UNIVERSIDADE FEDERAL DE MINAS GERAIS

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Letícia Diniz da Cruz

CONSTRUÇÃO DE ALGORITMOS DE OTIMIZAÇÃO PARA O PLANEJAMENTO DO DESLOCAMENTO DE BOBINAS EM UMA FÁBRICA DE TUBOS FLEXÍVEIS

Belo Horizonte

2024

Letícia Diniz da Cruz

CONSTRUÇÃO DE ALGORITMOS DE OTIMIZAÇÃO PARA O PLANEJAMENTO DO DESLOCAMENTO DE BOBINAS EM UMA FÁBRICA DE TUBOS FLEXÍVEIS

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutora em Engenharia Elétrica.

Orientador: Ricardo Hiroshi Caldeira Takahashi

Coorientador: Eduardo Gontijo Carrano

Belo Horizonte

2024

Cruz, Letícia Diniz da.
C957c Construção de algoritmos de otimização para o planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis [recuso eletrônico] / Letícia Diniz da Cruz. - 2024.
1 recurso online (131 f. : il., color.) : pdf.

Orientador: Ricardo Hiroshi Caldeira Takahashi.
Coorientador: Eduardo Gontijo Carrano.

.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 109-131.

Bibliografia: f. 106-108.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Heurística - Teses 3. Otimização combinatória - Teses. 4. Planejamento da produção - Teses. 5. Programação linear - Teses. I. Takahashi, Ricardo Hiroshi Caldeira. II. Carrano, Eduardo Gontijo. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo César Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

"Construção de Algoritmos de Otimização Para Planejamento do Deslocamento de Bobinas em Uma Fábrica de Tubos Flexíveis"

Leticia Diniz da Cruz

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica. Aprovada em 05 de junho de 2024. Por:

Prof. Dr. Ricardo Hiroshi Caldeira Takahashi
DMAT (UFMG) - Orientador

Prof. Dr. Eduardo Gontijo Carrano
DEE (UFMG) - Coorientador

Prof. Dr. Luiz Satoru Ochi
Instituto de Computação (UFF)

Prof. Dr. Marcone Jamilson Freitas Souza
Departamento de Computação (UFOP)

Prof. Dr. Martín Gómez Ravetti
DCC (UFMG)

Prof. Dr. Lucas de Souza Batista
DEE (UFMG)

Documento assinado eletronicamente por **Ricardo Hiroshi Caldeira Takahashi**, **Professor do Magistério Superior**, em 05/06/2024, às 23:18, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Lucas de Souza Batista**, **Professor do Magistério Superior**, em 06/06/2024, às 15:54, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Eduardo Gontijo Carrano**, **Professor do Magistério Superior**, em 06/06/2024, às 22:30, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcone Jamilson Freitas Souza**, **Usuário Externo**, em 07/06/2024, às 13:40, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Satoru Ochi**, **Usuário Externo**, em 07/06/2024, às 14:20, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Martin Gomez Ravetti**, **Professor(a)**, em 07/06/2024, às 16:15, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3280832** e o código CRC **5297C485**.

Referência: Processo nº 23072.229901/2024-19 SEI nº 3280832

Resumo

Em uma fábrica de tubos flexíveis, em função de uma série de restrições relacionadas às datas de entrega dos produtos, alocação das máquinas, limitações físicas dos equipamentos e espaço disponível, é necessário planejar como os tubos em produção devem se deslocar entre as máquinas responsáveis pela construção de cada camada dos tubos flexíveis. As camadas são definidas de acordo com a especificação dos clientes e por essa razão não existe uma única linha de produção contínua. Para garantir a eficiência operacional, é necessário que todas as etapas do processo produtivo sejam executadas com atraso mínimo, adiantamento máximo e redução do uso de recursos. O deslocamento dos tubos flexíveis é realizado a partir da movimentação de bobinas que os contém e, por essa razão, neste trabalho são propostas duas estratégias de otimização para planejamento do deslocamento de bobinas. A primeira é voltada ao planejamento diário da fábrica, com uso de algoritmos baseados em meta-heurísticas e modelos matemáticos e com restrições de tempo de execução menos rígidas. A segunda é destinada ao replanejamento em tempo real após imprevistos e foi concebida a partir de heurísticas de busca local que devem ser capazes de encontrar soluções de boa qualidade em baixo tempo computacional. Foram desenvolvidas múltiplas heurísticas para cada estratégia, mas todas as heurísticas, tanto para planejamento quanto para replanejamento, consideram as mesmas restrições, funções objetivo e horizonte de planejamento, se diferenciando apenas pelo tempo de execução disponível. Os algoritmos propostos foram testados em instâncias reais de uma planta industrial atualmente em operação e são capazes de encontrar resultados de acordo com a expectativa da fábrica em até quatro horas para planejamento e em até cinco minutos para replanejamento. Entre as heurísticas propostas, a que alcançou melhor resultado para o problema de planejamento foi desenvolvida a partir da meta-heurística VNS e para o replanejamento, uma heurística de busca local determinística.

