados da forma como abordado neste trabalho, que são: a forma como o tempo de *setup* é contabilizado, as datas de compromisso por poço e o sequenciamento de atividades

dentro de um agrupamento. Em geral, os tempos de *setup* são dependentes da sequência entre as atividades ou da máquina específica que realiza a atividade e não baseada no agrupamento de atividades, tendo uma regra específica que calcula a duração do *setup* pelo tipo de atividade e pela combinação de atividades de diferentes tipos realizadas em sequência. Quanto às datas de compromisso por poço, observa-se o término da última atividade relacionada a um poço para calcular sua data de término quando, em geral, cada atividade possui uma data específica de compromisso. Por último, a consideração do sequenciamento de atividades nas viagens se diferencia de uma abordagem em lotes, na qual a decisão de agrupar é feita de forma desvinculada da programação, não existindo a preocupação com o ordenamento dentro do lote.

Essas características tornam o problema mais desafiador, além de evidenciarem a importância do presente trabalho para a literatura, seja no âmbito de máquinas paralelas idênticas, quanto no âmbito de programação de frotas marítimas com ênfase em apoio a operações *offshore*.

Com base na notação utilizada em problemas de programação em máquinas, o problema aqui abordado se refere a um caso de reprogramação em máquinas paralelas idênticas (P_m), onde os navios assumem o papel das máquinas a serem programadas. Dado o conjunto M $(1 \leq i \leq m)$ de navios, o conjunto J $(1 \leq j \leq n)$ de atividades a eles alocadas e o conjunto P $(1 \leq k \leq p)$ de poços, os navios precisam ser reprogramados em um horizonte de tempo H $(t = 1 \ldots H)$, seguindo as especificações do problema detalhadas a seguir. Cada atividade possui um tempo de processamento p_j que independe do navio que a executa, uma data de liberação r_j e um conjunto elegível de embarcações M_j capazes de realizá-la. Além disso, as atividades possuem uma taxa ocupação $o_j \in [0, 1]$ e um tipo q_j, de forma que, um subconjunto de atividades só pode ser realizado em sequência se a soma de suas ocupações não ultrapassarem o limite máximo permitido. Esses subconjuntos são denominados como viagens $v \in$ V onde, para cada viagem, um tempo de *setup* s_v deve ser executado antes de sua realização, calculado com base nos tipos de atividades que nela constam. Cada tipo demanda uma duração de *setup* específica, sendo esta a duração estipulada para viagens com apenas um tipo. Para viagens que combinam diversos tipos de atividade, uma outra duração de *setup* para essa combinação é estipulada, sem que necessariamente corresponda a soma das

durações dos *setups* dos tipos agrupados. Para a programação dos PLSVs, a regra de calculo de *setup* atualmente utilizada pela companhia encontra-se na Seção 2.1.1 deste trabalho. Após a execução de cada subconjunto v um tempo de retorno r_v deve ser considerado. Os navios possuem disponibilidade com janela inferior (B_i) e superior (F_i) conhecidas e cada poço k possui uma data de compromisso d_k como instante de tempo alvo para sua conclusão e um conjunto de atividades J_k a ele associado. O critério de avaliação a ser minimizado não está entre os critérios regulares, ou seja, não se encontra entre aquelas mais comumente abordadas na literatura de programação de máquinas, tendo como objetivo a minimização de três componentes que medem o impacto operacional acarretado no que foi programado: atraso dos poços, excedente utilizado em cada navio e ociosidade do cronograma. Esta função é especificada com maiores detalhes na Seção 5.1 deste trabalho.

No Quadro 2 são sintetizadas as notações definidas para o problema de programação dos PLSVs.

Quadro 2 - Notações para o problema de programação dos PLSVs. Fonte: Elaboração própria.

n	Número de atividades
J	Conjunto de atividades
m	Número de máquinas
M	Conjunto de máquinas
p	Número de poços
P	Conjunto de poços
j	Indice das atividades, $j = 1 \dots n$
i	Indice das máquinas, $i = 1 \dots m$
k	Indice dos poços, $k = 1 \dots p$
H	Horizonte de planejamento
t	Indice dos instantes de tempo (períodos), $t = 1 \dots H$
V	Conjunto de viagens
v	Indice das viagens, $v = 1 \dots \lvert V \rvert$
p_j	Tempo de processamento da atividade j
r_j	Data de liberação da atividade j
M_j	Elegibilidade dos navios
J_k	Conjunto de atividades associadas a um poço k
d_k	Data de compromisso dos poços k
B_i	Janela inferior da embarcação i
F_i	Janela superior da embarcação i
s_v	*Setup* da viagem v
r_v	Retorno da viagem v

4
Metodologia

Este capítulo apresenta a classificação da pesquisa desenvolvida, detalha as etapas seguidas e as limitações encontradas. Após a primeira parte do capítulo, que aborda a classificação da pesquisa quanto aos fins e meios, são discriminados os passos utilizados para o desenvolvimento do trabalho proposto, detalhando cada um deles, evidenciando a sua contribuição para a evolução do estudo. O capítulo é finalizado com uma descrição sobre as limitações descobertas durante o desenvolvimento das etapas propostas no presente estudo.

4.1
Tipo de pesquisa

A taxonomia apresentada por Vergara (2007) será utilizada para classificar o tipo de pesquisa realizada, quantos aos fins e quantos aos meios de investigação.

Quanto aos fins, essa pesquisa se enquadra nos tipos: aplicada e exploratória. Aplicada, por tratar de um problema real de um processo diário de uma empresa de exploração e produção de óleo e gás, utilizando os dados por ela fornecidos, na resolução de um problema enfrentado diariamente pela companhia, com impacto direto na sua produtividade. Exploratória, por expandir o conhecimento no tipo de problema abordado, ampliando o campo de pesquisas de programação de navios por meio de modelos de programação de máquinas.

Quantos aos meios de investigação, a pesquisa é dos tipos: bibliográfica, experimental e estudo de caso. Bibliográfica, pois utiliza referencial teórico acerca do tema, oriundo de revistas, livros e meios eletrônicos; experimental, uma vez que trata de uma abordagem empírica, na qual a utilização de dados reais e testes estruturados visam o desenvolvimento de soluções de qualidade para o problema; e estudo de caso, por abordar o problema específico de uma única empresa.

A pesquisa tem um enfoque quantitativo, utilizando uma metaheurística ILS na proposição de soluções que sejam viáveis para o problema abordado e mais eficientes do que aquelas desenvolvidas atualmente de forma manual.

A aplicação do estudo em uma empresa específica, não inviabiliza a sua utilização em outros casos similares. Apesar das especificidades abordadas, seja pela forma de trabalhar da empresa, a característica das embarcações, os objetivos traçados ou outros fatores, a pesquisa contribui tanto no âmbito de programação de frotas marítimas com enfoque em serviços, como para a literatura de problemas aplicados com base em modelos de máquinas paralelas, indicando pontos importantes a serem seguidos no desenvolvimento desse tipo de estudo.

4.2 Etapas da pesquisa

O estudo foi estruturado a partir de etapas adaptadas da proposição feita por Gil (2002) para o desenvolvimento de estudos de caso. O seguimento estruturado propicia um alto grau de maturidade quanto ao entendimento do problema, auxiliando na decisão dos métodos a utilizar no seu tratamento. Dessa forma, foram estabelecidos seis passos a serem seguidos, que são:

- Formulação do problema
- Detalhamento do processo
- Coleta de dados
- Análise e tratamento dos dados
- Aplicação do método proposto
- Relatório de conclusão

A formulação do problema foi dividida em duas etapas, a primeira delas com relação à conceituação do caso estudado, feita através de reuniões realizadas com a gerência responsável pela exploração e produção de óleo e gás na companhia analisada, para que o problema fosse exposto e o objetivo geral definido. Essas reuniões consolidaram uma visão macro sobre o problema a ser tratado, delineando os interesses internos com relação ao estudo. Foram expostos o papel do navio PLSV na cadeia produtiva do petróleo, seu impacto nos indicadores da empresa e o interesse gerencial pela utilização de ferramentas computacionais no auxílio da gestão da programação da frota dessas embarcações. A segunda etapa consistiu em uma busca por referenciais bibliográficos relacionados ao assunto, a fim de que se localizasse o estudo na literatura e que fossem identificadas as principais técnicas utilizadas no tratamento de problemas similares.

O detalhamento do processo foi realizado em parceria com a área responsável pela programação das embarcações PLSV. Foram realizadas reuniões

tanto na própria área com os programadores, para conhecimento da atividade de desenvolvimento e gestão da programação da frota, como reuniões gerais com as diversas áreas impactadas pelo processo em questão. As reuniões envolvendo as demais áreas ocorrem semanalmente na empresa, com a participação daquelas diretamente envolvidas com a operação das embarcações em conjunto com os programadores. O intuito é alinhar as expectativas e necessidades de cada área, considerando-se as divergências ocorridas entre o que foi planejado e o executado pelas embarcações durante as operações. A participação nessas reuniões teve um importante papel na consolidação do entendimento acerca do processo de forma mais minuciosa e com relação às necessidades específicas das áreas envolvidas, evidenciando as diretrizes a serem seguidas na etapa de aplicação de um método para melhoria do processo.

Após as reuniões de detalhamento, os dados necessários foram fornecidos pela empresa. Ao todo, foram disponibilizados três anos de programação, dois destes relativos a períodos executados e o restante de planejamentos futuros. Para os períodos executados foram fornecidos também os planejamentos originais, como base de comparação, a fim de que fossem identificadas as discrepâncias entre o planejamento e a execução das operações. As programações consistem em cronogramas de execução de atividades por parte dos navios PLSV, indicando as operações realizadas ou a realizar por cada um deles em cada dia do horizonte de planejamento e a qual poço cada atividade está associada.

A etapa de análise dos dados se deu de forma interativa com a área responsável pelo processo. Os dados foram analisados para que fosse possível a identificação de padrões seguidos pelos programadores e validação das regras coletadas nas reuniões de detalhamento do processo. As dúvidas e discrepâncias encontradas eram levadas à área responsável para validação das análises feitas. Nessa etapa foram identificadas regras que não estavam sendo seguidas pelos programadores, além de ter ampliado ainda mais a compreensão com relação ao problema, proporcionando um indicativo de possíveis pontos de melhora. Ao término das análises, evidenciou-se a necessidade de que fosse realizado um tratamento prévio dos dados, a ser detalhado na Seção 5.3.1 deste trabalho, para que então os cronogramas pudessem ser submetidos a ajustes.

Com o conhecimento adequado do processo e após o tratamento dos dados, foi utilizada uma metaheurística ILS (*Iterated Local Search*) para o ajuste

automatizado dos cronogramas de curto prazo, visando aplicar inteligência computacional no auxílio aos programadores, aumentando a eficiência no processo quando comparado à forma atualmente praticada.

No relatório de conclusão, foram consolidados os resultados alcançados com o trabalho, descritos no Capítulo 6 do presente trabalho.

4.3 Limitações da pesquisa

A utilização de uma empresa específica, apesar de não inviabilizar o estudo como base para outros similares, conforme destacado anteriormente neste capítulo, limita a generalização do problema abordado quanto a aplicação em qualquer indústria do ramo, dado que todas as diretrizes seguidas se basearam em objetivos específicos traçados pela companhia avaliada.

Além disso, as diretrizes utilizadas correspondem a visão atual da companhia e cumprem objetivos e regras estipuladas pelas áreas envolvidas no processo, incluindo a própria área de programação e o entendimento dos programadores quanto à operação, durante a aplicação do estudo até o seu término, sendo a maior parte desses fatores passíveis de mudança ao longo do tempo.

Atualmente, os programadores da companhia estudada desconsideram, durante as etapas de programação e reprogramação das embarcações, a limitação do número de navios que podem efetuar simultaneamente o carregamento de materiais na base. Essa limitação se dá pelo número de berços disponíveis para atracação, de forma que, cada navio ocupa um berço durante o carregamento. Dessa forma, nesse estudo essa restrição também foi desconsiderada, porém, pode se caracterizar como uma limitação, dado o impacto que a programação de carregamentos simultâneos pode acarretar na operação.

O fato de usar dados não probabilísticos diminui a acurácia dos cronogramas desenvolvidos, dadas as inúmeras incertezas inerentes ao processo. A necessidade de intervenção por parte dos programadores tende a ser ainda maior por este motivo, podendo assim ser caracterizada como uma limitação do estudo.

Outra limitação a se destacar corresponde ao método de tratamento utilizado, nesse caso a metaheurística ILS, proposto com base nas características do problema, entendendo-se que este seria capaz de prover boas soluções. Todavia, sem a comparação com outros métodos, não é possível determinar sua eficácia para com outras metaheurísticas e se de fato este seria a melhor opção para o caso.

5
Métodos de resolução

Conforme abordado no Capítulo 2, as variações inerentes ao processo de interligação de poços submarinos em conjunto com as inconsistências geradas pelo trabalho manual de programação das embarcações resultam em cronogramas em desacordo com as proposições gerenciais da companhia, de forma que, a utilização de um método computacional de ajuste surge como uma potencial forma de reprogramar de maneira mais eficaz a execução das atividades realizadas pelos PLSVs. Desta forma, esse capítulo visa expor o método proposto para a reprogramação dos cronogramas dessas embarcações, feita com base na metaheurística ILS. Serão apresentadas a busca local aplicada e vizinhanças escolhidas, bem como as regras utilizadas em cada uma das etapas do método.

5.1
Função objetivo

O critério de avaliação ou função objetivo (FO) proposto para este trabalho, como destacado anteriormente, visa minimizar os impactos gerados no cronograma, estipulando prioridades entre os critérios avaliados para que se determine a qualidade de uma solução. Três componentes são considerados: atrasos dos poços considerados críticos, a extrapolação das janelas superiores (F_i) dos navios e a ociosidade da programação. Estes componentes definem uma função multicritério hierarquizada a ser minimizada.

O atraso relacionado aos poços críticos diz respeito ao número de dias além da sua data de compromisso (d_k), em que um poço k crítico é concluído (C_k). Esse instante de conclusão se refere ao término da última atividade associada ao poço e cada dia extra ($\max\{0, C_k - d_k\}$) é considerado como um dia a mais de atraso, contabilizado no montante a ser minimizado. Um poço é considerado crítico se o seu término acontece dentro do horizonte cortado, ou seja, se a última atividade referente a este poço está programada no trecho de cronograma avaliado e se, além disso, esse poço for do tipo produtor. As atividades que não estão associadas a poços críticos podem ser movimentadas livremente em um cronograma, sem que causem

impactos nesse critério. A forma como a implementação foi feita permite que de forma simples, outros critérios possam ser configurados, alterando-se a definição de quais poços são considerados críticos. Essa flexibilidade é importante por se tratar de um problema que lida diretamente com a operação.

Dentre os três critérios propostos, o atraso de poços possui a maior importância relativa para com os demais. Um exemplo de verificação de atraso em poço crítico é ilustrado na Figura 6, com dois poços críticos, A e B, programados, com o poço A apresentando sete dias de atraso com relação a sua data limite d_A para término.

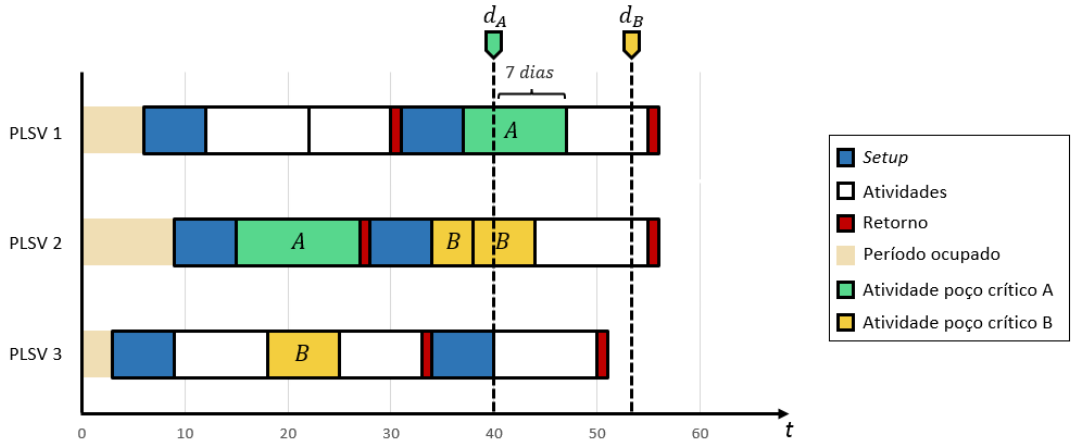


Figura 6 - Cálculo do componente atraso poço. Fonte: Elaboração própria.

Dentro do horizonte reprogramável, cada navio possui uma janela de disponibilidade com seus limites inferior (B_i) e superior (F_i). Contudo, a janela superior pode ser extrapolada mediante penalização. O período excedente é contabilizado a partir do término da última viagem programada para o navio (C_i) por meio da fórmula $\max\{0, C_i - F_i\}$, sendo este o segundo critério a ser minimizado na ordem hierárquica dos objetivos propostos. Esse critério é abordado como "excedente de navio", e um exemplo de como o mesmo é contabilizado é mostrado na Figura 7, com oito dias excedentes computados.

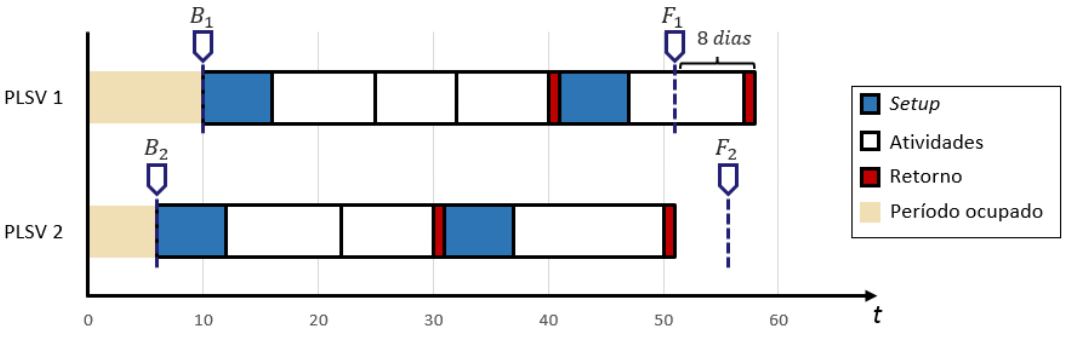


Figura 7 - Cálculo do componente excedente de navio. Fonte: Elaboração própria.

O último critério a ser minimizado diz respeito à ociosidade das embarcações na programação, calculada como o somatório dos períodos entre viagens, sem atividades associadas às embarcações. Dessa forma, são contabilizados todos os instantes de tempo em que cada embarcação ficou sem atividade a realizar, descontando o período final, correspondido entre o término da última atividade realizada pelo navio e sua janela superior de disponibilidade (F_i). Esse critério tem menor importância relativa entre os três a serem minimizados, mas deve ser tratado devido aos impactos nos indicadores internos da companhia.

Para que se respeitem os três critérios, mantendo-se a hierarquia entre eles, a função objetivo utiliza fatores ($\alpha, \beta\ e\ \gamma$) associados a cada um dos componentes, de forma que, $\alpha \in [0,1]$, $\beta \in [0,1]$, $\gamma \in [0,1]$ e $\alpha + \beta + \gamma = 1$. Dessa forma, considerando-se os critérios de atraso de poço (AP), excedente de navio (EN) e ociosidade (OC), o conjunto de navios M $(1 \leq i \leq m)$ e uma programação π qualquer, a função de avaliação proposta para o presente trabalho é definida como:

$$Minimizar: f(\pi) = \sum_{i=1}^{m} (\alpha\ AP_i + \beta\ EN_i + \gamma\ OC_i)\,, \forall i\ \in M$$

O componente que mede o atraso poço é associada ao navio para que se identifique qual embarcação está causando os maiores impactos na função objetivo. O atraso é contabilizado no navio que está programado para realizar a última atividade associada a um poço crítico, estando esta atividade atrasada em relação a data de compromisso do poço em questão. Dessa forma, os três componentes da função objetivo proposta são vinculados aos navios.

5.2 Famílias de PLSV

A elegibilidade dos navios conforme abordada no problema de reprogramação dos PLSVs se caracteriza pela condição imposta pela companhia de que as trocas de atividades só devem ocorrer entre navios de mesma família. A família, neste caso, corresponde à empresa detentora do navio ao qual a atividade encontra-se programada, conforme destacado na Seção 2.1.2. Essa característica traz consigo uma vantagem pelo fato de que se definem conjuntos restritos de máquinas capazes de realizar um grupo de atividades, sem interseção entre eles. Isso indica que não existe disputa de recursos por atividades atendidas em diferentes grupos de PLSVs, caracterizando o problema como um conjunto de subproblemas de programação a serem tratados.

Desta forma, um conjunto de famílias L ($l = 1 \ldots |L|$) pode ser definido, em que cada família possuí um subconjunto de navios M_l e um subconjunto de atividades N_l, ambos exclusivos, contendo os navios que fazem parte da família e as respectivas atividades a eles alocadas e que consequentemente não podem mudar de família. Além disso, a própria avaliação da solução através da função proposta pode ser feita por família, sendo então uma programação π qualquer desmembrada em um subconjunto de programações π_l.

A utilização desse conceito durante as etapas do método proposto aumenta a frequência na qual bons resultados são encontrados, exigindo um menor tempo computacional para tal. Esses resultados foram evidenciados durante os testes iniciais com o algoritmo e a estratégia da resolução de subproblemas por família foi então empregada.

5.3 Metaheurística ILS

A metaheurística ILS (*Iterated Local Search*) é um método iterativo que combina etapas de perturbação da solução, configuradas como pequenas alterações nesta solução, seguidas da aplicação de buscas locais a fim de melhorar a solução perturbada. A partir de uma solução s^*, representada pela melhor solução encontrada até então com base no critério de avaliação proposto para o problema, uma perturbação feita gera uma nova solução s', para que esta passe pelo processo de busca local, resultando em uma solução $s^{*\prime}$. Caso $s^{*\prime}$ seja aceita mediante um critério previamente estipulado, esta passa a ser a solução s^*, caso contrário, s^* não é modificada. O processo se repete dessa forma até que um critério de finalização do método seja alcançado (Lourenço *et al.*, 2010). O pseudocódigo do algoritmo do ILS é apresentado na Figura 8.

Algoritmo ILS (*Iterated Local Search*)

1: $s_0 = GeraSoluçãoInicial$
2: $s^* = BuscaLocal\ (s_0)$
3: repita
  4: $s' = Perturbação\ (s^*)$
  5: $s^{*'} = BuscaLocal\ (s')$
  6: $s^* = CritériodeAceitação\ (s^*, s^{*'})$
7: até Critério de Parada

Figura 8 - Pseudocódigo ILS. Fonte: Adaptado de (Lourenço *et al.*, 2010).

O ILS tem sido aplicado em diversos trabalhos de programação, se mostrando uma metaheurística eficiente no tratamento de problemas correlatos. Na Seção 3.3 é exibido um levantamento de alguns trabalhos que utilizaram o ILS em problemas de programação de máquinas.

Cada etapa do ILS, desenvolvido neste estudo, será detalhada a seguir, desde a geração da solução inicial até o critério de parada do algoritmo.

5.3.1 Geração da Solução Inicial

Apesar do cronograma utilizado ser fornecido pela companhia, são necessárias algumas etapas de pré-processamento para que seja delimitado o horizonte que será trabalhado e se corrijam os problemas que tornam a programação inviável. O cronograma definido após os ajustes passa a ser a solução inicial do ILS, servindo de entrada para a etapa de busca local que ocorre em seguida.

5.3.1.1 Corte no Horizonte

A primeira etapa realizada no tratamento do cronograma recebido é a definição do intervalo de horizonte a ser reprogramado. De acordo com a definição do problema, esse intervalo corresponde aos três meses subsequentes à data observada. Porém, como forma de abordar o problema de forma mais abrangente, um procedimento de corte do cronograma foi elaborado. O procedimento se baseia em dois instantes de tempo (inicial e final) quaisquer, definidos dentro do horizonte de planejamento avaliado, delimitando o período que será reprogramado.

O corte tem como objetivo definir as viagens que poderão ser reprogramadas e quais as janelas disponíveis em cada uma das embarcações. Apesar de serem utilizados os mesmos instantes de referência para o corte, as janelas de disponibilidade podem divergir entre os navios. Essa discrepância pode se dar pela indisponibilidade do navio em parte do trecho definido, ou, devido ao respeito à integridade das viagens programadas. Quando uma viagem se encontra em execução no instante inicial definido para o corte, a mesma é desconsiderada por completo e não entra no horizonte reprogramável. Aquelas com execução programadas no instante final de corte são consideradas por completo e passíveis de reprogramação.

A partir das datas programadas para as viagens definidas como reprogramáveis, da disponibilidade original de cada embarcação e dos instantes

escolhidos para o corte, são definidas as janelas inferior (B_i) e superior (F_i) de cada embarcação. Para os casos em que um navio programado não se encontra disponível no intervalo definido para o corte, o mesmo passa a ser desconsiderado, não podendo ter atividades a ele alocadas na etapa de reprogramação.

Na Figura 9 é ilustrado o processo de corte do cronograma feito com base em dois instantes t_1 e t_2, destacando no cronograma base as viagens que resultarão do corte dentre todas as que foram programadas, e na sequência apresentando o cronograma final, indicando o período ocupado pelas viagens executadas ou em execução, as viagens a serem reprogramadas e a janela de disponibilidade de cada embarcação.

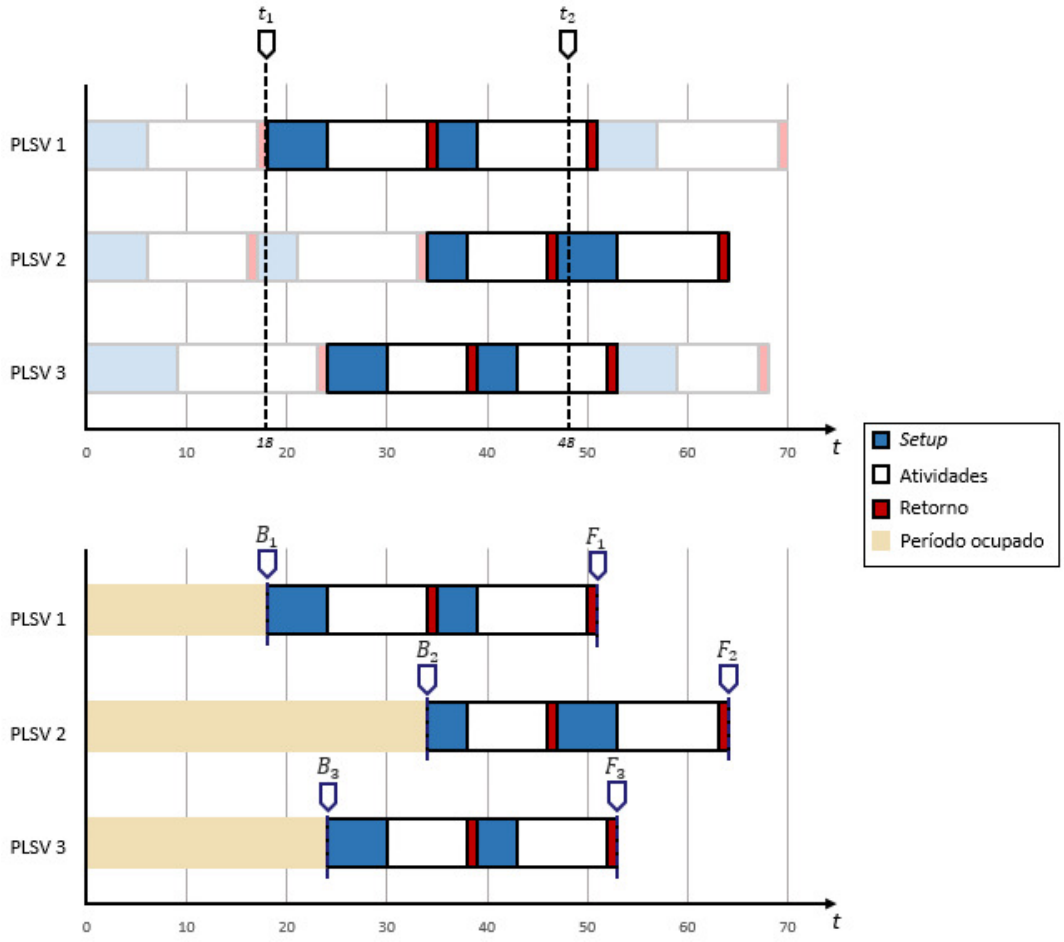


Figura 9 - Procedimento de corte do cronograma. Fonte: Elaboração própria.

5.3.1.2 Avaliação e ajuste do Cronograma

O impacto causado pelas incertezas que afetam as durações das atividades executadas pelos PLSVs e pelo processo de programação manual realizado na companhia estudada, conforme descrito no Capítulo 2, dificulta o cumprimento de todas as premissas do problema de programação de PLSVs, provocando

irregularidades evidenciadas durante a etapa de análise dos dados. Dessa forma, após o corte do cronograma ser finalizado, a solução de entrada precisa ser avaliada, a fim de que sejam identificadas e listadas essas irregularidades, para posterior correção. Aquelas que aparecem com maior frequência são:

- Tempos de *setup* com duração indevida;
- Descumprimento dos limites de capacidade de cada viagem;
- Não cumprimento das datas de liberação das atividades.

Após a identificação das irregularidades, o cronograma é tratado para que estas sejam eliminadas, tornando-o viável quanto às restrições impostas pelo problema.

Quando o tempo de *setup* programado para uma determinada viagem está aquém do seu tempo devido, o mesmo é recalculado e modificado para a sua duração correta, prorrogando o início de todas as atividades programadas após esse *setup*, conforme indica a Figura 10.

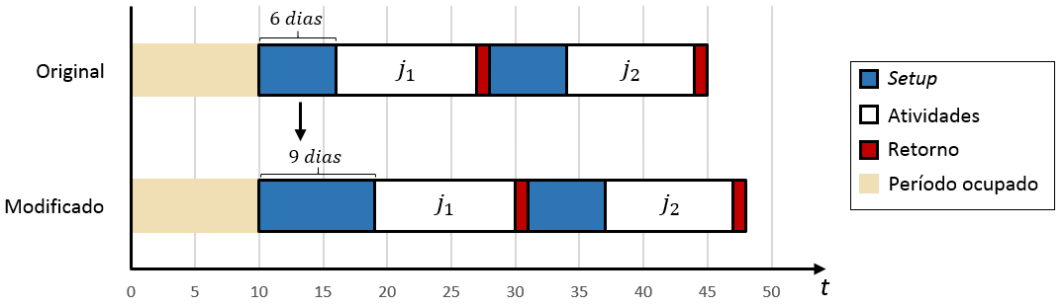


Figura 10 - Ajuste de *setup* com duração menor que a devida. Fonte: Elaboração própria.

Quando o oposto ocorre, ou seja, o tempo de *setup* é maior do que o correto, a sua duração é recalculada e o *setup* modificado, sem que se alterem os instantes de início das atividades posteriores, mantendo a alocação originalmente proposta. Na etapa de ajuste de datas de liberação, esta viagem e as demais posteriores a ela serão antecipadas com relação às datas de liberação, se possível. Um exemplo desse ajuste é mostrado na Figura 11.

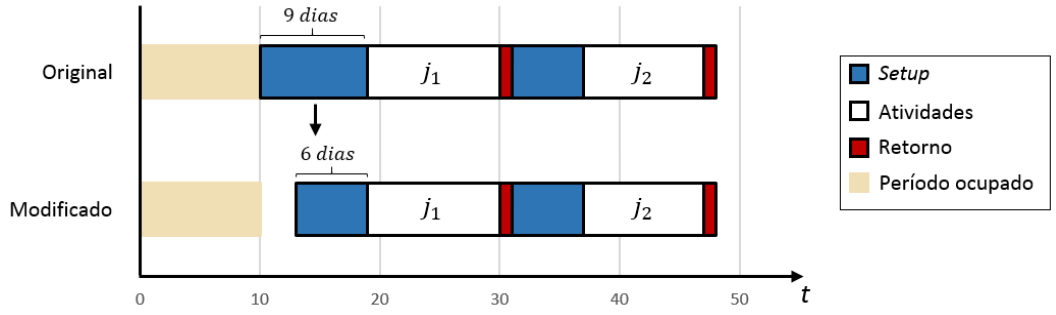


Figura 11 - Ajuste de *setup* com duração maior que a devida. Fonte: Elaboração própria.

Para o tratamento de viagens com capacidade indevida, aquelas que ultrapassem o limite permitido são desmembradas em quantas forem necessárias para viabilizar o cronograma. As viagens são desmembradas mantendo-se a alocação e sequenciamento das atividades propostas pelos programadores, de forma que, a capacidade é avaliada de forma cumulativa da primeira para a última atividade programada, sendo a viagem desmembrada no momento em que a entrada da próxima atividade ultrapasse o limite de ocupação permitido. A criação de novas viagens desloca todas as viagens posteriores ao trecho alterado, conforme indicado na Figura 12. Nesse exemplo, duas viagens foram desmembradas em três devido ao excedente de ocupação identificado em uma delas.

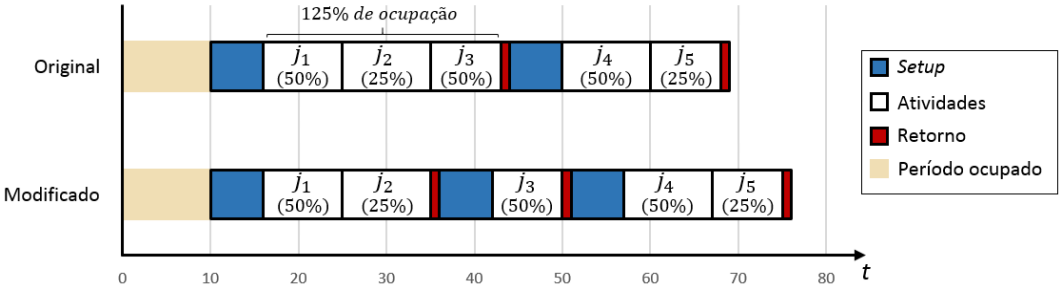


Figura 12 - Ajuste da capacidade das viagens. Fonte: Elaboração própria.

O último tratamento lida com as datas de liberação das atividades, ajustando o cronograma com relação ao cumprimento dessas datas. Como a alocação da viagem é determinada por essas datas, a maior delas dentre todas as atividades programadas em uma viagem, indica o instante mais cedo no qual o *setup* pode ser iniciado. O ajuste é feito por viagem, antecipando-as ou atrasando-as quando necessário a fim de viabilizar todas as datas de liberação das atividades programadas. Na Figura 13 uma programação com 2 viagens é ajustada. Nesse caso, a primeira viagem foi antecipada devido a maior data de liberação das atividades ($r_2 = 10$) que permite que a mesma inicie no instante $t = 10$. Já a segunda viagem foi atrasada para iniciar em $t = 50$, data de liberação da atividade 4 ($r_4 = 50$).

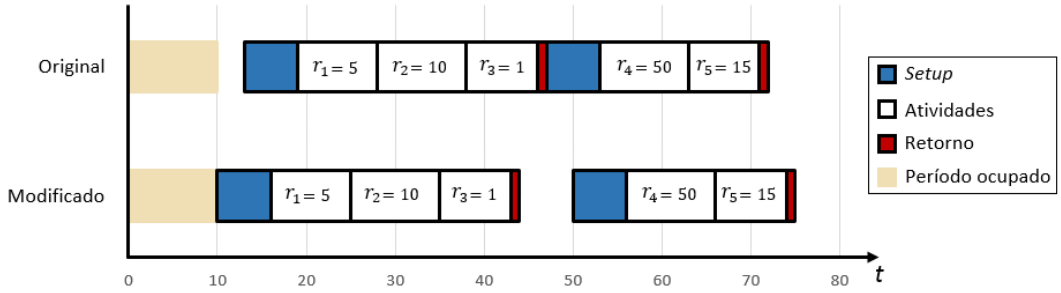


Figura 13 - Ajuste das viagens com base nas datas de liberação. Fonte: Elaboração própria.

Ao término dos ajustes no cronograma, o mesmo passa a ser a solução de referência para a reprogramação, cumprindo todas as premissas e restrições impostas, passando então a ser avaliado com base na função multicritério proposta.

5.3.2 Busca Local

Uma busca local (*local search*) consiste em um método iterativo baseado nas vizinhanças do conjunto de soluções de um problema, de modo que, partindo-se de uma solução inicial, esta é substituída por uma solução vizinha quando existe melhora no critério de avaliação proposto para o problema (Talbi, 2009).

Uma vizinhança N constitui um conjunto de soluções s' chamadas de vizinhas, geradas a partir de operações de movimento, caracterizadas como pequenas modificações realizadas em uma solução s pertencente ao conjunto S de soluções viáveis de um determinado problema. Define-se então que, para cada solução $s \in S$, tem-se um subconjunto de soluções $N(s) \subset S$ (Talbi, 2009). Para a busca local proposta para o problema dos PLSVs foi utilizada a vizinhança *insert* como base.

Na vizinhança *insert*, uma atividade qualquer é retirada de uma posição e inserida em outra (Dong *et al.*, 2009). A inserção pode ser feita tanto dentro de uma viagem existente como em uma posição entre viagens. A viagem escolhida pode ser a mesma onde a atividade encontra-se alocada ou uma viagem distinta, no mesmo PLSV de origem ou em um PLSV diferente. As inserções entre viagens também podem ser feitas tanto no PLSV de origem como em PLSVs distintos. Na Figura 14 é demonstrado o *Insert* realizado dentro de uma viagem em um único PLSV.

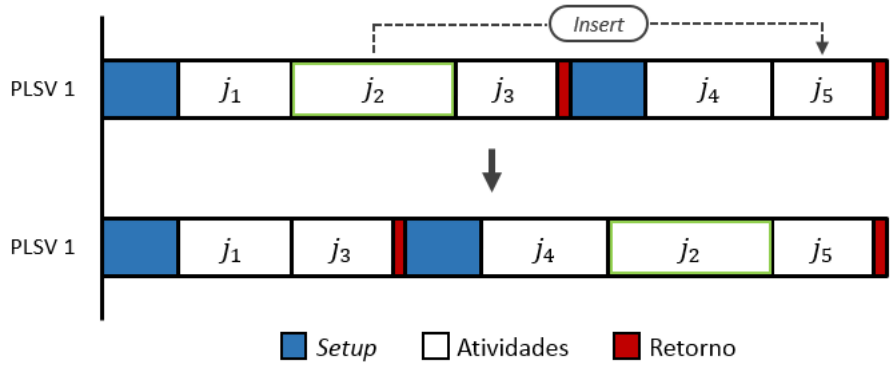


Figura 14 - *Insert* de atividade em uma viagem em um mesmo PLSV. Fonte: Elaboração própria.

Para o *insert* entre viagens, a atividade escolhida passa então a fazer parte de uma nova viagem criada, conforme ilustrado na Figura 15, que exemplifica a inserção de uma atividade entre viagens em PLSVs distintos.

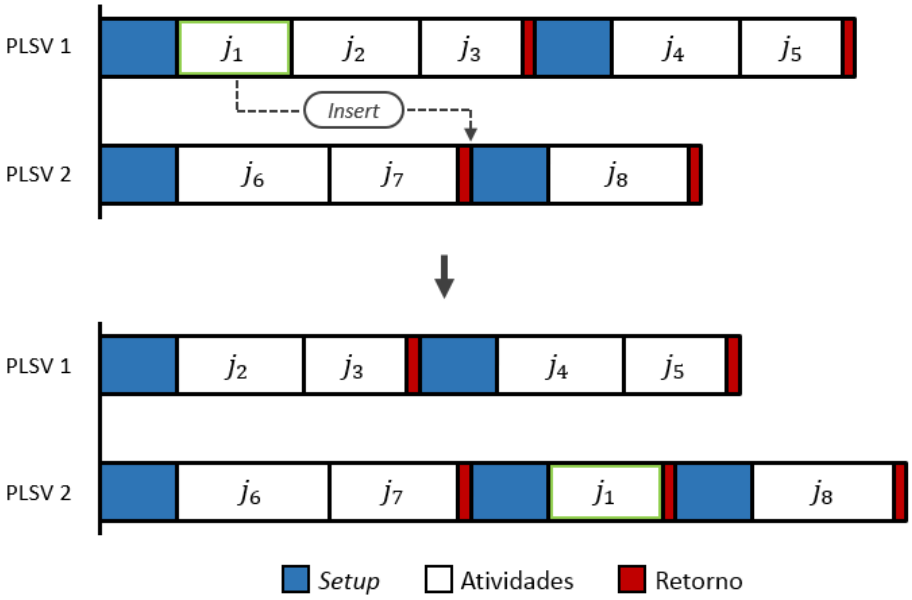


Figura 15 - *Insert* de atividade entre viagens em PLSVs distintos. Fonte: Elaboração própria.

Para que uma inserção seja realizada são necessárias validações para atestar se o movimento cumpre a restrição de capacidade de uma viagem, além de ser necessária a realocação das viagens, com base nas datas de liberação das atividades afetadas pela modificação.

Para o tratamento do problema em questão foi desenvolvida uma busca local *BL.INSERT*, utilizando a vizinhança *insert* como base. A busca é feita a partir de todas as atividades programadas no horizonte observado, onde são testadas posições para inserção - tanto dentro de viagens existentes como entre viagens. Sempre que uma solução melhor é encontrada, o movimento é imediatamente aceito e o método passa para a atividade seguinte, testando novas inserções em busca de melhorias. Os testes são feitos a partir de uma hierarquia que define se as inserções se iniciarão dentro do próprio PLSV de origem da atividade ou um PLSV distinto e se será realizada em uma viagem existente ou entre viagens. A hierarquia utilizada é indicada no fluxograma exposto na Figura 16.