Palavras-chave: busca local; heurísticas; otimização combinatória; planejamento da produção; programação linear inteira.

Abstract

In a flexible pipe factory, due to a series of restrictions related to product delivery dates, machine allocation, physical limitations of equipment and available space, it is necessary to plan how pipes in production should move between the machines responsible for construction of each layer of flexible pipes. The layers are defined according to customer specifications and for this reason there is no single continuous production line. To ensure operational efficiency, it is necessary that all stages of the production process are carried out with minimum delay, maximum advance and reduced use of resources. The displacement of flexible pipes is carried out by moving the reels that contain them and, for this reason, in this work two optimization strategies are proposed for planning the movement of reels. The first is aimed at daily factory planning, using algorithms based on meta-heuristics and mathematical models and with less rigid execution time restrictions. The second is intended for real-time replanning after unforeseen events and was designed based on local search heuristics that should be able to find good quality solutions in low computational time. Multiple heuristics were developed for each strategy, but all heuristics, both for planning and replanning, consider the same restrictions, objective functions and planning horizon, differing only by the available execution time. The proposed algorithms were tested in real instances of an industrial plant currently in operation and are capable of finding results in accordance with the factory's expectations in up to four hours for planning and in up to five minutes for replanning. Among the proposed heuristics, the one that achieved the best results for the planning problem was developed from the VNS meta-heuristic and for replanning, a deterministic local search heuristic.

Keywords: local search; heuristics; combinatorial optimization; production planning; integer linear programming.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

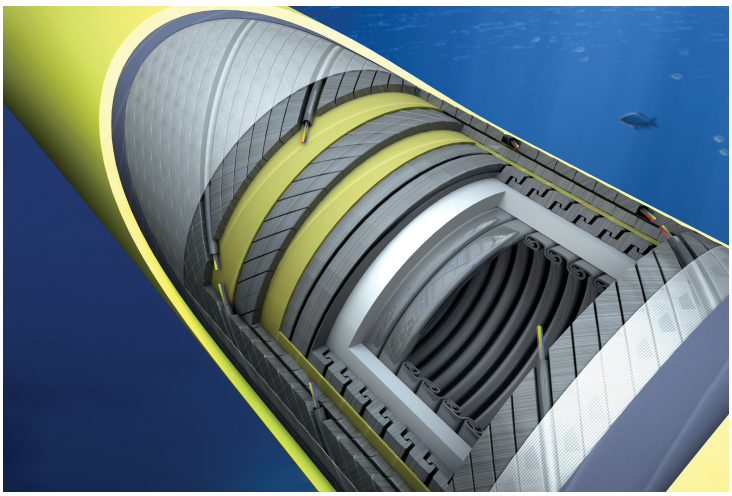
1.1 Motivação

Soluções tecnológicas sofisticadas permitem levar o petróleo e o gás natural do fundo do mar até à superfície. Entre essas soluções, os tubos flexíveis são empregados em todo sistema submarino de coleta e escoamento, conduzindo os fluidos produzidos pelo poço para as unidades de produção. Eles podem ser utilizados para a interligação de uma unidade a outra, injeção ou o descarte de fluidos em reservatórios ou exportação da produção para terra. Seu formato é tubular, com diferentes camadas de materiais metálicos e não metálicos, cada qual com uma função específica, conforme a aplicação. Uma representação gráfica dessas camadas pode ser visualizada na Figura 1.

Para confeccionar tubos flexíveis e garantir a eficiência operacional, a produção é planejada em diferentes setores e momentos, desde uma visão macro sobre quando é possível atender cada cliente, até como deve ser a operação no chão de fábrica. Primeiramente são definidas as datas limite de entrega de cada produto para um grande horizonte de tempo. Em seguida, para períodos de tempo mais curtos e próximos do momento atual, as ordens de produção são sequenciadas e agendadas. Neste ponto determina-se quando cada máquina estará reservada para cada tubo flexível a ser produzido. Durante a construção das camadas ou armazenamento, os tubos flexíveis são mantidos enrolados em bobinas. Por essa razão, em um terceiro passo, de acordo com as especificações técnicas do tubo ou com a etapa de produção, são definidas quais bobinas devem transportar os tubos entre as diferentes máquinas ou armazená-los até a entrega ao cliente. O transporte via bobinas é realizado majoritariamente com pontes rolantes e como esses recursos e o espaço físico do galpão são limitados, é necessário planejar, em um último passo, a trajetória das bobinas a fim de evitar atrasos e melhorar os resultados da produção.