De forma a atender às restrições do problema, alguns aspectos devem ser respeitados, impossibilitando que algumas inserções sejam efetuadas. São eles:

- Atividades do tipo *manifold* não podem mudar de navio;
- O movimento de uma atividade entre navios, só é permitido se ambos pertencerem à mesma família;
- Paradas preventivas para manutenção não podem mudar de embarcação.

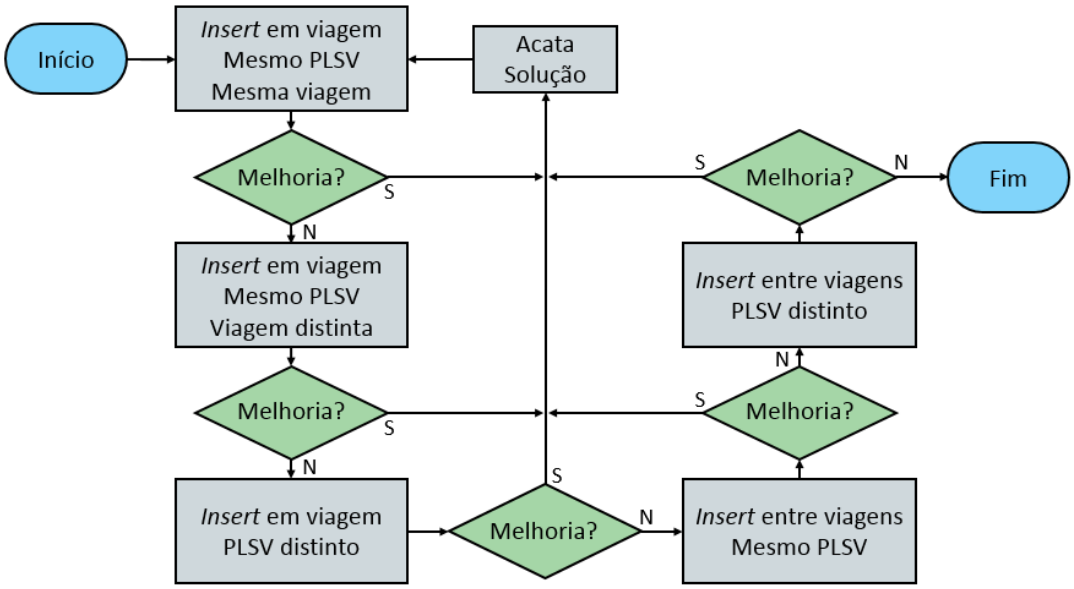


Figura 16 - Fluxograma de hierarquia da busca local. Fonte: Elaboração própria.

5.3.3
Perturbação

A perturbação corresponde a pequenas modificações aleatórias em uma solução existente, com o intuito de fugir de ótimos locais distantes da solução ótima global para o problema. Quanto maior o número de perturbações efetuadas, ou seja, mais elementos dentro de uma solução forem modificados, mais próximo de um método aleatório o algoritmo se comporta, modificando drasticamente a solução como se estivesse gerando uma nova solução inicial para o problema sem relação com a solução que a originou (Lourenço *et al*., 2010).

Desta forma, é importante que se defina quantos movimentos serão realizados durante a perturbação e a vizinhança a ser utilizada para estes movimentos. Para o presente estudo foi utilizada a vizinhança *swap*, buscando divergir daquela utilizada na busca local, minimizando as chances de o método voltar para o ótimo local encontrado anteriormente à perturbação.

A vizinhança *swap* consiste em escolher duas atividades programadas quaisquer e efetuar a troca de uma pela outra (Dong *et al.*, 2009). O *swap* pode ser realizado em atividades contidas em uma mesma viagem ou em viagens distintas, sejam elas de um mesmo PLSV ou de PLSVs diferentes. Os exemplos de *swap* em um mesmo PLSV e entre PLSVs distintos são mostrados nas Figura 17 e Figura **18**, respectivamente.

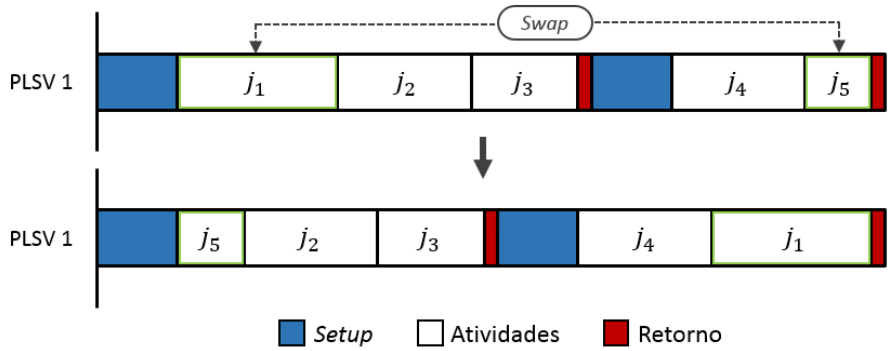


Figura 17 - *Swap* de atividades em viagens distintas no mesmo PLSV. Fonte: Elaboração própria.

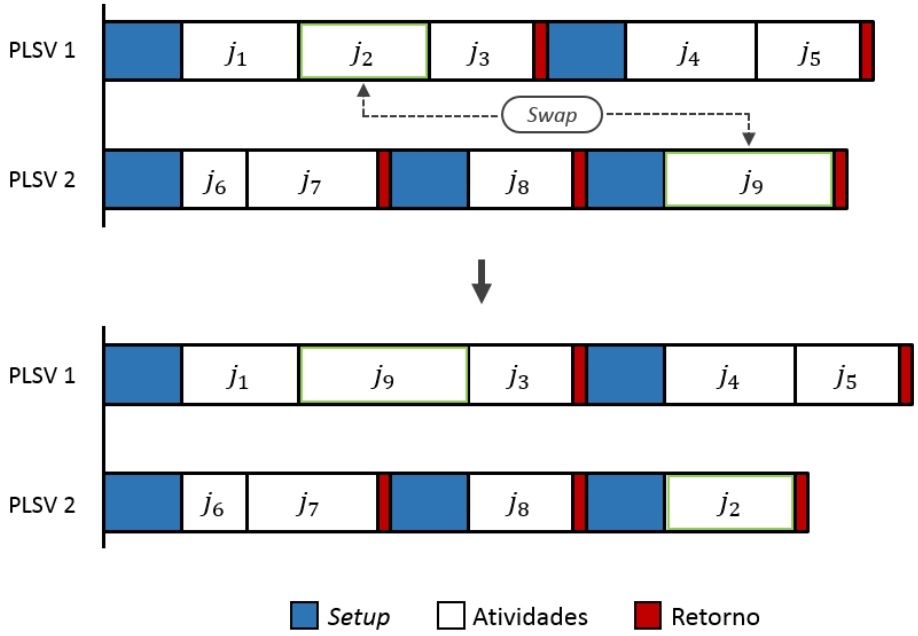


Figura 18 - *Swap* de atividades em viagens distintas em diferentes PLSVs. Fonte: Elaboração própria.

Devido à característica de elegibilidade dos navios, as perturbações realizadas só podem trocar elementos da solução entre navios de mesma família. Levando em consideração este aspecto e as variações quanto ao número total de navios por família e número total de atividades executadas em cada uma dessas famílias, o procedimento de perturbação a ser utilizado é feito de forma proporcional por família. Um fator de perturbação $\delta \in [0, 1]$ define o percentual a ser perturbado com relação ao montante de atividades executadas em cada família. A partir do conjunto de famílias L $(l = 1 \ldots |L|)$ e dos subconjuntos de atividades n_l executadas em cada família, o número de movimentos ou perturbações realizadas por família é calculado pela equação abaixo:

$$Perturbações_l = \max\{1, \lfloor \delta \times n_l \rfloor\}$$

Cada movimento corresponde ao sorteio aleatório de duas atividades e realização de um *swap* entre elas, trocando de posição uma com a outra. O procedimento de perturbação proposto, aqui definido como *nSWAP,* recebe a solução de cada família a ser perturbada e o número de perturbações a realizar para cada uma delas e realiza as trocas de forma aleatória.

Após a perturbação ser aplicada e a busca local executada, a nova solução precisa ser avaliada para que se verifique a aceitação ou não desta. A especificação do critério utilizado é feita a seguir.

5.3.4 Critério de aceitação

O critério de aceitação visa determinar se a solução s'^{*} obtida a partir da busca local realizada na solução perturbada s' deve ou não ser aceita como a melhor solução s^{*} vigente. Dependendo de como se define tal critério, diferentes estratégias de busca por soluções são determinadas, entre estratégias de diversificação e intensificação. Uma estratégia de alta intensificação corresponde em aceitar somente soluções estritamente melhores do que a vigente enquanto uma de alta diversificação diz respeito a aceitação de qualquer nova solução oriunda da busca. É importante que se encontre um balanço entre essas estratégias, definindo uma configuração que melhor se adapte ao problema abordado (Lourenço *et al.*, 2010).

Foi utilizado para o problema de reprogramação de PLSVs aqui tratado um critério proporcional de aceitação. A partir de um fator de aceitação $\theta \in [0, 1]$ são aceitas novas soluções inferiores à solução utilizada como base para avaliação, se $f(s'^{*}) \leq f(s^{*}) \times (1 + \theta)$.

5.3.4.1 Avaliação por família de navio

Devido à especificidade do problema abordado com relação às famílias de navio existentes, a aceitação ou não de uma nova solução é feita por família. A cada iteração do ILS todas as famílias são avaliadas independentemente, assim, aquelas que apresentem soluções que atendam ao critério de avaliação especificado são aceitas e substituídas na solução completa. Portanto, a solução gerada após cada iteração pode ser resultado de uma combinação de famílias aceitas pelo critério e outras que não passaram e mantiveram assim sua estrutura original. Na Figura 19 um exemplo em um cronograma com duas famílias é mostrado. Nesse exemplo, uma solução s qualquer passa pelas etapas de perturbação e busca local dando

origem a uma solução s'^{*} a ser avaliada. Com o indicativo de melhora apenas na família 1, uma nova solução s^{*} é então construída a partir da composição da família 1 obtida em s'^{*} e da família 2 da solução s original. Dessa forma, cada solução s pode ser desmembrada em um conjunto de soluções s_l para todas as famílias l tratadas.

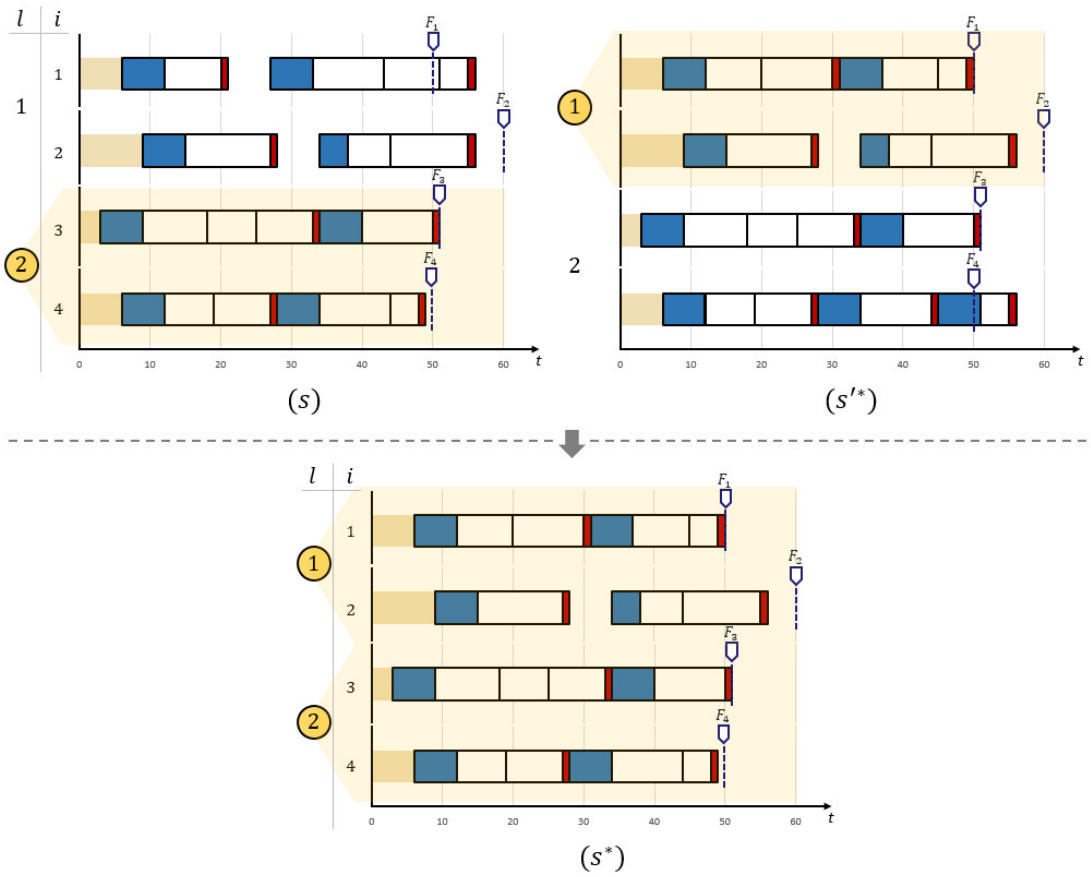


Figura 19 - Combinação de soluções entre famílias. Fonte: Elaboração própria.

A composição dessa solução pode ou não gerar uma melhora quando a mesma é avaliada em sua formação completa. Isso ocorre devido à componente AP (Atraso de poço) da função objetivo, que é determinada a partir do navio que está realizando a última atividade de cada um dos poços, podendo estes possuírem atividades programadas em navios de diferentes famílias. Por esse motivo, após a obtenção da solução completa, a mesma precisa ser avaliada com relação a sua função objetivo global, incluindo todos os navios programados. Essa avaliação não tem como objetivo rejeitar a solução, ou seja, esta continua sendo aceita como solução vigente para a próxima rodada, porém, é avaliado se na função objetivo global ela pode ser considerada a melhor solução conhecida até então.

A utilização da composição da solução por famílias possibilita que boas soluções sejam identificadas e construídas sem que o método tenha efetivamente passado por essa solução durante a sua execução. Essa técnica reduz o tempo necessário para a melhoria da solução inicial e usa um conceito específico do problema em benefício do método de resolução proposto.

5.3.5
Critério de Parada

Foram utilizados dois critérios de parada para o problema abordado, utilizando também nesse caso a característica de divisão da solução em famílias, interrompendo a execução do algoritmo em partes quando um número máximo de iterações sem melhoria é atingido por cada uma das famílias. São consideradas como iterações com melhoria apenas aquelas nas quais a solução de uma rodada obtida para uma família é estritamente melhor do que a melhor solução conhecida até então, não sendo consideradas como melhoria o fato da solução passar pelo critério de aceitação.

O parâmetro μ define o número máximo de iterações sem melhoria para as famílias a ser utilizado no $CritérioParada\_1$. Um contador marca o número de rodadas sem melhoria para cada família e compara com o máximo permitido. Caso uma família atinja o limite fixado, a mesma não é considerada nas iterações seguintes do ILS. A segunda forma de retirar a família do laço de repetição, ou seja, parar sua avaliação no algoritmo, é através do $CritérioParada\_2$ que depende de dois fatores a serem cumpridos simultaneamente, que são:

- Valor da função objetivo zerada na família ($f(s_l) = 0$);
- Todos os poços com atividades atreladas a essa família, com o atraso mínimo possível em todas as famílias existentes.

O atraso mínimo possível para cada poço k pode ser determinado com base nas datas de liberação r_j, tempo de *setup* s_j e tempo de processamento p_j das atividades j vinculadas a esse poço e com a sua data de compromisso d_k, de acordo com a equação a seguir:

$$AtrasoMínimo_k = \max\{0, \max\{r_j + s_j + p_j - d_k\}, \ \forall j \in k\}$$

5.3.6
Pseudocódigo ILS implementado

Com base no conjunto L de famílias a reprogramar, nos subconjuntos n_l contendo o número de atividades em cada família $l \in L$, nos parâmetros μ (máximo

de iterações sem melhoria), θ (fator de aceitação) e δ (fator de perturbação) e nos procedimentos descritos anteriormente, o pseudocódigo do ILS aplicado ao problema de reprogramação de PLSVs pode ser visto na Figura 20.

```
1:  DADO (δ, θ, μ) FAÇA
2:  s_0 ← Solução de entrada após o ajuste
3:  s** ← BL.INSERT(s_0)
4:  s* ← s**
5:  FamíliasAtivas ← L
6:  Perturbações_l ← max{1, ⌊δ × n_l⌋}, ∀ l ∈ L
7:  IterSemMelhoria_l ← 0, ∀ l ∈ L
8:  ENQUANTO | FamíliasAtivas | > 0 FAÇA
9:      PARA cada l ∈ FamíliasAtivas
10:         s'_l ← nSWAP(s*_l, Perturbações_l)
11:         s'*_l ← BL.INSERT(s'_l)
12:         SE f(s'*_l) < f(s**_l) × (1 + θ)
13:             s*_l ← s'*_l
14:             SE f(s'*_l) < f(s**_l); IterSemMelhoria_l ← 0; FIM-SE
15:         SENÃO s*_l ← s**_l
16:             IterSemMelhoria_l ← IterSemMelhoria_l + 1
17:         FIM-SE
18:         SE IterSemMelhoria_l > μ OU CritérioParada_2
19:             FamíliasAtivas ← FamíliasAtivas\{l}
20:         FIM-SE
21:     FIM-PARA
22:     SE f(s*) < f(s**); s** ← s*; FIM-SE
23: FIM-ENQUANTO
24: RETORNAR (s**)
```

Figura 20 - Pseudocódigo do ILS implementado. Fonte: Elaboração própria

Na primeira linha do pseudocódigo, exibido na Figura 20, o algoritmo é iniciado a partir da parametrização proposta para a perturbação δ, a aceitação θ e o critério de parada μ. Na sequência a solução inicial (s_0) é recebida, oriunda do ajuste descrito na seção 5.3.1. Na linha 3, a solução s^{**} é armazenada, representando a melhor solução obtida até então, resultante da aplicação da busca local $BL.INSERT$ na solução inicial. Essa solução é copiada para a solução s^{*} na linha seguinte, solução esta que servirá como a que será iterada durante o algoritmo. Em seguida,

o conjunto $FamíliasAtivas$ é definido como uma cópia do conjunto de famílias (L) existentes em uma determinada instância de programação tratada. Além disso, são definidos, nas linhas 6 e 7, o número de perturbações que serão realizadas em cada uma das famílias e iniciado o contador que define o número de iterações totais sem melhoria decorridos por família, com o valor zero.

Após as definições descritas, o laço principal do algoritmo é iniciado na linha 8, sendo este repetido enquanto o conjunto $FamíliasAtivas$ possuir elementos. Dito isso, para cada família contida nesse conjunto, as etapas do ILS são executadas, onde, a solução de iteração s_l^* da família é perturbada por meio do procedimento $nSWAP$, dando origem a solução s_l' que passa então pela busca $BL.INSERT$, resultando na solução $s_l'^*$, conforme descrito nas linhas 10 e 11.

Após essas etapas, a solução gerada é avaliada, na linha 12, por meio do critério de aceitação proposto, comparada com a melhor solução obtida (s_l^{**}) para a família até o momento. Em sendo aceita, a solução passa a ser a solução de iteração vigente e tem o seu contador de iterações sem melhoria zerado. Caso contrário, a solução é substituída pela melhor solução (s_l^{**}) obtida até então para a família, tendo o seu contador de iterações sem melhoria incrementado em uma unidade.

Em seguida, o algoritmo verifica, conforme indicado na linha 18, se algum dos critérios de parada propostos foram atingidos. Em caso afirmativo, a família é retirada do conjunto $FamíliasAtivas$. Ao final da execução das etapas do ILS para todas as famílias ativas, o algoritmo verifica, na linha 22, se a solução completa (s^*) composta pela junção das soluções de todas as famílias é melhor do que a melhor solução obtida (s^{**}). Quando o conjunto $FamíliasAtivas$ encontra-se vazio, a execução é interrompida e o valor de s^{**} é retornado, conforme indicado na última linha do algoritmo, sendo esta a solução final gerada.

5.4
Estrutura Geral do método de reprogramação

Após a apresentação de todas as etapas que compõem a metodologia proposta para o problema de reprogramação das embarcações PLSV, a Figura 21 exibe o fluxograma que estrutura o passo a passo executado, subdividindo a resolução em duas etapas, sendo a primeira delas a de viabilização e a segunda, de aprimoramento. Na etapa de viabilização são realizados o corte do horizonte e os ajustes descritos na etapa de geração de solução inicial, responsáveis por tornar o cronograma factível em consideração às especificidades do problema tratado. Em

seguida, na etapa de aprimoramento, visa-se refazer a programação buscando por melhores soluções, com base na metaheurística ILS, descrita anteriormente nesse capítulo.

A fase de viabilização, independentemente da fase de aprimoramento, contém importantes ferramentas de apoio a decisão para os programadores. Isso se dá principalmente pela identificação de forma automática de inconsistências na programação. Dessa forma, o programador pode ser alertado caso alguma irregularidade seja gerada durante a gestão das programações. A ferramenta pode ser utilizada tanto na programação quanto na reprogramação da frota e pode ser facilmente incorporada ao processo atual realizado pela companhia.

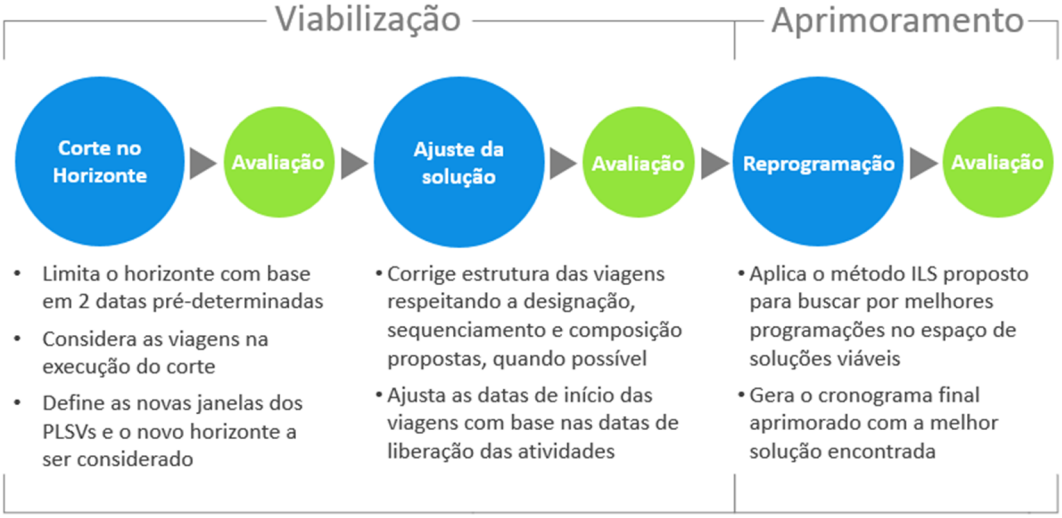


Figura 21 - *Framework* do método de reprogramação dos PLSVs. Fonte: Elaboração própria.

6
Aplicação e Resultados

Neste capítulo serão apresentadas as etapas de aplicação do método de reprogramação dos cronogramas dos PLSVs de acordo com o *framework* exposto no capítulo anterior. Foram utilizados dados reais de uma programação das embarcações, fornecida pela companhia que detém o processo, onde foram realizadas as etapas: corte no horizonte, ajuste da solução e reprogramação. No ínicio do capítulo, os coeficientes associados à função objetivo são definidos e assim serão empregados ao longo de todos os experimentos. Em seguida, uma instância base é desenvolvida a partir dos dados recebidos e utilizada para a parametrização da metaheurística ILS. Diversos experimentos utilizando a instância base são realizados, sendo o capítulo concluído com testes em diferentes instâncias, derivadas dos dados reais, para que se avalie a robustez e eficiência do algoritmo em cronogramas variados. O algoritmo foi codificado em C++ e compilado com Visual C/C++ utilizando a otimização /Ox. Todos os testes foram realizados em ambiente Windows 7 de 64 bits com processador Intel Core i5-4440 CPU @ 3.10GHz e 16GB de memória RAM.

6.1
Coeficientes da função objetivo

Com o intuito de seguir uma hierarquia coerente com o impacto acarretado por cada um dos fatores associados à função de avaliação, alinhando-a com os objetivos da companhia quanto à reprogramação do cronograma, os coeficientes α, β e γ foram estipulados com os valores de 0,6, 0,35 e 0,05 respectivamente. Dessa forma, a função a ser minimizada, a partir de uma programação de PLSVs π qualquer, pode ser descrita como:

$$Minimizar: f(\pi) = \sum_{i=1}^{m} (0{,}6 \times AP_i + 0{,}35 \times EN_i + 0{,}05 \times OC_i), \forall i \in M$$

Essa função objetivo (FO) será utilizada em todos os testes a seguir, desde a etapa de parametrização do ILS até a apuração dos resultados finais do método.

6.2
Instância base

Para a parametrização do ILS foi utilizada uma instância com três meses de uma programação real das embarcações PLSV, proveniente de um cronograma de 12 meses, gerada por meio do procedimento de corte descrito na Seção 5.3.1.1. O cronograma completo, compreendido entre $t = 1$ e $t = 365$, foi limitado com base nos instantes $t_1 = 90$ e $t_2 = 180$. Esse intervalo foi definido por ser o que melhor representa o problema real, dado o período no qual os dados foram fornecidos pela companhia. Sendo este, o próximo horizonte a ser tratado pelos programadores. A programação resultante é indicada na Figura 22, com um total de 13 embarcações de quatro famílias distintas, sendo 12 ativas e uma inativa, e 55 atividades programadas, com 14 delas associadas a sete poços críticos. A embarcação 7 foi considerada inativa por sua indisponibilidade no intervalo definido.

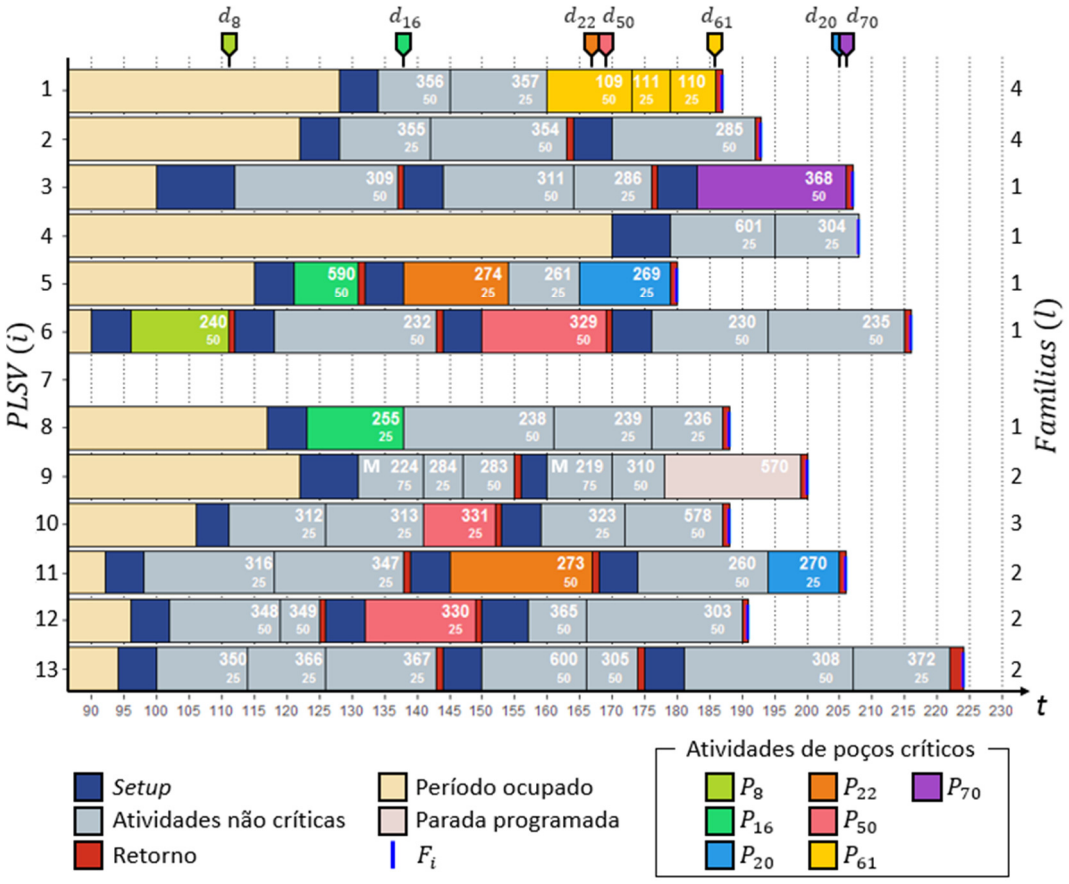


Figura 22 - Cronograma cortado a partir da instância proposta. Fonte: Elaboração própria

Cada atividade programada é indicada com seu código individual no canto superior direito e com a sua taxa de ocupação abaixo de seu código. Atividades do tipo *manifold* são assinaladas com um M no canto superior esquerdo e não podem mudar de embarcação. Além disso, um código de cores destaca aquelas vinculadas

a poços críticos (k) e que devem respeitar as suas datas de compromisso (d_k) indicadas na parte superior da figura. Cada embarcação tem a sua janela superior (F_i) destacada e sua família informada à direita do cronograma. A programação da empresa não apresenta problemas quanto aos indicadores da função objetivo, contudo, inconsistências relacionadas aos tempos de *setup*, descumprimento de datas de liberação e capacidade de viagens são observadas. A Tabela 1 sintetiza a avaliação da solução pós corte, onde, 13 inconsistências foram detectadas.

Tabela 1 - Indicadores da solução cortada. Fonte: Elaboração própria.

Indicadores		**Solução cortada**
Viabilidade	Data de liberação violada	6 atividades
	Excesso de ocupação	4 viagens
	Setups indevidos	3 viagens
Função objetivo	Atraso Poço (AP)	0 dias
	Excedente Navio (EN)	0 dias
	Ociosidade (OC)	0 dias

Com o intuito de viabilizar o cronograma, eliminando as inconsistências identificadas, o procedimento de ajuste conforme descrito na Seção 5.3.1.2 foi realizado, dando origem a programação exibida na Figura 23.

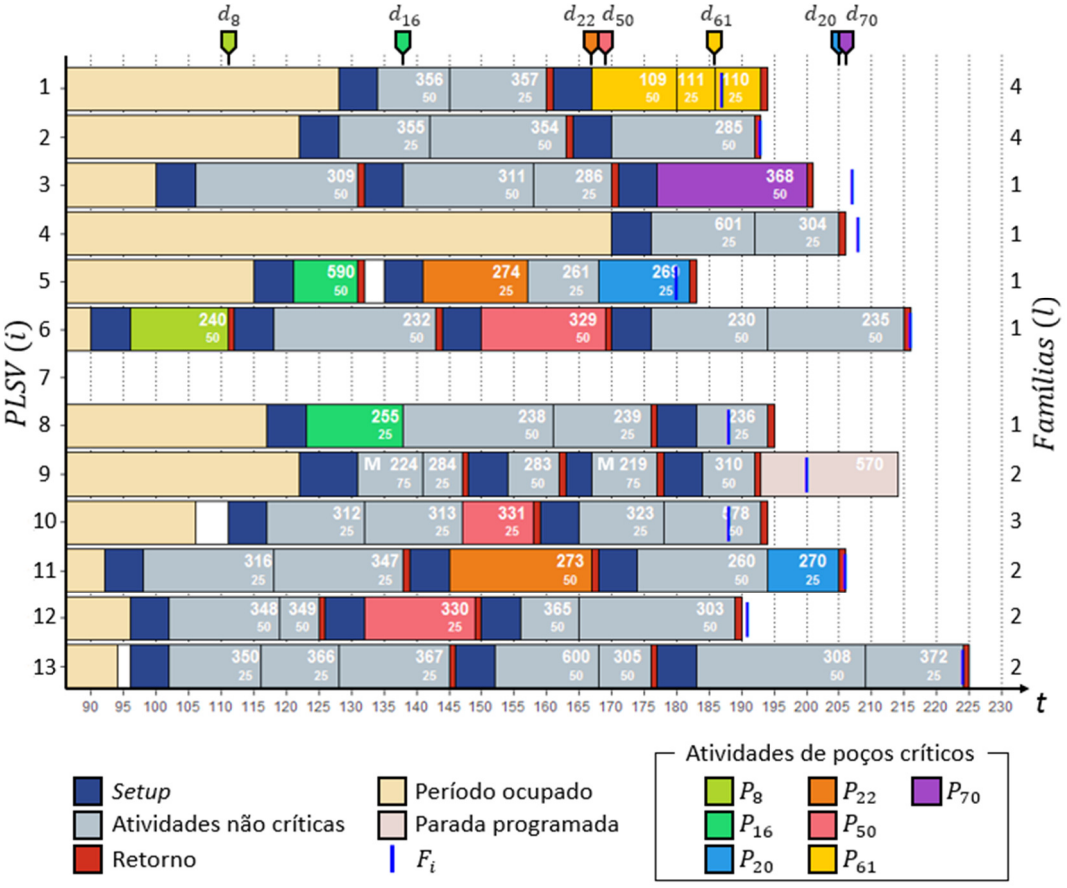


Figura 23 - Cronograma ajustado a partir da instância proposta. Fonte: Elaboração própria.

Na solução resultante, após etapa de ajuste, os indicadores de viabilidade aparecem zerados, assinalando que a nova solução está em acordo com todas as restrições impostas pelo problema, não obstante, alguns problemas relacionados à função objetivo passam a existir. Os resultados da avaliação dessa solução foram sintetizados na Tabela 2.

Tabela 2 - Indicadores da solução ajustada. Fonte: Elaboração própria.

Indicadores		**Solução ajustada**
Viabilidade	Data de liberação violada	0 atividades
	Excesso de ocupação	0 viagens
	Setups indevidos	0 viagens
Função objetivo	Atraso Poço (AP)	7 dias
	Excedente Navio (EN)	38 dias
	Ociosidade (OC)	10 dias

A solução ajustada passa a ser a solução inicial (π_0) para a reprogramação e pode ser avaliada a partir da função objetivo proposta, onde, com base nos indicadores destacados na Tabela 2 e nos coeficientes estabelecidos para cada componente da função, tem-se que $f(\pi_0) = 18$, conforme indica a equação abaixo:

$$f(\pi_0) = (0{,}6 \times 7) + (0{,}35 \times 38) + (0{,}05 \times 10)$$

Essa instância de programação serviu de base para os testes computacionais realizados para a parametrização do algoritmo e avaliação geral do método. O valor da sua função objetivo foi usado como referência para que se atestasse a melhoria percentual gerada pelo ILS durante os testes realizados.

Com o objetivo de conhecer a instância proposta de forma mais detalhada, foram realizados testes a partir de conhecimento tácito do problema e análises de combinação de atividades, sendo definida assim a melhor solução conhecida (*Best Known Solution* - BKS) π^* para essa instância com $f(\pi^*) = 0{,}7$. Dentre os três componentes da função objetivo, apenas o excedente navio não pôde ser zerado nesta solução, resultando em dois dias excedentes, um dia na família 3 e um dia na família 4. Esse excedente de dois dias multiplicado pelo coeficiente de 0,35 associado a esse critério na função objetivo resultou no valor de 0,7. O detalhamento que comprova esse excedente mínimo encontra-se no Apêndice II desse trabalho. A BKS obtida, exibida na Figura 24 e com seus indicadores sintetizados na Tabela 3, também foi usada para comparação durante os testes. O valor de sua função objetivo serve como referência para que se avalie a capacidade do método em gerar boas soluções.

A avaliação de forma comparativa entre a melhor solução conhecida (π^*) e a solução inicial (π_0) para a instância proposta indica uma diferença de 96,1%. Esse valor serve de referência para que sejam avaliadas as melhorias obtidas pelo ILS.

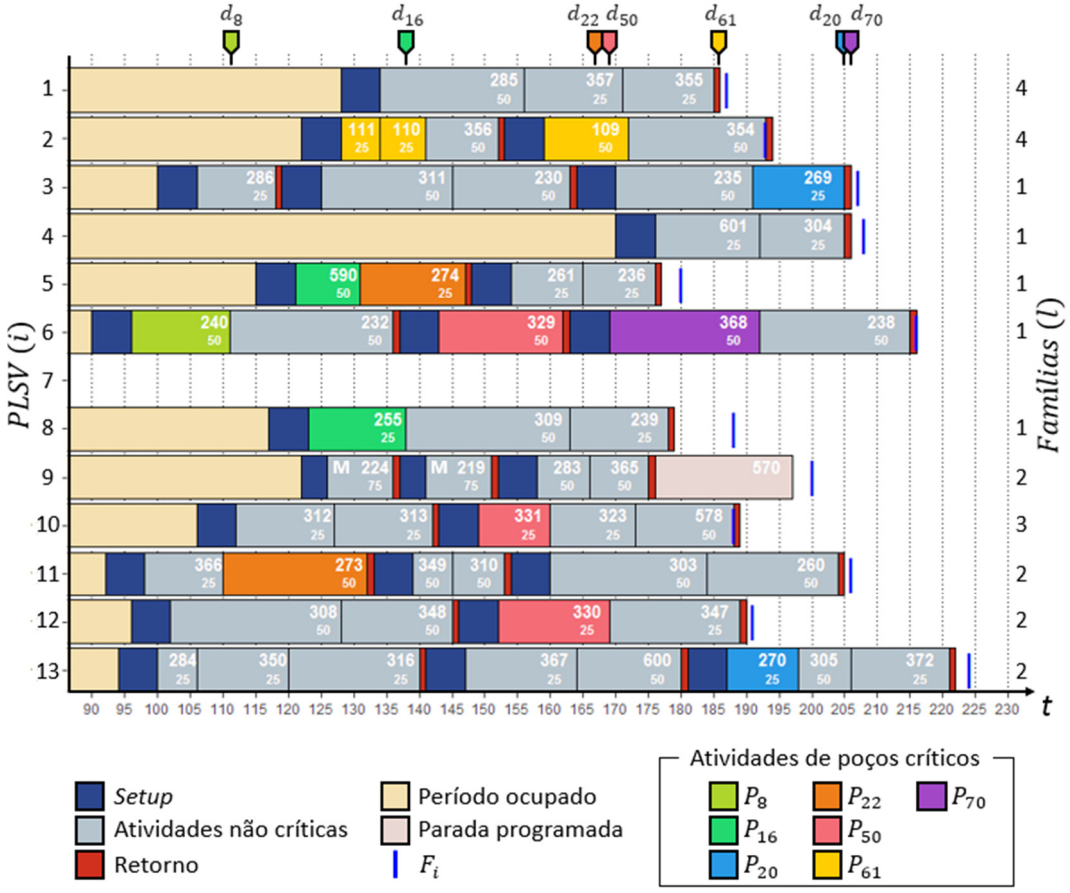


Figura 24 - Melhor solução conhecida para a instância proposta. Fonte: Elaboração própria.

Tabela 3 - Indicadores da melhor solução para a instância proposta. Fonte: Elaboração própria.

Indicadores		Melhor solução
Viabilidade	Data de liberação violada	0 atividades
	Excesso de ocupação	0 viagens
	Setups indevidos	0 viagens
Função objetivo	Atraso Poço (AP)	0 dias
	Excedente Navio (EN)	2 dias
	Ociosidade (OC)	0 dias

A programação exibida na Figura 24 não é a única capaz de resultar no valor de 0,7 da função objetivo, tendo em vista que a qualidade da solução não é afetada por antecipações dos poços críticos, ociosidade após a última atividade do navio e nem pela sequência de execução de atividades não críticas em uma mesma viagem. Contudo, é importante que seja demonstrada a existência de uma solução com o menor valor conhecido da função objetivo para a instância base.

6.3
Parametrização do ILS proposto

A etapa de parametrização da metaheurística foi feita com base em dez sementes distintas (100 a 109) e diversas combinações entre os parâmetros de perturbação $\delta = \{0{,}05, 0{,}10, 0{,}15, 0{,}20\}$, de aceitação $\theta = \{0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5, 0{,}6\}$ e de parada $\mu = \{50, 75, 100\}$. Ao todo foram realizadas 720 rodadas. Na Figura 25, cada ponto representa a média dentre todas as sementes para cada combinação entre os parâmetros, totalizando 72 pontos indicados no gráfico. Para uma melhor identificação, os pontos foram realçados por diferentes cores para os fatores de perturbação δ definidos. O gráfico indica a média da função objetivo encontrada entre as dez sementes testadas no eixo vertical e o tempo computacional médio exigido no eixo horizontal.