Como a produção de tubos flexíveis é realizada sob demanda, existem características e requisitos específicos para cada produto construído, como quantidade e composição das camadas. Essas especificidades adicionam uma complexidade extra ao planejamento da produção, uma vez que não existe uma única linha de operação, com planos padrão. As camadas dos tubos flexíveis podem ser construídas com diferentes materiais ou técnicas, levando ao uso de diferentes máquinas em cada caso e também gerando a necessidade de escolher bobinas específicas para deslocar e armazenar estes produtos. Apesar das especificidades, no geral os tubos flexíveis são construídos com grande comprimento e, por essa razão, precisam ser armazenados em bobinas que comportem essa dimensão. É possível visualizar a escala de dimensão em comparação à um ser humano na Figura 2.

Figura 1 – Gráfico representando as várias camadas de um tubo flexível



Fonte: *www.bakerhughes.com*

Figura 2 – Visão de um tubo flexível e um ser humano.

Fonte: *www.bakerhughes.com*

Como dito anteriormente, o plano de produção dos tubos flexíveis é composto por algumas etapas principais, planejadas por diferentes setores. Na empresa que representa o estudo de caso deste trabalho, os passos de definição da data de entrega, sequenciamento/agendamento da construção das camadas e escolha das bobinas são executados e finalizados antes da construção do plano de movimentação das bobinas. Por essa razão, o plano de movimentação deve conter instruções sobre como e quando movimentar as bobinas na fábrica, considerando as decisões tomadas anteriormente como objetivos ou restrições. Vale ressaltar que existe uma hierarquia para a tomada de decisão entre as diferentes etapas de elaboração do plano de produção e durante a geração do plano de movimentação das bobinas a definição entre diferentes possibilidades de planejamento são baseadas apenas nas informações disponíveis nesta etapa.

O principal objetivo durante o planejamento do deslocamento das bobinas é garantir a confecção de todas as camadas dos tubos com atraso mínimo conforme os prazos estipulados nas etapas anteriores, sem diferença de prioridade entre clientes. Em acréscimo, deseja-se maximizar o adiantamento para que possíveis intervalos de tempo ociosos sejam aproveitados por tarefas que seriam executadas nas próximas horas ou dias. Em geral a maximização do adiantamento para a construção de uma nova camada não gera custos extras de armazenamento uma vez que os tubos em produção já estão armazenados na fábrica. Por fim, para obter a melhor eficiência operacional, é necessário minimizar a quantidade de movimentos a fim de diminuir o desgaste dos equipamentos ao longo do tempo.

Esses objetivos são restringidos pela limitação do espaço coberto. Essa é uma limitação importante a ser considerada, uma vez que a exposição a condições climáticas como chuva e sol pode afetar a integridade dos tubos em produção e, portanto, deve ser evitada. No espaço disponível, as bobinas são içadas e deslocadas por pontes rolantes, operadas manualmente, que podem realizar apenas alguns movimentos específicos, adicionando ainda mais complexidade ao planejamento.

Devido à limitação de altura do galpão, uma ponte rolante não é capaz, por exemplo, de içar e deslocar uma bobina por cima de outra e a construção das rotas de movimentação das bobinas pode demandar a retirada de outras bobinas que estejam no trajeto. Em acréscimo, algumas posições para armazenamento de bobinas também são utilizadas e bloqueadas durante a operação das máquinas. Um plano de movimentação mal elaborado pode levar a um grande atraso ou a um *deadlock* na planta, impedindo o início de outras atividades até que alguma máquina termine a operação e libere a posição para movimentação.

As posições de destino das bobinas deslocadas para liberar a rota de uma determinada atividade podem dificultar ou facilitar o trajeto da próxima tarefa. Por essa razão os movimentos de retirada devem ser elaborados de maneira que as próximas atividades sejam facilitadas, o que significa que todos os movimentos têm grande importância para evitar *deadlocks* e reduzir atrasos. Assim, deve existir uma estratégia global para elaborar o plano de movimentação, considerando todas as atividades simultaneamente, em vez de executar as tarefas sequencialmente, sem

considerar as demais.