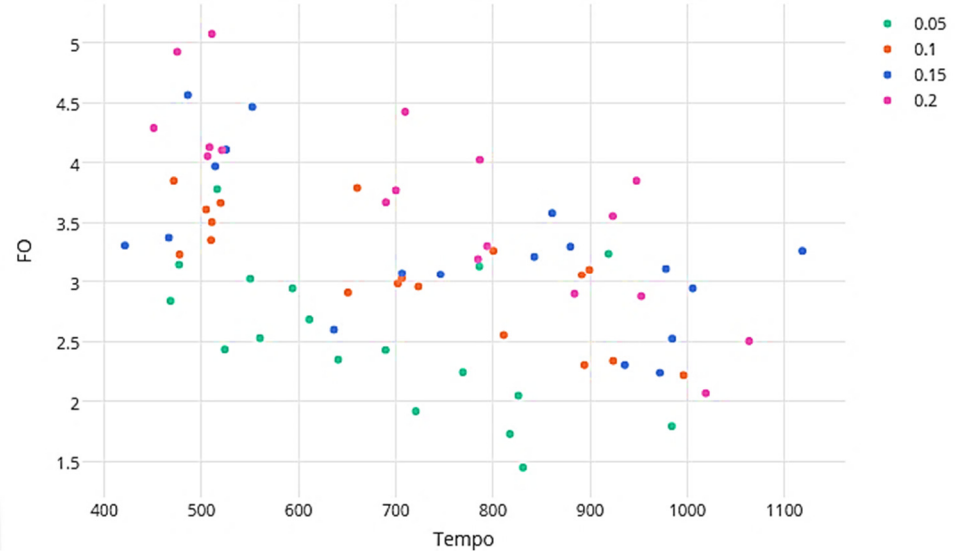


Figura 25 - Resultados da parametrização por fator de perturbação. Fonte: Elaboração própria.

Na parte superior do gráfico, observa-se que as perturbações maiores produzem piores resultados quando os fatores são avaliados comparativamente. Enquanto na parte inferior, o fator de perturbação $\delta = 0{,}05$ apresenta os melhores resultados médios, ocupando toda a região inferior do gráfico.

Na Figura 26 o mesmo conjunto de soluções é apresentado realçando-se, dessa vez, por número máximo de iterações sem melhoria (μ), onde é possível observar uma forte correlação entre o aumento desse parâmetro e a melhoria da solução. Contudo, esse incremento implica também no aumento do tempo computacional médio exigido. Dessa forma, uma análise mais aprofundada do *trade-off* entre qualidade da solução e tempo computacional, que esse critério

proporciona, será feita na obtenção dos resultados gerais da instância base após a definição dos fatores de perturbação e aceitação.

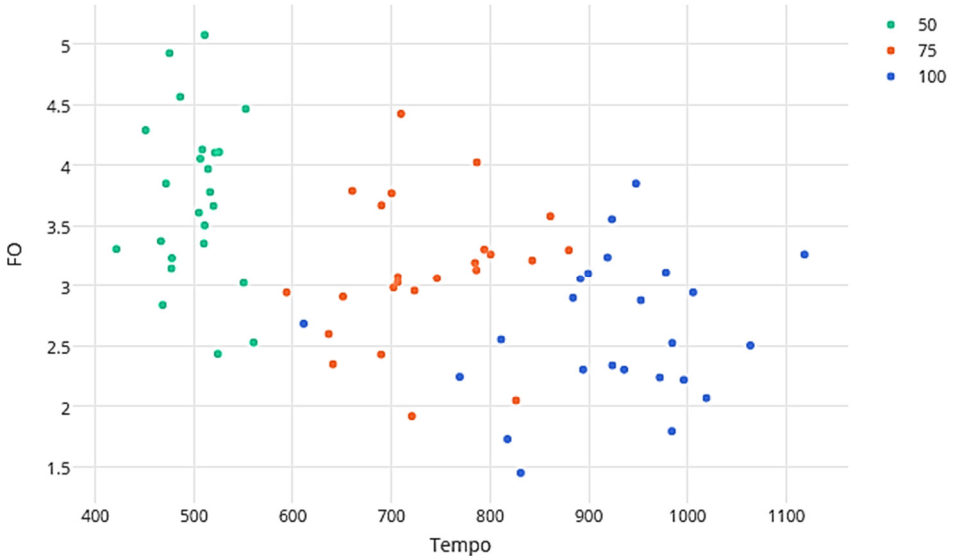


Figura 26 - Resultados da parametrização por critério de parada. Fonte: Elaboração própria.

A utilização do realce por fator de aceitação (θ) não foi capaz de prover evidências que apontassem para uma melhor definição desse parâmetro. Optou-se então pela definição prioritária do fator de perturbação (δ) para que em uma etapa posterior o fator de aceitação pudesse ser configurado.

Em outra análise envolvendo o comparativo entre os fatores de perturbação, percebe-se com base nos indicadores exibidos na Figura 27(a) que tanto a média geral da função objetivo quanto do tempo computacional exigido são menores para $\delta = 0{,}05$. Além disso, na Figura 27(b) o percentual de vezes em que a melhor solução conhecida (BKS) foi obtida é maior para essa configuração, com uma frequência acima do dobro de qualquer outra configuração para esse parâmetro.

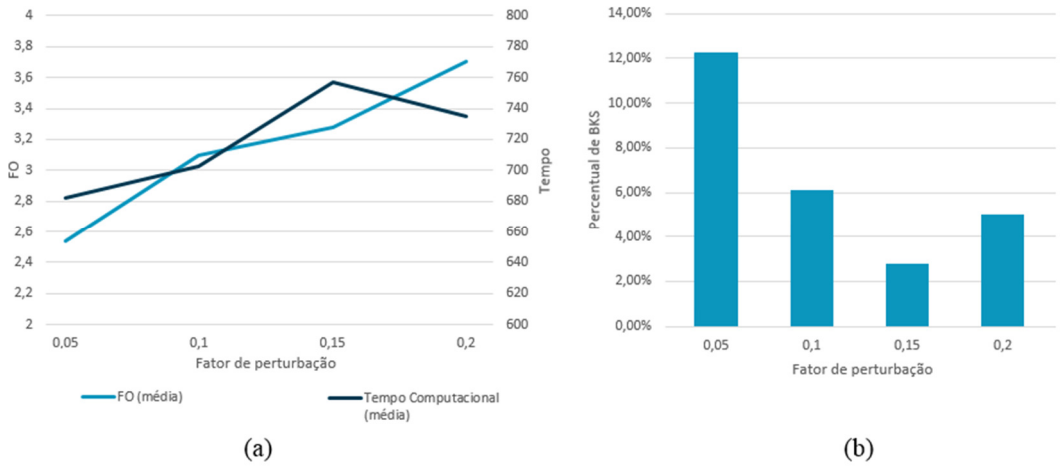


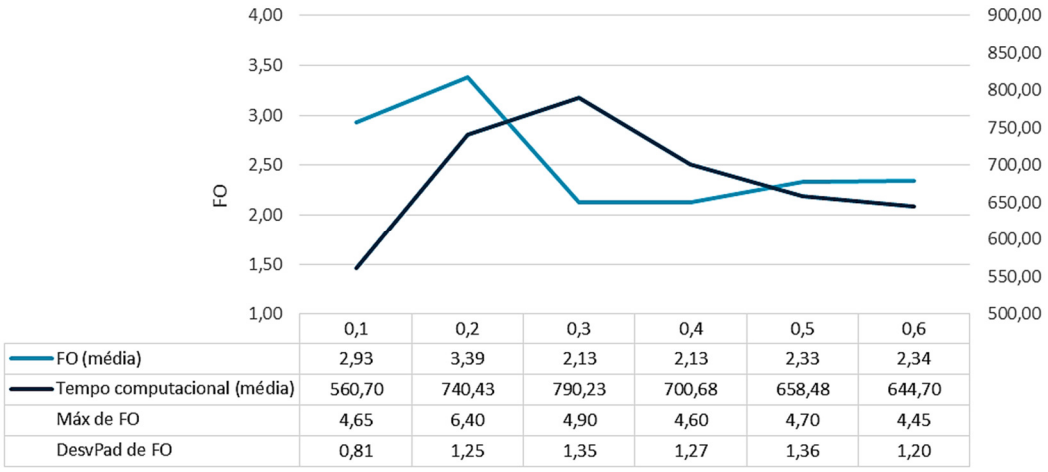
Figura 27 - Indicadores do fator de perturbação. Fonte: Elaboração própria.

A fim de consolidar a parametrização da perturbação, foram destacadas na Tabela 4 as médias das soluções encontradas na relação entre a perturbação e a aceitação, onde observa-se que em cinco dos seis fatores de aceitação propostos, a perturbação $\delta = 0{,}05$ se mostrou a melhor configuração.

Tabela 4 - Média da função objetivo na combinação entre parâmetros. Fonte: Elaboração própria.

Fator de aceitação	**Fator de Perturbação**			
	0,05	**0,1**	**0,15**	**0,2**
0,1	2,93	3,57	3,75	4,27
0,2	3,39	3,29	3,28	4,35
0,3	2,13	2,97	3,68	3,49
0,4	2,13	3,01	3,33	3,60
0,5	2,33	2,81	2,74	3,12
0,6	2,34	2,94	2,90	3,41

Dessa forma, o fator de perturbação δ foi definido como 0,05, onde foram então analisados os resultados para o fator de aceitação, com esse parâmetro fixado. Os resultados para os valores médios da função objetivo e dos tempos computacionais são exibidos na Figura 28. Observa-se que os melhores resultados encontrados estão nos fatores intermediários, com $\theta = 0{,}3$ e $\theta = 0{,}4$, que tiveram um valor médio de 2,13 na função objetivo. Contudo, a análise de outros critérios indica vantagem para $\theta = 0{,}4$ em relação a $\theta = 0{,}3$, por possuir um menor tempo médio computacional exigido em sua execução (700,68 segundos), menor desvio padrão (1,27) e menor valor máximo da função objetivo (4,6) na amostra gerada.



	0,1	0,2	0,3	0,4	0,5	0,6
FO (média)	2,93	3,39	2,13	2,13	2,33	2,34
Tempo computacional (média)	560,70	740,43	790,23	700,68	658,48	644,70
Máx de FO	4,65	6,40	4,90	4,60	4,70	4,45
DesvPad de FO	0,81	1,25	1,35	1,27	1,36	1,20

Figura 28 - Avaliação por fator de aceitação. Fonte: Elaboração própria.

Outras avaliações foram feitas a fim de confirmar a escolha da combinação $\delta = 0{,}05$ e $\theta = 0{,}4$. Para isso, o parâmetro μ que define o número máximo de iterações sem melhoria foi fixado, com os resultados médios para cada combinação

entre os fatores de perturbação e aceitação sintetizados na Figura 29, Figura 30 e Figura 31com $\mu = 50$, $\mu = 75$ e $\mu = 100$, respectivamente.

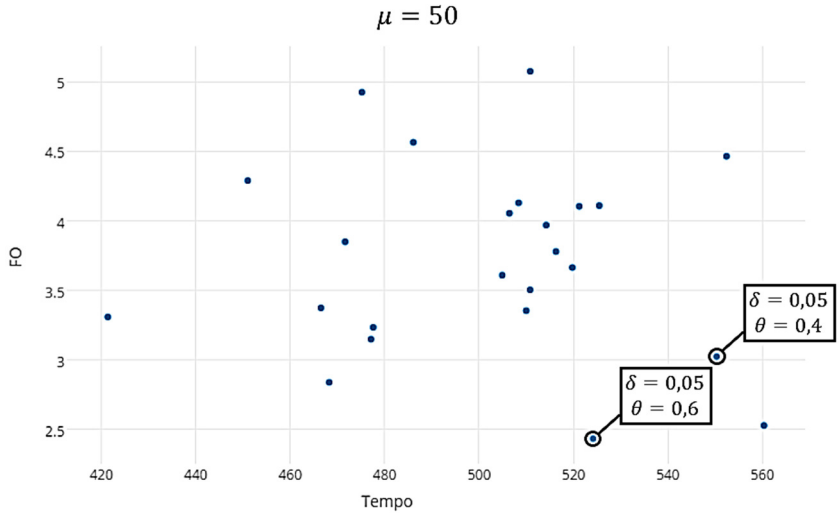


Figura 29 - Avaliação com máximo de 50 iterações sem melhoria. Fonte: Elaboração própria.

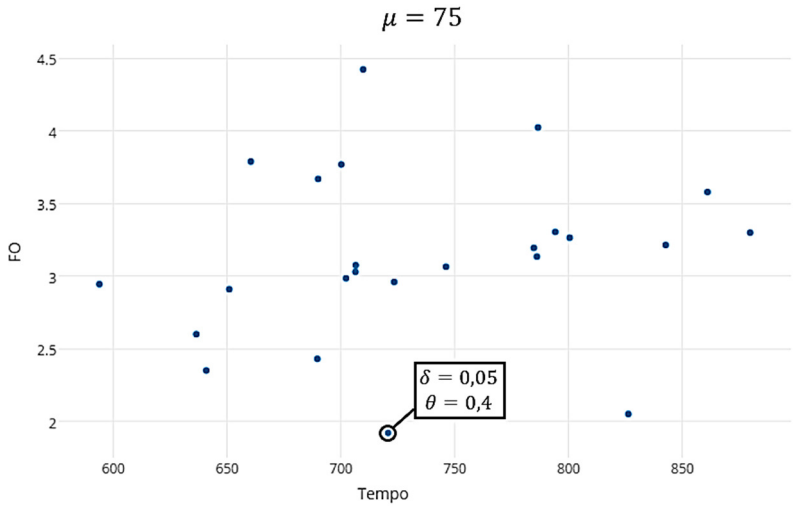


Figura 30 - Avaliação com máximo de 75 iterações sem melhoria. Fonte: Elaboração própria.

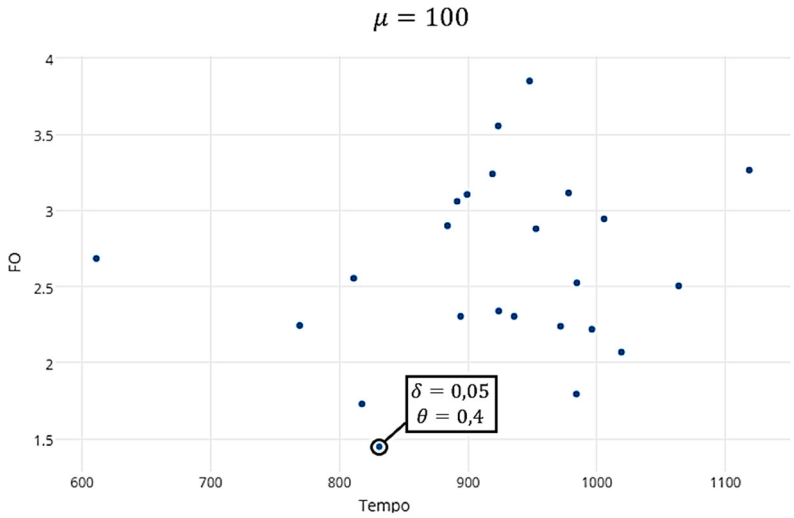


Figura 31 - Avaliação com máximo de 100 iterações sem melhoria. Fonte: Elaboração própria.

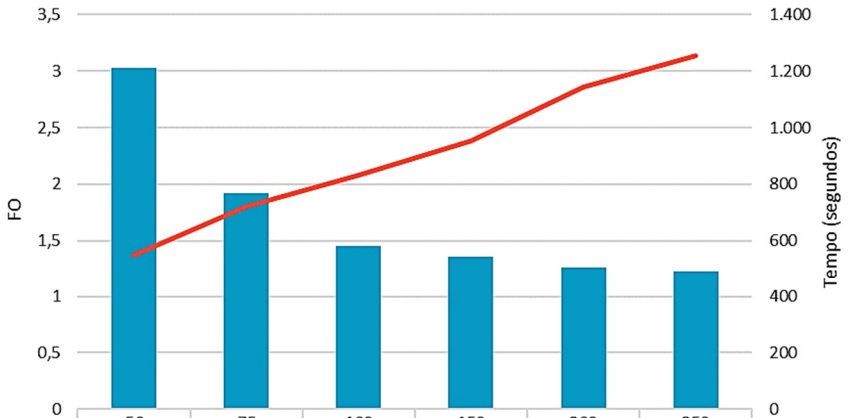
A análise por combinação de critérios indicou que em duas das três definições para μ escolhidas, a combinação $\delta = 0{,}05$ e $\theta = 0{,}4$ se mostrou a melhor opção, com tempos de execução bem abaixo da média geral e melhor valor de função objetivo. Essa configuração só não foi a melhor para $\mu = 50$, permanecendo porém, entre aquelas com melhores resultados. O pouco tempo de execução para $\mu = 50$ aumenta a variância dos resultados, tornando a análise menos precisa para esses testes. Dessa forma, os fatores foram estabelecidos como $\delta = 0{,}05$ e $\theta = 0{,}4$ e utilizados com essa configuração em todos os experimentos realizados a seguir.

6.4 Resultados gerais para a instância base

Após a parametrização dos dois principais fatores do ILS, novos testes computacionais foram realizados, utilizando a mesma instância base proposta na seção 6.2. Inicialmente, outros valores para o número máximo de iterações sem melhoria foram testados e, em seguida, dois testes com diferentes critérios de parada foram realizados. O primeiro deles considerando tempos fixos de execução do algoritmo, enquanto, o segundo utiliza a melhor solução conhecida (BKS), especificada na seção 6.2, como critério de parada do algoritmo.

6.4.1 Avaliação por máximo de iterações

Com o intuito de avaliar o impacto do critério de parada por número máximo de iterações, em seu *trade-off* entre qualidade da solução e tempo computacional, três novos valores (150, 200 e 250) para μ foram testados para as mesmas 10 sementes utilizadas anteriormente. Os resultados dos experimentos realizados para todos os valores de μ, que incluem os anteriores e os novos valores propostos, são exibidos na Figura 32. O gráfico exibe os valores médios para a função objetivo e para o tempo computacional, além de indicar o número de vezes na qual a melhor solução conhecida foi encontrada para cada teste e a melhoria percentual obtida.



	50	75	100	150	200	250
FO (média)	3,025	1,92	1,45	1,36	1,26	1,225
Tempo Computacional (média)	550	721	831	954	1.142	1.255
BKSs encontradas	1	0	3	4	5	6
Melhoria Percentual	83,19%	89,33%	91,94%	92,44%	93,00%	93,19%

Figura 32 - Resultados gerais por máximo de iterações sem melhoria. Fonte: Elaboração própria.

Conforme esperado, o comportamento apresentado nos testes iniciais se manteve, apontando para um aprimoramento na qualidade da solução conforme o número máximo de iterações sem melhoria aumenta. Todavia, como também era esperado, com incremento no tempo computacional exigido.

Resultados significativos foram alcançados, chegando a 93,19% de melhoria para $\mu = 250$, encontrando a melhor solução conhecida em seis das dez sementes utilizadas (60% dos casos). Considerando a melhoria de 96,1% como um limitante superior, conforme destacada na seção 6.2, o desvio percentual ficou em apenas 2,9% nesse caso.

Apesar de em determinado ponto o incremento na solução parecer dispensável, cada pequena melhoria pode evitar a perda de milhares de reais para a companhia. Analisando como exemplo o comparativo entre a função objetivo média para $\mu = 150$ e $\mu = 200$, com valores de 1,36 e 1,26 respectivamente, tem-se uma diferença de 0,1 entre as soluções, o que pode significar uma ociosidade média de dois dias. Isso representa o dispêndio de dois dias de navio, com um custo diário aproximado de US$ 300.000,00 por embarcação, sem que nenhuma atividade seja realizada.

Um tempo máximo considerado bom para a execução do algoritmo foi estipulado em aproximadamente 30 minutos de acordo com a expectativa dos programadores na companhia. Nesse caso, os resultados foram bem aderentes à expectativa, com duração média de 1255 segundos, ou aproximadamente 20 minutos, na maior configuração de μ.

6.4.2
Avaliação por tempo de execução

Por ser uma proposta de ferramenta de apoio a decisão aplicável ao processo interno da companhia que programa a frota de PLSVs, testes com tempos fixos de execução foram realizados, tornando o experimento mais aderente às expectativas dos gestores do processo. Desse modo, cinco tempos de execução foram testados (10, 20, 30, 45 e 60 minutos) para que pudesse ser observada a capacidade de melhoria do método ao longo do tempo. Os resultados são exibidos na Tabela 5.

Tabela 5 – Resultados consolidados por tempo limite de execução. Fonte: Elaboração própria.

Tempo Máximo (minutos)	BKSs encontradas	Função objetivo (média)	Melhoria percentual (média)
10	0	1,74	90,3%
20	3	1,13	93,7%
30	5	0,96	94,7%
45	7	0,82	95,4%
60	8	0,75	95,8%
Média Geral		**1,08**	**94,0%**

Os resultados por tempo limite de execução mostram a capacidade que o método tem de encontrar boas soluções conforme o seu tempo de execução aumenta, indicando que o mesmo consegue fugir de ótimos locais e continuar melhorando conforme se mantém rodando. Para o tempo limite de 60 minutos, a média dos resultados para a função objetivo é bem próxima da melhor solução conhecida, sendo esta encontrada em 80% das rodadas. A diferença percentual entre a melhoria média obtida e a que foi calculada para a melhor solução conhecida (96,1%) é de apenas 0,3%. Além disso, a execução limite de 30 minutos se mostrou uma estratégia eficiente, capaz de melhorar em 94,7%, com uma solução média de 0,96 e desvio de apenas 1,1% quando comparada com a melhoria de 95,8% para a rodada com 60 minutos, na metade do tempo despendido.

Na Figura 33 os resultados obtidos para cada semente são indicados como um ponto no gráfico, onde pode ser observado o intervalo das soluções da amostra e a respectiva média no valor da função objetivo para cada tempo fixado. Fica evidente por essa análise a tendência em achar resultados dentro de uma região menor conforme o tempo de execução é ampliado, aumentando a acurácia do algoritmo para tempos fixos maiores.

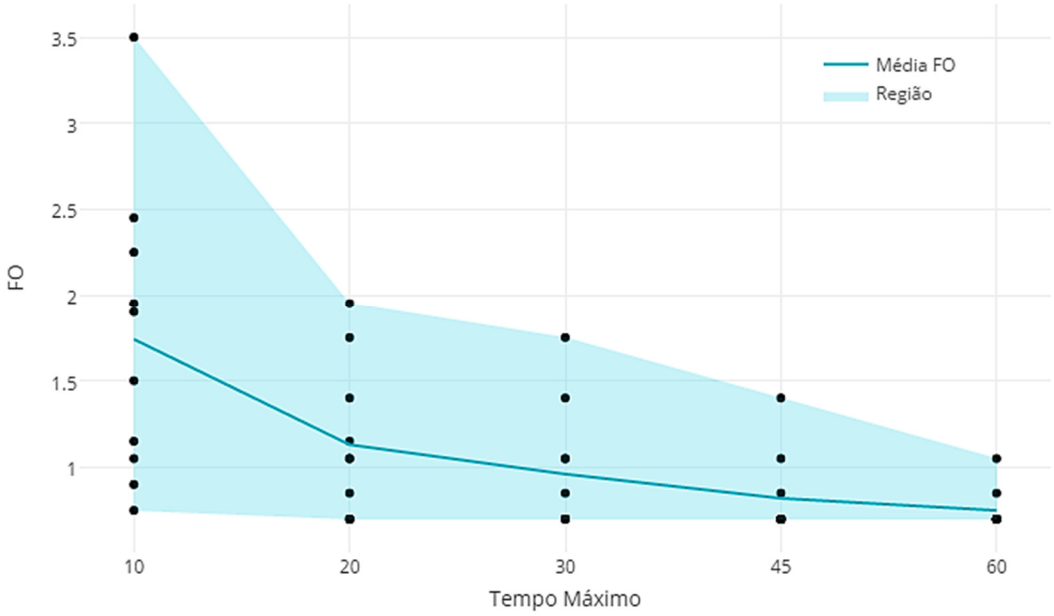


Figura 33 – Resultados gerais por tempo limite de execução. Fonte: Elaboração própria.

Essa análise é importante para que se avalie a confiabilidade do método caso apenas uma rodada seja realizada, o que pode vir a ser o caso, sendo implementada uma ferramenta de reprogramação que utilize o algoritmo como base. Dessa forma, fica evidente que o tempo de execução em 60 minutos é capaz de garantir resultados mais eficazes devido a menor variabilidade na qualidade das soluções encontradas, todavia, o tempo de execução ultrapassa o tempo estabelecido pelos programadores de PLSV.

6.4.3 Tempo para o alvo (*Time to target*)

Utilizar a melhor solução conhecida como critério de parada ajuda a avaliar a capacidade do método em encontrá-la, considerando o tempo necessário despendido para tal como um indicador de performance do algoritmo (Chiarandini *et al.*, 2007). Dito isso, a melhor solução conhecida para a instância base foi definida como solução alvo, fazendo com que a execução do algoritmo fosse interrompida no momento em que o valor da função objetivo de 0,7 fosse atingido. Foram testadas 50 sementes (100 a 149) com as configurações de $\delta = 0{,}05$ e $\theta = 0{,}4$. Como segundo critério de parada, a fim de evitar laços de execução ininterruptos e estipular um limite aceitável, foi definido um tempo limite de 7.000 segundos (aproximadamente 2 horas) de execução, sendo armazenado o tempo decorrido até que a solução alvo fosse encontrada, caso a mesma de fato obtida. Os resultados são apresentados na Figura 34.

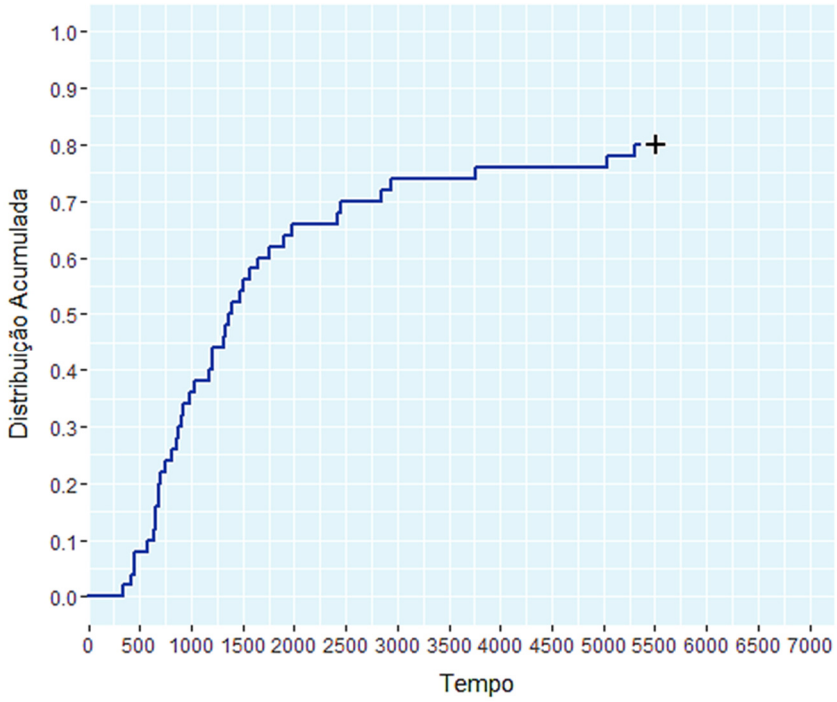


Figura 34 – Distribuição acumulada dos tempos para atingir o alvo. Fonte: Elaboração própria.

No exemplo exposto, uma medida de performance X (tempo de execução) é descrita em termos da sua distribuição de probabilidade acumulada empírica, dada pela equação (Chiarandini *et al.*, 2007):

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \leq x)$$

O fato da curva aparecer truncada indica que nem todos os testes foram capazes de encontrar a solução alvo com sucesso no limite de tempo estipulado, sendo esta alcançada em 80% dos casos. Nesses 80%, a solução foi encontrada em um tempo inferior a 5500 segundos (aproximadamente 1h30 minutos), porém, em 70% dos casos o tempo decorrido é menor do que a metade desse valor, abaixo dos 2500 segundos (aproximadamente 40 minutos). Isso significa que se o algoritmo for executado por 2500 segundos, a probabilidade de encontrarmos a solução alvo é de 70%. Indicando que apenas 10% das sementes testadas, dentre as que foram capazes de encontrar a solução alvo, exigiram um tempo computacional acima desses 2500 segundos.

Esse estudo é importante para que se avalie a probabilidade do método em achar a solução alvo ao longo de sua execução. Se definirmos, por exemplo, o tempo de 30 minutos (ou 1800 segundos) de execução, conforme a expectativa dos

programadores, temos uma probabilidade aproximada de 60% de encontrar a melhor solução conhecida, para essa instância, com esse tempo computacional.

6.5 Aplicação em outras instâncias

Os testes realizados com a instância base ajudaram a conhecer o algoritmo e avaliar o impacto dos diferentes parâmetros e critérios em sua aplicação. Todavia, a utilização de apenas uma instância limita a avaliação a apenas uma composição de cronograma com características particulares daquele período programado. Dessa forma, com o intuito de avaliar a sensibilidade e a capacidade do método frente a diferentes cronogramas, com variadas composições de atividades, novas instâncias foram geradas a partir da programação disponibilizada pela companhia. Foram desenvolvidas nove instâncias novas a partir de diferentes intervalos de corte. A diferença de 90 dias entre os instantes t_1 e t_2 foi mantida, sendo então avaliadas dez instâncias ao todo, incluindo a instância base. Os resultados consolidados dos testes a partir das 10 sementes utilizadas anteriormente (100 a 109) foram sintetizados na Tabela 6.

Tabela 6 – Resultados gerais com máximo de 250 iterações sem melhoria. Fonte: Elaboração própria.

Instância (t_1 a t_2)	m	n	$f(\pi_0)$	**Melhor solução obtida**	**FO média**	**FO (dp)**	**Melhoria percentual média**	**Tempo médio**	**Tempo (dp)**
90 a 180	12	55	18	0,7	1,22	0,93	93,2%	1255	327
105 a 195	13	59	21,3	0,7	2,21	0,94	89,6%	1230	216
120 a 210	13	59	17,9	0	0,91	0,72	94,9%	1862	791
135 a 225	13	58	22,45	2,1	2,96	0,91	86,8%	3747	854
150 a 240	13	56	23,05	1,4	2,98	1,16	87,1%	2597	891
165 a 255	13	58	22,45	0	0,08	0,22	99,6%	938	416
180 a 270	12	57	23,1	0,7	2,02	0,94	91,2%	1617	556
195 a 285	13	55	11,65	0,35	0,52	0,18	95,5%	328	114
210 a 300	13	52	10,3	0,35	0,45	0,17	95,6%	901	349
225 a 315	13	39	6,85	0,7	1,01	0,26	85,2%	1284	863
Média Geral							**91,9%**	**1576**	

Cada instância é diferenciada na tabela pelos instantes (t_1 e t_2) escolhidos para delimitar seu horizonte. Algumas informações sobre a solução de entrada gerada pelo corte e que definem a composição da instância são destacadas na parte esquerda da tabela, que são: número de navios programados (m), número de atividades programadas (n) e a qualidade da solução inicial ($f(\pi_0)$) obtida após a viabilização da solução de entrada.

Os demais campos da tabela informam os resultados obtidos a partir do ILS, indicando a melhor solução obtida em cada instância para o conjunto de sementes testadas, os resultados para a média e o desvio padrão (dp) tanto para o valor da função objetivo (FO) quanto para o tempo computacional exigido em segundos e o percentual de melhoria indica pela redução percentual obtida no valor da FO a partir do ILS.

Uma melhoria média geral de 91,9% entre todas as instâncias, em um tempo computacional médio de 1576 segundos (aproximadamente 26 minutos), é apontada nos resultados. O tempo médio está dentro de um limite de execução aderente ao processo da companhia. Contudo, para algumas instâncias o tempo extrapola a média geral, chegando a ultrapassar uma hora de execução para a instância com $t_1 = 135$ e $t_2 = 225$.

Essa variabilidade dos tempos de execução pode não ser adequada para o caso da implementação de uma ferramenta computacional de apoio a decisão, que auxilie o processo de reprogramação. Todavia, a solução ótima (função objetivo zerada) foi encontrada para a instância com $t_1 = 165$ e $t_2 = 255$, que obteve uma melhoria média de 99,6%, em um tempo médio de execução de 937,9 segundos (ou aproximadamente 16 minutos) e um dos menores desvios padrão para a função objetivo (0,22). A instância com $t_1 = 120$ e $t_2 = 210$ também conseguiu alcançar o valor ótimo, contudo, ultrapassando o tempo médio esperado de 30 minutos em aproximadamente 60 segundos. Sete instâncias tiveram tempos médios abaixo do dos 30 minutos, sendo o menor deles para a instância com $t_1 = 195$ e $t_2 = 285$, de 328 segundos (aproximadamente 5 minutos) e desvio padrão de 114 segundos, com solução média para a FO (0,52) bem próxima ao menor valor encontrado (0,35) e com um dos menores desvios padrão para a FO (0,18) dentre todas as instâncias avaliadas.

Devido à variabilidade dos tempos de execução, conforme destacado anteriormente, novos testes foram feitos com as mesmas instâncias, utilizando dessa vez como critério de parada um tempo fixo de 30 minutos de execução. Esse tempo foi definido devido aos bons resultados obtidos na etapa de teste com a instância base em um tempo adequado ao processo da companhia. O intuito é avaliar a capacidade de melhoria do método em um tempo controlado. Os resultados foram consolidados na Tabela 7.

Tabela 7 - Resultados gerais com 30 minutos de tempo limite de execução. Fonte: Elaboração própria

Instância (t_1 a t_2)	m	n	$f(\pi_0)$	**Melhor solução obtida**	**FO média**	**FO (dp)**	**Melhoria percentual média**
90 a 180	12	55	18	0,7	0,96	0,36	94,7%
105 a 195	13	59	21,3	0,7	2,21	0,94	89,6%
120 a 210	13	59	17,9	0	0,99	0,72	94,5%
135 a 225	13	58	22,45	2,1	3,9	1,25	82,6%
150 a 240	13	56	23,05	1,4	3,25	1,28	85,9%
165 a 255	13	58	22,45	0	0,08	0,22	99,6%
180 a 270	12	57	23,1	0,7	2,12	0,93	90,8%
195 a 285	13	55	11,65	0,35	0,52	0,18	95,5%
210 a 300	13	52	10,3	0,35	0,45	0,17	95,6%
225 a 315	13	39	6,85	0,7	1,01	0,26	85,2%
Média Geral							**91,4%**

De forma similar à tabela exibida nos testes por máximo de iterações, na Tabela 7 são indicadas as instâncias, suas características e os respectivos resultados obtidos com a execução do algoritmo. O tempo médio foi suprimido, dado que, trata-se de um parâmetro fixo estipulado previamente.

Como destacado anteriormente, o uso de tempos fixos de execução para uma ferramenta de suporte à decisão pode ser uma opção mais adequada ao ambiente de negócios, por facilitar a gestão do processo e proporcionar um maior controle da ferramenta. Todavia, essa limitação pode implicar em queda na qualidade da solução para determinadas instâncias, bem como no aproveitamento do tempo, o que ocorre quando o algoritmo já não consegue melhorar a solução por muitas iterações consecutivas e se mantém em execução pelo tempo estipulado. Como o tempo médio para os testes com $\mu = 250$ era praticamente equivalente ao tempo de execução fixo proposto, a melhoria média geral alcançada se mostrou semelhante entre as duas abordagens com 91,4% por tempo fixo e 91,9% por máximo de iterações. Como era de se esperar, as instâncias que demandaram um tempo de execução superior aos 30 minutos nos testes por máximo de iterações apresentaram queda na qualidade da solução média. Em contrapartida, a instância com $t_1 = 90$ e $t_2 = 180$ obteve avanço significativo entre um critério e outro passando de uma melhoria de 93,2% para 94,7% na execução por tempo fixo em 30 minutos.

De forma geral, as abordagens apresentam vantagens e desvantagens em sua aplicação quando comparadas entre si, porém, com uma condição em comum entre as duas, que é a de prover bons resultados e melhorias significativas, com uma média que ultrapassa os 91%, por meio da etapa de reprogramação. Mesmo para

instâncias mais complexas, as melhorias proporcionadas pelo método ficam acima de 82% chegando a 99,6% no melhor dos casos. Além disso, o desvio padrão dos valores obtidos para a função objetivo ficaram baixos em ambas as abordagens, provando a robustez do método quanto a melhoria das soluções durante a etapa de reprogramação da frota PLSV.

7
Considerações finais

Os principais objetivos dessa dissertação foram mapear o processo de reprogramação das embarcações PLSV, desenvolver uma ferramenta computacional baseada na metaheurística ILS, capaz de mimizar o impacto operacional nos cronogramas de curto prazo das embarcações, e aplicá-la em uma programação real fornecida pela companhia estudada.

O problema é tratado como base em modelos de programação de máquinas paralelas idênticas (P_m), em que m navios precisam atender a uma demanda composta por n atividades de interligação de poços submarinos. Vernalha *et al*. (2008), Queiroz e Mendes (2012) e Moura (2012) utilizaram essa abordagem para a programação de PLSVs. Contudo, com simplificações, a fim de aproximar o problema daqueles tratados na literatura de programação de máquinas paralelas, tornando-o menos aderente ao problema real enfrentado. Uma das simplificações é a consideração de que as viagens são definidas em uma etapa prévia, desvinculada da programação, onde a partir de então, cada viagem passa a ser tratada como uma única atividade a ser programada. Isso evita a consideração de tempos de *setup* e a preocupação com mudanças na composição ou no sequenciamento das atividades dentro de cada viagem. Outra simplificação feita considera que cada viagem atende a um único poço específico, e assim, o término de uma viagem determina a conclusão do poço a ela associado. Facilitando a determinação do instante de conclusão de cada poço.

O método proposto no presente trabalho estende o tratamento existente na literatura, considerando todas as especificidades do problema real de reprogramação dos PLSVs, respeitando as particularidades inerentes ao processo de interligação dos poços, regras de negócio e objetivos da companhia. Dessa forma, o trabalho foi capaz de expandir o conhecimento relativo aos problemas de reprogramação de frotas marítimas, desenvolvendo uma ferramenta aplicável e adequada ao processo diário vivenciado pelos programadores.

São diversas as características que tornam o problema aqui abordado único e de tão complexa resolução, entre elas: a composição de viagens com atividades de poços distintos, a alocação de atividades de um mesmo poço em viagens e PLSVs diferentes, o cálculo do término do poço com base em um conjunto de atividades a ele associadas, as atividades que não podem mudar de embarcação, a forma como os *setups* e a ocupação de cada viagem são calculados, o conceito das famílias de embarcações e o uso de uma função objetivo não regular, específica para o problema.

Apesar da complexidade do problema de reprogramação dos PLSVs, o método de resolução utilizado com base na metaheurística ILS se mostrou uma abordagem eficiente para o problema. O ILS foi capaz de reprogramar o cronograma das embarcações, provendo melhorias significativas, em um tempo computacional adequado ao processo real, enquanto mantém a viabilidade da programação para com as restrições do problema. Duas abordagens principais foram consideradas visando a manutenção de um tempo médio de execução de 30 minutos, proposto pelos programadores. A primeira delas com um critério de parada após 250 iterações sem melhoria para cada família de PLSVs e a segunda com um tempo fixo de execução de 30 minutos. O uso do conceito de famílias de PLSV para a resolução de subconjuntos da programação se mostrou uma forma eficiente de se atingir melhores soluções de forma mais ágil. O método faz uso de uma particularidade do problema, construindo soluções que poderiam não ser encontradas pelo ILS caso aplicado a programação completa.

A partir de uma função multicritério não regular capaz de medir o impacto operacional na programação, aferindo a ociosidade das embarcações e postergações na entrada em operação dos poços, obteve-se uma melhoria média superior a 91% para ambas as abordagens. Essa melhoria percentual é calculada na comparação entre o valor da função objetivo das soluções encontradas na etapa prévia à aplicação do ILS e das soluções aprimoradas após o método.

A sensibilidade aos dados de entrada mostra variações na melhoria proporcionada pelo método para diferentes instâncias, porém, com percentuais de melhoria em um intervalo que vai de 82,6% a 99,6%, com 30 minutos de execução média. Sendo assim, o objetivo foi alcançado, tendo sido desenvolvido um método capaz de evitar a perda de milhões de reais para a companhia em poucos minutos de execução, se mostrando uma boa base para o desenvolvimento de uma

ferramenta de apoio a decisão para o processo de reprogramação das embarcações PLSV. A capacidade do método em propor novas programações a partir dos dados recebidos em um curto espaço de tempo pode ser um fator decisivo para a melhoria de um processo que, atualmente, é feito de forma manual e que não garante a viabilidade das soluções desenvolvidas. O programador pode então contar com uma ferramenta de auxílio que agiliza a proposição de cronogramas eficientes enquanto garante o cumprimento de todas as restrições do processo.

7.1 Sugestões para trabalhos futuros

Alguns aspectos importantes podem ser incrementados ao método proposto, a fim de que aprimorá-lo - tanto no aspecto de geração de boas soluções quanto em seu tempo de execução. Dito isso, algumas recomendações para trabalhos futuros foram relacionadas, como:

- Utilizar um critério de parada que se adapte durante a execução do algoritmo. A ideia é minimizar a variabilidade nos tempos de execução para a abordagem por máximo de iterações como critério de parada, adaptando esse número máximo conforme o algoritmo é executado.
- Avaliar diferentes coeficientes para a função objetivo. Os coeficientes da função objetivo têm um importante papel na avaliação da qualidade da solução e consequentemente na execução do algoritmo. Com isso, testar variações nesses coeficientes, respeitando a hierarquia proposta, pode ser uma alternativa para avaliar o método de forma mais ampla.
- Avaliar diferentes parametrizações no ILS implementado. Os testes para definição dos parâmetros a serem utilizados no ILS utilizam um conjunto limitado de possibilidades. Ampliar e criar intervalos maiores entre fatores que propiciaram bons resultados podem ajudar a aumentar a precisão do método.
- Variar o porte das instâncias. Apesar de se tratar de um problema específico que lida com a reprogramação de três meses, o teste com base em problemas maiores ajuda a avaliação da capacidade do método em resolver problemas mais complexos.
- Utilizar diferentes buscas locais. A aplicação das buscas é determinante para a qualidade da solução gerada pelo método, dessa forma, avaliar diferentes buscas locais e diferentes ordens de execução dessas buscas, podem ser

formas de aperfeiçoá-lo. Um teste simples a ser feito sem a exigência de desenvolvimento é a inversão entre as buscas utilizadas, usando o *insert* para a perturbação e o *swap* para melhoria da solução no ILS.

- Testar diferentes metaheurísticas. Cada metaheurística tem as suas características particulares e capacidade em prover bons resultados para diferentes tipos de problema. Desse modo, avaliar diferentes métodos pode ajudar a encontrar aquele que melhor se adapte a reprogramação dos PLSVs.
- Adicionar incertezas ao problema. A consideração de dados probabilísticos relacionados ao tempo de execução das atividades e às suas datas de liberação, entre outros fatores, podem ajudar a minimizar os impactos no processo, reduzindo por consequência a necessidade de ajustes a serem feitos durante a reprogramação.
- Desenvolvimento de um modelo matemático para o problema. A aplicação por modelagem pode auxiliar na constatação das soluções ótimas para instâncias do problema, tornando possível a comparação entre diferentes métodos e avaliando a relação qualidade da solução e tempo computacional exigido entre eles.

8
Referências bibliográficas

AZIZOGLU, M.; ALAGÖZ, O. Parallel-machine rescheduling with machine disruptions. **IIE Transactions**, v. 37, n. 12, p. 1113-1118, 2005.

BAI, Y.; BAI, Q. **Subsea Engineering Handbook.** Burlington – MA: Elsevier, 2010.

BARBOZA, L. da S. **Pipe laying support vessel - PLSV Medusa.** 2015. Disponível em: <http://www.oceanica.ufrj.br/deno/prod_academic/relatorios/2015/Luanda/relat1/plsv_luanda_rell.htm>. Acesso em: 16 fev. 2016.

BATHRINATH, S.; SANKAR, S. S.; PONNAMBALAM, S. G.; KANNAN, B. K.V. Bi-objective optimization in identical parallel machine scheduling problem. In: **Proceedings of the 4th International Conference on Swarm, Evolutionary, and Memetic Computing.** Chennai, India, 2013. p. 377-388.

BATHRINATH, S.; SANKAR, S. S.; PONNAMBALAM, S. G.; LENO, I. J. VNS-based heuristic for identical parallel machine scheduling problem. In: **Proceedings of the ICAEES 2015 - Artificial Intelligence and Evolutionary Algorithms in Engineering Systems.** Hyderabad, India, 2015. p. 693-699.

BECK, J. C.; PROSSER, P.; SELENSKY, E. On the reformulation of vehicle routing problems and scheduling problems**.** In: **Proceedings of the 5th International Symposium on Abstraction, Reformulation, and Approximation.** Alberta, Canadá, 2002. p. 282-289.

BECK, J. C.; PROSSER, P.; SELENSKY, E. Vehicle routing and job shop scheduling: What's the difference?. In: **Proceedings of the 13rd ICAPS – International Conference on Automated Planning and Scheduling.** Trento, Itália, 2003. p. 267-276.

BEHERA, D. K. Complexity on parallel machine scheduling: A review. In: **Proceedings of the INCOSET 2012 – International Conference on Emerging Trends in Science, Engineering and Technology.** Tiruchirappalli, Tamilnadu, India, 2012. p. 373-381.

BISKUP, D.; HERRMANN, J.; GUPTA, J. N. Scheduling identical parallel machines to minimize total tardiness. **International Journal of Production Economics**, v. 115, n. 1, p. 134-142, 2008.

CHEN, B.; POTTS, C. N.; WOEGINGER, G. J. A review of machine scheduling: Complexity, algorithms and approximability. In: DU, D.-Z.; PARDALOS, P. M. (Eds) **Handbook of combinatorial optimization**. Nova York: Springer, 1998. p. 1493-1641.

CHENG, T. C. E.; SIN, C. C. S. A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, v. 47, n. 3, p. 271-292, 1990.

CHIARANDINI, M.; PAQUETE, L.; PREUSS, M.; RIDGE, E. **Experiments on metaheuristics: Methodological overview and open issues.** Technical Report DMF-2007-03-003, The Danish Mathematical Society, Dinamarca, 2007.

CHRISTIANSEN, M.; FAGERHOLT, K.; NYGREEN, B.; RONEN, D. Maritime transportation. **Handbooks in Operations Research and Management Science**, v. 14, p. 189-284, 2007.

CHRISTIANSEN, M.; FAGERHOLT, K.; NYGREEN, B.; RONEN, D. Ship routing and scheduling in the new millennium. **European Journal of Operational Research**, v. 228, n. 3, p. 467-483, 2013.

CHRISTIANSEN, M.; FAGERHOLT, K.; RONEN, D. Ship routing and scheduling: status and perspectives. **Transportation Science**, v. 38, n. 1, p. 1-18, 2004.

CRUZ, R. C.; LOURENÇO, H. R.; COELHO, V. N.; SOUZA, M. J. F.; GRASAS, A. TWTJSSP-ILS: Um algoritmo heurístico para resolver o problema job-shop scheduling com penalidade pelo tempo de atraso. In: **SBPO 2013 – XLV Simpósio Brasileiro de Pesquisa Operacional**, Natal – RN, 2013.

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

EDIS, E. B.; OGUZ, C.; OZKARAHAN, I. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. **European Journal of Operational Research**, v. 230, n. 3, p. 449-463, 2013.

FANJUL-PEYRO, L.; RUIZ, R. Iterated greedy local search methods for unrelated parallel machine scheduling. **European Journal of Operational Research**, v. 207, n. 1, p. 55-69, 2010.

FRAMINAN, J. M.; LEISTEN, R.; GARCÍA, R. R. **Manufacturing scheduling systems**. Londres: Springer, 2014.

FRENCH, S. **Sequencing and scheduling**: An introduction to the mathematics of the job-shop. Chichester: Ellis Horwood, 1982.

GENDREAU, M.; POTVIN, J. Y. **Handbook of metaheuristics.** Nova York: Springer. 2010. v. 2.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2002. 61p.

GOKHALE, R.; MATHIRAJAN, M. Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing. **The International Journal of Advanced Manufacturing Technology**, v. 60, n. 9-12, p. 1099-1110, 2012.

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.

HAMZADAYI, A.; YILDIZ, G. Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. **Computers & Industrial Engineering**, v. 91, p. 66-84, 2016.

JUAN, A. A.; LOURENÇO, H. R.; MATEO, M.; LUO, R.; CASTELLA, Q. Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. **International Transactions in Operational Research**, v. 21, n. 1, p. 103-126, 2014.

KAABI, J.; HARRATH, Y. A survey of parallel machine scheduling under availability constraints. **International Journal of Computer and Information Technology**, v. 3, n. 2, p. 238-245, 2014.

KAID, H.; ALHARKAN, I.; GHALEB, A.; GHALEB, M. A. Metaheuristics for identical parallel machines scheduling to minimize mean tardiness. In: **Proceedings of the IEOM 2015 – International Conference on Industrial Engineering and Operations Management.** Dubai, Emirados Árabes Unidos, 2015. p. 1-6.

KOUKI, Z.; CHAAR, B. F.; HAMMADI, S.; KSOURI, M. Analogies between flexible job shop scheduling and vehicle routing problems. In**: Proceedings of the IE & EM 2007 - International Conference on Industrial Engineering and Engineering Management**. Singapura, 2007. p. 880-884.

LABANCA, E. L. **Metodologia para a seleção de arranjos submarinos baseada na eficiência operacional.** 2005. 92p. Dissertação (Mestrado em Engenharia Oceânica) – COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ. 2005.

LABBI, W.; BOUDHAR, M.; OULAMARA, A. Scheduling two identical parallel machines with preparation constraints. **International Journal of Production Research**, DOI: 10.1080/00207543.2014.978032, p. 1-18, 2014.

LALLA-RUIZ, E.; VOß, S. Minimizing total tardiness on identical parallel machines using vns with learning memory. In: **Proceedings of the 9th LION - International Conference on Learning and Intelligent Optimization.** Lille, França, 2015. p. 119-124.

LAM, K.; XING, W. New trends in parallel machine scheduling. **International Journal of Operations & Production Management**, v. 17, n. 3, p. 326-338, 1997.

LAWRENCE, S. A. **International sea transport**: the years ahead. Massachusetts: Lexington Books, 1972.

LEHMER, D. H. Mathematical methods in large-scale computing units. In: **Proceedings of the 2nd Symposium on Large-Scale Digital Calculating Machinery.** Cambridge, Massachusetts, 1951. p. 141-146.

LI, K.; YANG, S. L. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. **Applied Mathematical Modelling**, v. 33, n. 4, p. 2145-2158, 2009.

LIAO, C. J.; CHAO, C. W.; CHEN, L. C. An improved heuristic for parallel machine weighted flowtime scheduling with family set-up times. **Computers & Mathematics with Applications**, v. 63, n. 1, p. 110-117, 2012.

LIU, L.; ZHOU, H. On the identical parallel-machine rescheduling with job rework disruption. **Computers & Industrial Engineering**, v. 66, n. 1, p. 186-198, 2013.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search: Framework and applications. In: GENDREAU, M.; POTVIN, J. Y. (Eds.). **Handbook of metaheuristics**. Nova York: Springer, 2010. p. 363-397.

LUSTOSA, L.; MESQUITA, M. A.; QUELHAS, O.; OLIVEIRA, R. J. **Planejamento e controle da produção**. Rio de Janeiro: Elsevier, 2008.

MA, Y.; CHU, C.; ZUO, C. A survey of scheduling with deterministic machine availability constraints. **Computers & Industrial Engineering**, v. 58, n. 2, p. 199-211, 2010.

MELLOULI, R.; SADFI, C.; CHU, C.; KACEM, I. Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times. **European Journal of Operational Research**, v. 197, n. 3, p. 1150-1165, 2009.

MENDES, A. B. **Programação de frota de apoio a operações "offshore" sujeita à requisição de múltiplas embarcações para uma mesma tarefa.** 2007. 224p. Tese (Doutorado em Engenharia Naval e Oceânica) - Universidade de São Paulo, USP, São Paulo, SP, 2007.

MG VOWGAS. **Subsea engineering.** 2016. Disponível em: http://www.mgvowgas.com/what-do-we-do/subsea-engineering. Acesso em: 16 fev. 2016.

MOKOTOFF, E. Parallel machine scheduling problems: a survey. **Asia-Pacific Journal of Operational Research**, v. 18, n. 2, p. 193-242, 2001.

MORAIS, J. M. de. **Petróleo em águas profundas:** uma história tecnológica da Petrobras na exploração e produção offshore. Brasília: R C IPEA: Petrobras, 2013.

MOURA, V. C. **Programação de frota de embarcações de lançamento de dutos.** 2012. 73f. Dissertação (Mestrado em Engenharia Naval e Oceânica) – Escola Politécnica da Universidade de São Paulo, USP, São Paulo, SP, 2012.

NAGAR, A.; HADDOCK, J.; HERAGU, S. Multiple and bicriteria scheduling: A literature survey. **European Journal of Operational Research**, v. 81, n. 1, p. 88-104, 1995.

NATIONAL OILWELL VARCO. **National oilwell varco completes flexible pipe acquisition**. 2012. Disponível em: <https://www.nov.com/News_and_Events/News/Archived/2012/National_ Oilwell_Varco_Completes_Flexible_Pipe_Acquisition.aspx>. Acesso em: 16 fev. 2017.

NOCEDAL, J.; WRIGHT, S. **Numerical optimization**. 2. ed. Nova York: Springer, 2006.

OFFSHORE ENERGY TODAY. **Subsea 7 Scores $600 Mln in Petrobras PLSV Contracts**. 2013. Disponível em: < http://www.offshoreenergytoday.com/subsea-7-scores-600-mln-in-petrobras-plsv-contracts/>. Acesso em: 27 mar. 2017.

PEARL, J. **Heuristics:** intelligent search strategies for computer problem solving. Boston: Addison-Wesley, 1984. 382p.

PFUND, M.; FOWLER, J. W.; GUPTA, J. N. A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. **Journal of the Chinese Institute of Industrial Engineers**, v. 21, n. 3, p. 230-241, 2004.

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

QIN, T.; PENG, B.; BENLIC, U.; CHENG, T. C. E.; WANG, Y.; LÜ, Z. Iterated local search based on multi-type perturbation for single-machine earliness/tardiness scheduling. **Computers & Operations Research**, v. 61, p. 81-88, 2015.

QUEIROZ, M. M.; MENDES, A. B. Heuristic approach for solving a pipe layer fleet scheduling problem. In: RIZZUTO, E.; SOARES, C. G. (Eds.) **Sustainable maritime transportation and exploitation of sea resources.** Londres: Taylor & Francis Group, 2012. p. 1073-1080.

RAJKANTH, R.; RAJENDRAN, C.; ZIEGLER, H. Heuristics to minimize the completion time variance of jobs on a single machine and on identical parallel machines. **The International Journal of Advanced Manufacturing Technology**, v. 88, n. 5, p. 1-14, 2016.

REEVES, C. R. **Modern heuristic techniques for combinatorial problems**. Nova York: John Wiley & Sons, Inc., 1993.

RONEN, D. Cargo ships routing and scheduling: Survey of models and problems. **European Journal of Operational Research**, v. 12, n. 2, p. 119-126, 1983.

RONEN, D. Ships scheduling: The last decade. **European Journal of Operational Research**, v. 71, n. 3, p. 325-333, 1993.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033-2049, 2007.

SABOUNI, M. Y.; JOLAI, F.; MANSOURI, A. Heuristics for minimizing total completion time and maximum lateness on identical parallel machines with setup times. **Journal of Intelligent Manufacturing**, v. 21, n. 4, p. 439-449, 2010.

SANTOS, V. L. A.; ARROYO, J. E. C.; CARVALHO, T. F. Iterated local search based heuristic for scheduling jobs on unrelated parallel machines with machine deterioration effect. In: **Proceedings of the GECCO 2016 - Genetic and Evolutionary Computation Conference Companion.** Denver, Colorado, 2016. p. 53-54.

SERPA, F. G. **Modelo de programação matemática para suporte à decisão na compra e distribuição de dutos e umbilicais.** 2012. 73 f. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ. 2012.

SINAVAL. **Technip e DOF conquistam contrato para quatro navios PLSV**. 2013. Disponível em: <http://sinaval.org.br/2013/08/technip-e-dof-conquistam-contrato-para-quatro-navios-plsv/>. Acesso em: 27 mar. 2017.

SPEIGHT, J. G. **Handbook of offshore and gas operations**. Amsterdan: Elsevier, 2014.

SU, L. H. Scheduling on identical parallel machines to minimize total completion time with deadline and machine eligibility constraints. **The International Journal of Advanced Manufacturing Technology**, v. 40, n. 5-6, p. 572-581, 2009.

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

T'KINDT, V.; BILLAUT, J. C. **Multicriteria scheduling**: theory, models and algorithms. Nova York: Springer Science & Business Media, 2006.

UNCTAD – United Nations Conference on Trade and development. **World Investment Report 2015**: Reforming International Investment Governance. Nova York e Genebra: United Nations Publication. 2015.

VERGARA, S. C. **Projetos e relatórios de pesquisa em administração**. 9 ed. São Paulo: Atlas, 2007.

VERNALHA, F. C.; KADOUAKI, B. A.; MOURA, C. V.; CUNHA, V. C.; MENDES, A. B.; BRINATI, M. A. Programação de embarcações PLV em uma operação "offshore". In: **Anais do 22º Congresso Nacional De Transporte Aquaviário, Construção Naval e Offshore**, 2008. Rio de Janeiro. SOBENA, 2008.

VIEIRA, G. E.; HERRMANN, J. W.; LIN, E. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. **Journal of Scheduling**, v. 6, n. 1, p. 39-62, 2003.

WANG, X.; CHENG, T. C. E. A heuristic for scheduling jobs on two identical parallel machines with a machine availability constraint. **International Journal of Production Economics**, v. 161, p. 74-82, 2015.

WOLLNER, G.; ROSA, P. H. **Projeto de sistemas oceânicos II** - Pipe Laying Suport Vessel - PLSV JUBARTE II - Engenharia Naval e Oceânica - UFRJ. 2015. Disponível em: <http://www.oceanica.ufrj.br/deno/prod_academic/relatorios/2015/PedroR+GabrielW/relat1/Relat%201_comentado.htm> Acesso em: 14 fev. 2017.

WOLSEY, L. A. **Integer programming**. Nova York: Wiley, 1998. v. 42.

XU, D.; CHENG, Z.; YIN, Y.; LI, H. Makespan minimization for two parallel machines scheduling with a periodic availability constraint. **Computers & Operations Research**, v. 36, n. 6, p. 1809-1812, 2009.

XU, H.; LÜ, Z.; CHENG, T. C. E. Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness. **Journal of Scheduling**, v. 17, n. 3, p. 271-287, 2014.

YIN, Y.; CHENG, T. C. E.; WANG, D. J. Rescheduling on identical parallel machines with machine disruptions to minimize total completion time. **European Journal of Operational Research**, v. 252, n. 3, p. 737-749, 2016.

ZHAO, C.; JI, M.; TANG, H. Parallel-machine scheduling with an availability constraint. **Computers & Industrial Engineering**, v. 61, n. 3, p. 778-781, 2011.

APÊNDICE I – Programação de máquinas

A programação de recursos, seja em um ambiente de manufatura ou em serviços, tem um importante papel como ferramenta de apoio a decisão dentro das empresas, por lidar em geral com atividades que impactam diretamente nos objetivos propostos pela companhia e com recursos limitados para a execução destas atividades. Devido à maior abordagem se dar na indústria, o tema é tratado como "programação de máquinas", contudo, podendo ser aplicado aos mais variados tipos de recursos produtivos ou serviços existentes, como em: programação de centros de trabalho ou funcionários, alocações de aviões a portões de embarque, alocação de salas de reunião, agendamento de consultas, entre outros. De forma geral, o problema consiste em programar n atividades $(1 \leq j \leq n)$ em m máquinas $(1 \leq j \leq m)$ disponíveis, assumindo-se que ambos são finitos. As principais características de uma atividade a ser programada, são (Pinedo, 2008):

- Tempo de processamento / *Processing Time* (p_{ij}): Representa a duração de uma determinada atividade j em uma máquina i. Quando essa duração independe da máquina que está executando a atividade, o indice *i* é retirado, sendo o tempo de processamento definido como p_j.
- Data de liberação / *Release date* (r_j): Define o menor instante de tempo no qual uma atividade pode ser programada para iniciar.
- Data de compromisso / *Due date* (d_j): Data limite ou data desejada para o término de uma atividade.
- Peso ou prioridade / Weight (w_j): Indica a prioridade da atividade em relação às outras a serem programadas, ou seja, quanto maior seu peso, maior a sua importância relativa às demais.

Lustosa *et al*. (2008) apontam as principais decisões em um ambiente de programação de máquinas, com três dentre elas importantes para a conceituação do presente trabalho, que são:

- Designação / Assignment: A determinação da máquina ou máquinas onde cada atividade será executada.
- Sequenciamento / Sequencing: Definição da sequência de execução das atividades designadas a cada uma das máquinas.

- Programação / Scheduling: Determinação dos instantes em que cada atividade inicia e termina em cada máquina que a executará, baseando-se na designação e sequenciamento propostos.

O intuito da programação de máquinas é que sejam elaborados cronogramas de execução das atividades em um horizonte de tempo limitado, previamente estipulado, visando atender objetivos específicos (Cheng e Sin, 1990).

Graham *et al*. (1979) introduziram a notação de três campos $\alpha \mid \beta \mid \gamma$, a fim de classificar os problemas de programação, onde o primeiro campo (α) representa o ambiente em que se trabalha, ou seja, as características das máquinas e do problema a ser tratado, o segundo campo (β) indica as particularidades do problema e o último campo (γ) o critério ou função de avaliação a ser seguido.

Segundo Lustosa *et al*. (2008), Pinedo (2008) e Mokotoff (2001), é importante conhecer o ambiente relativo às máquinas, informado no campo (α) da notação de três campos, destacando os principais, que são:

- Máquina única (1): Ambiente composto por apenas uma máquina a ser programada, eliminando a etapa de designação, dado que a única máquina disponível deverá executar todas as atividades a processar. Esse ambiente é importante, pois em determinados casos, pretende-se estudar apenas o recurso gargalo em uma operação.
- Máquinas paralelas idênticas (P_m): Generalização do problema de máquinas únicas. Neste ambiente, máquinas idênticas são disponibilizadas em paralelo, adicionando ao problema a decisão de designação das atividades a cada uma das máquinas disponíveis, para que possam então ser sequenciadas. As atividades precisam ser realizadas apenas uma vez, e seu tempo de processamento p_j ($j = 1, \dots n$) independe da máquina que irá executá-la.
- Máquinas paralelas uniformes (Q_m): Diverge do problema com máquinas idênticas apenas pelo fato de cada máquina possuir uma velocidade de processamento v_i ($i = 1, \dots m$) pré-determinada, fazendo com que o tempo de execução de uma atividade j em uma máquina i, seja dado pela razão entre o tempo de processamento da atividade e a velocidade da máquina, $p_{ij} = p_j / v_{ij}$.

- Máquinas paralelas não-relacionadas (R_m): Mais uma variação do problema com máquinas paralelas idênticas, neste ambiente, o tempo de processamento de cada atividade é diferente de máquina para máquina, p_{ij} $(j = 1, \dots n; i = 1 \dots m)$, sem nenhuma relação de proporção entre as máquinas, como no caso com máquinas paralelas uniformes.
- Máquinas em série / *Flow shop* (F_m): Composto por *m* máquinas em paralelo, neste ambiente, as atividades devem passar por todas as máquinas de forma sequencial. Cada atividade possui um tempo de processamento específico em cada uma das máquinas.
- Máquinas em série - flexível / *Flexible flow shop* (FFc): Generalização dos modelos de máquinas em série, esse ambiente é composto por estágios de trabalho em série a programar, onde em cada estágio são disponibilizadas *m* máquinas em paralelo, de forma que, a atividade a ser executada em cada estágio deve ser realizada apenas em uma das máquinas deste estágio.
- Oficina de máquinas / *Job shop* (J_m): Nesta configuração, n atividades devem ser processadas em m máquinas paralelas, cada uma delas com uma rota a seguir, ou seja, cada atividade deve ser processada por um subconjunto de máquinas, em uma sequência específica para cada atividade, com tempos de processamento distintos em cada uma das máquinas.
- Oficina de máquinas - flexível / *Flexible Job shop* (FJc): Composto por centros de trabalho programados como uma oficina de máquinas, com m máquinas disponibilizadas em paralelo dentro de cada centro. Cada atividade é processada por apenas uma das máquinas pertencentes aos centros designados.
- *Open shop* (O_m): Ambiente formado por *m* máquinas, onde as atividades devem passar por todas as máquinas em qualquer ordem a ser definida pelo programador. Os tempos de execução de cada atividade são distintos em cada máquina.

As características das atividades ou particularidades do problema, apontadas no campo (β) da notação, podem estar presentes ou não em cada caso, sendo o campo indicado como vazio na hipótese da não existência, e com uma ou múltiplas entradas indicando as particularidades, caso existam (Pinedo, 2008).

As particularidades encontradas na literatura são inúmeras, se limitando o presente trabalho a expor apenas as que são mais comumente estudadas ou aquelas que possuem correspondência com o problema a ser tratado, detalhadas conforme descrição feita por Pinedo (2008):

- Preempção: (prpm): Implica que uma atividade pode ser interrompida após o seu início para terminar de ser processada em outra máquina no mesmo instante ou posteriormente em qualquer máquina capaz de realizá-la, disponibilizando o espaço para que outra atividade possa ser alocada.
- Precedência (prec): Indica que uma ou mais atividades devem terminar antes que outra atividade a elas relacionadas possa iniciar.
- *Setup* dependente da sequência (s_{jk}): Representa o tempo incorrido de *setup* ou preparação entre as atividades j e k. Sendo indicado como s_{ijk} para os casos em que existe também a dependência da máquina i na qual as atividades serão realizadas. O *setup* incorrido antes da primeira atividade e após a última, são denotados como s_{oj} e s_{jo} respectivamente.
- Elegibilidade da máquina (M_j): Aparece quando nem todas as máquinas são capazes de processar todas as atividades, indicando que para cada atividade j existe um subconjunto de máquinas M_j aptas a realizá-la.

Os tempos de *setup* podem ser também, independentes da sequência (s_j), não existindo assim qualquer relação com a máquina na qual a atividade é executada e nem com a sua predecessora imediata. Outra característica importante dos *setups* é que estes podem ser antecipatórios ou não antecipatórios. No caso de *setup* antecipatório, o mesmo pode ser realizado antes do instante de liberação da atividade, enquanto que para *setups* não antecipatórios, a sua execução só pode ser feita a partir do momento no qual a atividade encontra-se disponível para execução (Framinan *et al.,* 2014).

Outra característica importante dos problemas de programação é que os mesmos podem ser do tipo *off-line* ou *online*. Nos problemas *off-line* os dados de entrada do problema (tempos de processamento, datas de liberação, datas de compromisso entre outros) são todos conhecidos a priori, enquanto em modelos *online* a data de liberação só é conhecida no instante em que a atividade é de fato liberada e passa a existir para o programador e o seu tempo de processamento, somente quando a mesma é finalizada (Pinedo, 2008).

Ma *et al.* (2010) destacam a condição de disponibilidade das máquinas, em que se definem períodos nos quais as máquinas podem ter atividades a elas associadas para execução. Cada Janela S é composta uma janela inicial (B_i^S) e uma janela final (F_i^S), podendo se utilizar apenas B_i e F_i para o caso de existir apenas uma janela de disponibilidade.

Os critérios de avaliação a serem minimizados em geral são função dos tempos de conclusão das atividades ou das suas datas de compromisso. O tempo de conclusão (*completion time*) é indicado por C_j e marca o instante no qual uma atividade deixa o sistema, ou seja, quando sua execução termina (Pinedo, 2008).

No Quadro 3 são listadas as principais notações citadas nesse trabalho e utilizadas na literatura para a definição dos parâmetros dos problemas de programação.

Quadro 3 - Notações para os problemas de programação. Fonte: Elaboração própria.

n	Número de atividades
J	Conjunto de atividades
m	Número de máquinas
M	Conjunto de máquinas
j, k	Indice das atividades, $j, k = 1 \ldots n$
i	Indice das máquinas, $i = 1 \ldots m$
H	Horizonte de planejamento
t	Indice dos instantes de tempo (períodos), $t = 1 \ldots H$
p_j	Tempo de processamento da atividade j
r_j	Data de liberação da atividade j
d_j	Data de compromisso da atividade j
w_j	Peso ou prioridade da atividade j, indicando a importância dessa atividade com relação as demais
M_j	Elegibilidade das máquinas
s_{jk}	*Setup* dependente da sequência
C_j	Tempo de conclusão *(Completion Time)*

Para cada atividade j, com data de compromisso d_j especificada, alguns valores para essa atividade podem computados, partindo-se de uma programação σ estipulada (Chen *et al.,* 1998):

- *Completion time:* $C_j(\sigma)$
- *Flow time:* $F_j(\sigma) = C_j(\sigma) - r_j$

- *Lateness:* $L_j(\sigma) = C_j(\sigma) - d_j$
- *Earliness:* $E_j(\sigma) = \max\{d_j - C_j(\sigma), 0\}$
- *Tardiness:* $T_j(\sigma) = \max\{C_j(\sigma) - d_j, 0)$
- *Unit penalty:* $U_j(\sigma) = 1$ se $C_j(\sigma) > d_j$ e $U_j(\sigma) = 0$, caso contrário.

Podendo ser escritos como C_j, F_j, L_j, E_j, T_j e U_j, no caso de não existir ambiguidade sobre a programação a ser considerada.

Cheng e Sin (1990), French (1982) e Pinedo (2008), dividem os critérios de avaliação especificados no campo (γ), entre os que se baseiam nos tempos de conclusão das atividades e aqueles que se baseiam nas datas de compromisso.

Baseados nos tempos de conclusão:

- *Total completion time*: $\sum C_j$
- *Mean total completion time*: $\sum C_j / n$
- *Weighted completion time*: $\sum w_j C_j$
- *Makespan*: C_{max} (maior valor de C_j)
- *Total flowtime*: $\sum F_j$
- *Mean flowtime*: $\sum F_j / n$
- *Weighted flowtime*: $\sum w_j F_j$

Baseados nas datas de compromisso:

- *Total lateness*: $\sum L_j$
- *Mean lateness*: $\sum L_j / n$
- *Weighted lateness*: $\sum w_j L_j$
- *Maximum lateness*: L_{max} (maior valor de L_j)
- *Total tardiness*: $\sum T_j$
- *Mean tardiness*: $\sum T_j / n$
- *Weighted tardiness*: $\sum w_j T_j$
- *Maximum tardiness*: T_{max} (maior valor de T_j)
- *Number of tardy jobs*: $\sum U_j$
- *Weighted number of tardy jobs*: $\sum w_j U_j$

De acordo com Gokhale e Mathirajan (2012), existe um equilíbrio na proporção entre os estudos que tratam problemas tradicionais com objetivos

baseados em tempos de conclusão e problemas com objetivos baseados nas datas de compromisso das atividades.

Em um ambiente de manufatura é pouco comum que apenas um critério de avaliação seja utilizado, devido às necessidades concomitantes em atender a demanda de produtos cada vez com ciclos de vida menores, reduzindo custos com equipamentos e instalações. A utilização de multicritério na avaliação de uma programação aumenta ainda mais a complexidade de problemas que com um único objetivo já se caracterizam como de alta complexidade em seu tratamento (Pfund *et al.*, 2004; Nagar *et al.*, 1995; T´kindt e Billaut, 2006).

Em alguns casos, existe a necessidade de que as atividades sejam finalizadas exatamente na data de compromisso, similar a filosofia *just-in-time*, sendo então penalizados de forma simultânea, o atraso e a antecipação das atividades com relação a sua data de compromisso. O atraso é penalizado com base nos custos do não cumprimento da data prometida e a antecipação devido ao acúmulo de estoques que pode acarretar. Esses modelos penalizam o *Earliness* (E_j) e o *Tardiness* (T_j), atribuindo pesos α_j e β_j respectivamente a cada um dos critérios para cada atividade, sendo então o critério de avaliação de acordo com Lam e Xing (1997), definido como:

$$\sum(\alpha_j E_j + \beta_j T_j)$$

Para a resolução dos problemas clássicos, diversas heurísticas baseadas em regras de prioridade são utilizadas, devido à baixa complexidade computacional e a facilidade de aplicação destas. As que são utilizadas com maior frequência, de acordo com Kaabi e Harrath (2014) são:

- *Shortest Processing Time* (SPT): Com base nos tempos de processamento, as atividades são alocadas em ordem crescente, de forma que aquelas que possuem menor duração são programadas prioritariamente.
- *Preemptive Shortest Processing Time* (PSPT): Similar à regra SPT e aplicada a problemas nos quais a preempção é permitida, as atividades são ordenadas de forma crescente com base no tempo restante de processamento. Alocando prioritariamente aquelas com menor tempo faltante nos instantes nos quais uma das máquinas fica disponível.
- *Weighted Shortest Processing Time* (WSPT): Em existindo um peso ou prioridade a ser aplicada a cada atividade, a ordem de alocação se dá pela

resultante entre o tempo de processamento da atividade j em uma máquina i e o eu respectivo peso w_j, calculada por p_{ij}/w_j.

- *Earliest Due Date* (EDD): As atividades são programadas de acordo com a sua data de compromisso. Prioritariamente aquelas com data de compromisso menor.
- *Longest Processing Time* (LPT): As atividades são ordenadas decrescentemente de acordo com os tempos de processamento, onde aquelas com maiores tempos têm prioridade de programação.
- *Critical Path* (CP): Em existindo precedência entre as atividades, o caminho crítico é traçado e aquelas com maiores níveis, são programadas antes.
- *Largest Number of Successors* (LNS): Também para problemas com precedência, as atividades com maior número de sucessores no grafo de precedência têm prioridade.
- *Longest Remaining Processing Time first* (LRPT): Em existindo preempção, as atividades com maior tempo de processamento restante são programadas antes.
- *Longest Remaining Processng Time first-Fastest Machine* (LRPT-FM): Também para o caso em que a preempção é permitida e similar a regra anterior, atividades com maior tempo de processamento restante são programadas antes, priorizando as máquinas de maior velocidade de processamento.
- *Shortest Remaining Processing Time* (SRPT): Para problemas com preempção, de acordo com o menor tempo restante de processamento, as atividades são programadas em qualquer instante de tempo.
- *List Scheduling* (LS): As atividades são assinaladas às máquinas de acordo com uma sequência pré-definida, onde cada atividade é assinalada à máquina na qual seu término se dará o mais cedo possível.

Dentre os ambientes de máquinas expostos, a programação de PLSVs conforme abordada neste trabalho, se equivale a um problema de programação de máquinas paralelas, com os navios representando as máquinas que devem ser programadas. Esse assunto bastante estudado na literatura, com importantes trabalhos de revisão acerca do tema publicados. Cheng e Sin (1990) apresentam uma revisão completa, com mais de 80 artigos sobre o assunto analisados; Lam e

Xing (1997) direcionam a pesquisa para problemas com critérios de avaliação diferentes dos regulares, como foco em modelos com *just-in-time* entre outros; Mokotoff (2001) foca nos problemas de minimização do *makespan* (C_{max}) em máquinas paralelas idênticas; Pfund *et al*. (2004) direcionam o estudo para os problemas com máquinas paralelas não-relacionadas; Li e Yang (2009) revisam os trabalhos feitos com máquinas não idênticas com foco na minimização do *total completion time* ($\sum C_j$), apresentando modelos e métodos de resolução; Edis *et al*. (2013) analisam o estado da arte para os problemas com recursos adicionais, nos quais se faz necessário não só a consideração das máquinas a programar, mas também os diversos recursos utilizados na manufatura, como: *pallets*, veículos, ferramentas, moldes, entre outros; Behera (2012) analisa os problemas de programação em máquinas paralelas quanto a sua complexidade, avaliando os métodos de resolução aplicados na área; Kaabi e Harrath (2014) concentra o estudo em problema com restrições de disponibilidade das máquinas, onde as máquinas estão sujeitas a perdas de tempo por quebras, manutenção preventiva, disponibilidade de mão-de-obra, entre outros fatores.

APÊNDICE II – Cálculo do excedente navio mínimo da instância base

Após a definição da instância base, diversos testes foram feitos a partir do conhecimento tácito adquirido durante o mapeamento do processo feito junto à companhia. Inúmeras soluções foram desenvolvidas, ainda de forma manual, com base em algumas proposições seguidas pelos programadores, onde evidenciou-se a viabilidade na obtenção de soluções para essa instância sem atraso nos poços críticos e sem ociosidade dos navios. Contudo, não foi possível encontrar nenhuma solução que mantivesse esses componentes zerados e que, ao mesmo tempo, não possuísse excedente de navio.

Em seguida, durante testes com o algoritmo, não foi possível encontrar nenhuma solução com excedente navio menor do que dois dias, resultando no valor de função objetivo de 0,7 de acordo com os coeficientes propostos. Assim sendo, uma análise mais aprofundada baseada nas combinações de atividades para os navios que estavam com excedente foi realizada. O excedente apurado é referente as embarcações 2 e 10 pertencentes as famílias 4 e 3, respectivamente, com um dia de excedente para cada família.

A família 3 é composta apenas pela embarcação 10, facilitando a comprovação do atraso mínimo de um dia apurado. Em ambas as avaliações, considera-se que todas as atividades possuem data de liberação em $t = 0$ e que as datas de compromisso de todos os poços críticos se dão em $t = \infty$. Dessa forma, essas restrições são desconsideradas do problema para que seja avaliada apenas a relação entre as janelas disponíveis dos navios e a duração de todas as atividades a serem executadas. Outro aspecto a ser considerado nas avaliações a seguir é o fato de que nenhum navio dessas famílias está apto a receber atividades do tipo *manifold* e nem possuem atividades de parada programada. Sendo assim, os tempos de *setup* de todas as viagens tem duração de seis dias.

A janela disponível do navio 10 é de 82 dias, correspondida entre $t = 107$ e $t = 188$. O conjunto de atividades programadas na família 3, da qual a embarcação faz parte, é exposto na Tabela 8, onde, são especificados as durações e o percentual de ocupação de cada atividade.

Tabela 8 - Atividades programadas na família 3. Fonte: Elaboração própria.

Atividade	Duração	Ocupação
312	15	25%
313	15	25%
331	11	25%
323	13	25%
578	15	50%
Total	**69**	**150%**

O somatório das ocupações das atividades em 150%, conforme mostrado na Tabela 8, aponta a necessidade mínima de duas viagens para a execução de todas as atividades listadas, dado o limite de ocupação em 100% para cada viagem. Desse modo, dois tempos de *setup* e duas navegações de retorno devem ser consideradas. Como a duração de um *setup* é de seis dias e o retorno é de um dia, 14 dias devem ser acrescidos ao tempo total de execução das atividades, que é de 69 dias, totalizando 83 dias a serem alocados ao navio.

Considerando a janela de 82 dias dessa embarcação, fica evidente a impossibilidade de execução das viagens com seus 83 dias demandados, sem que exista pelo menos um dia de excedente navio a ser contabilizado para a família em questão.

Para a família 4, a análise é um pouco mais complexa devido sua composição com as embarcações 1 e 2, sendo necessária uma avaliação a partir das combinações possíveis entre as atividades para dar origem as viagens. O conjunto de atividades programadas na família 4 é exibido na Tabela 9.

Tabela 9 - Atividades programadas na família 4. Fonte: Elaboração própria.

Atividade	Duração	Ocupação
356	11	50%
109	13	50%
354	21	50%
285	22	50%
357	15	25%
110	7	25%
111	6	25%
355	14	25%
Total	**109**	**300%**

A janela da embarcação 1 é de 59 dias, contida entre $t = 129$ e $t = 187$ e a da embarcação 2 é de 71 dias, compreendida entre $t = 123$ e $t = 193$. O somatório

das ocupações em 300%, indica uma necessidade mínima de três viagens para a execução dessas atividades, cada uma delas com a ocupação em 100%. Se forem executadas as três viagens mínimas, o tempo total incorrido com *setups* e retornos é de 21 dias. Somando-se a esse tempo a duração das atividades, o tempo total a ser alocado aos navios é de 130 dias. O mesmo valor é encontrado quando somadas as janelas das embarcações (59 e 71 dias), indicando que uma solução hipotética sem excedente navio só seria possível caso as atividades fossem distribuídas em três viagens exatamente. Qualquer solução com um maior número de viagens representaria em excedente de navio de no mínimo sete dias (tempo de um *setup* e uma navegação de retorno).

Para avaliar a existência de uma composição de viagens que de fato não gerem excedente de navio, todas as combinações possíveis entre as atividades foram realizadas, considerando apenas viagens com ocupação completa (100%) e desconsiderando a ordem das atividades dentro das viagens, dado que as restrições que envolvem o cumprimento das datas foram desconsideradas nesta avaliação. A combinação resultou em 31 viagens v possíveis, listadas na Tabela 10, com as atividades contidas em cada uma e sua respectiva duração total destacadas, considerando os tempos de *setup* e retorno em sua duração.

Tabela 10 - Listagem de viagens possíveis para a família 4. Fonte: Elaboração própria.

v	Atividades				Duração
1	356	109			**31**
2	356	354			**39**
3	356	285			**40**
4	109	354			**41**
5	109	285			**42**
6	354	285			**50**
7	356	357	110		**40**
8	356	357	111		**39**
9	356	357	355		**47**
10	356	110	111		**31**
11	356	110	355		**39**
12	356	111	355		**38**
13	109	357	110		**42**
14	109	357	111		**41**
15	109	357	355		**49**
16	109	110	111		**33**
17	109	110	355		**41**
18	109	111	355		**40**
19	354	357	110		**50**
20	354	357	111		**49**
21	354	357	355		**57**
22	354	110	111		**41**
23	354	110	355		**49**
24	354	111	355		**48**
25	285	357	110		**51**
26	285	357	111		**50**
27	285	357	355		**58**
28	285	110	111		**42**
29	285	110	355		**50**
30	285	111	355		**49**
31	357	110	111	355	**49**

Para que as três viagens necessárias sejam executadas nos dois navios disponíveis, um dos navios deve executar duas viagens enquanto o outro navio executa a viagem remanescente. Dito isso, o primeiro caso avaliado foi com a embarcação 1 executando 2 viagens. Devido a sua janela limitada, procurou-se encontrar duas viagens com menores durações e sem repetição entre atividades. Essa viagens são $v = 1$ e $v = 22$, com durações de 31 e 41 respectivamente, resultando em 72 dias totais a alocar ao navio. Como a janela disponível da embarcação 1 é de 59 dias, isso resultaria em um excedente de navio mínimo de 13 dias. A outra hipótese seria o navio 1 executar apenas uma viagem. Nesse caso, o ideal é encontrar uma viagem que ocupe o máximo possível da embarcação, minimizando o tempo total das viagens a serem executadas pela embarcação 2. A maior duração encontrada foi para $v = 27$ com 58 dias. Considerando o montante de tempo total a alocar de 130 dias, referente a todas as atividades, *setups* e retornos, conforme destacado anteriormente. O tempo remanescente a ser alocado na embarcação 2 é dado pela diferença desse montante com o que foi alocado no navio 1, resultando em 72 dias a serem executados pela embarcação 2. Considerando sua janela de 71 dias, o excedente mínimo para essa embarcação e consequentemente sua família é de um dia.