Ainda que um plano inicial seja elaborado diariamente, cobrindo todos os movimentos daquele dia e alguns subsequentes, a planta está sujeita a imprevistos que podem inviabilizar este planejamento, como atrasos de início ou término de tarefas, paradas das máquinas, necessidade de reprocessamento dos tubos, dentre outros. É importante que o replanejamento das rotas possa ser feito rapidamente, a fim de minimizar os impactos na operação, mas o horizonte de planejamento deve considerar desde o instante em que ocorreu o imprevisto até o fim do período considerado na solução inicial. A simples concatenação de um trecho de solução replanejado com o restante da solução inicialmente planejada, provavelmente levaria a um resultado infactível ou, em um contexto otimista, distante do ótimo.

O planejamento e o replanejamento da movimentação de bobinas devem ser baseados em quatro objetivos principais: factibilidade, minimização do atraso, maximização do adiantamento e minimização da quantidade de movimentos. Como o replanejamento deve fornecer uma solução mais rapidamente, é possível que a qualidade da solução seja inferior à do planejamento. Essa diferença na qualidade da solução pode acontecer pois o problema abordado é combinatório e indexado no tempo. Não apenas a sequência de movimentos é importante, mas também o seu agendamento e a posição das bobinas em cada instante de tempo, garantindo que a construção das rotas seja factível e que os objetivos temporais sejam atendidos. Como o plano de produção é composto de algumas dezenas de bobinas, posições e tubos a serem manufaturados, a dimensão do problema inviabiliza uma abordagem exata, mas, em contrapartida, não é possível garantir a solução ótima com o uso de heurísticas e pode ser necessário mais tempo para explorar o espaço de busca de forma mais ampla.

Dada a diferença no tempo de execução para os algoritmos de planejamento e replanejamento, este trabalho propõe o desenvolvimento de duas estratégias de otimização. Ambas recebem como informação inicial a condição atual da fábrica (posicionamento das bobinas e status das máquinas), o agendamento da confecção de cada camada dos tubos flexíveis em cada máquina e quais bobinas devem transportar ou armazenar os tubos. As duas estratégias também consideram o mesmo horizonte de produção, restrições de movimentação das bobinas e objetivos. Dado o menor rigor de tempo de execução para o algoritmo de planejamento, utilizado diariamente para construção de um plano inicial de movimentação, as estratégias propostas almejam encontrar soluções eficientes utilizando meta-heurísticas e modelos matemáticos. O replanejamento, em contrapartida, é baseado em uma heurística de busca local, com menos alternativas para explorar o espaço de soluções, a fim de gerar novas rotas em poucos minutos em caso de imprevistos e reduzir os impactos na operação.

1.2 Objetivos

O principal objetivo desde trabalho é encontrar soluções factíveis e no tempo estipulado para o planejamento do deslocamento de bobinas em uma planta de produção de tubos flexíveis tanto para a versão de planejamento quanto na de replanejamento. É desejável que as soluções encontradas, além de factíveis, tenham boa qualidade a fim de garantir a eficiência operacional da fábrica. Por essa razão, são propostas duas estratégias de otimização, uma para cada versão do problema:

- A primeira estratégia é voltada ao planejamento diário das movimentações. Ela será executada uma vez por dia, considerando não só as tarefas previstas para aquele dia mas também tarefas previstas para os próximos cinco a dez dias, resultando em um planejamento de "longo prazo". A operação é beneficiada com a agilidade do algoritmo, que diminui a necessidade de replanejamento durante o planejamento, mas foi definido que essa estratégia tenha alguma flexibilidade para o tempo de execução, podendo demorar, no máximo, quatro horas para retornar sua solução. Logo, são utilizadas ferramentas de otimização que potencialmente exploram mais o espaço de soluções, mas que demandam um considerável tempo computacional.

- A segunda estratégia tem por intuito ser capaz de realizar o replanejamento da movimentação em tempo real após a ocorrência de algum imprevisto que inviabilize o planejamento inicial. Ela deve ser baseada essencialmente em métodos heurísticos de busca local e deve ser capaz de encontrar uma solução razoável em tempo curto (até cinco minutos). O atendimento deste limite de tempo é importante pois, durante a execução do algoritmo, não é possível permitir a movimentação de bobinas na planta, uma vez que estas movimentações podem tornar o novo plano gerado pela heurística infactível. É importante ressaltar que essa estratégia deve considerar o mesmo horizonte de produção que a primeira, para garantir uma solução eficiente em todo o horizonte e potencialmente facilitar as rotas dos próximos dias. Ela também deve gerar uma solução sem reaproveitar o planejamento anterior, uma vez que podem ser grandes as modificações na solução do planejamento inicial para atender ao imprevisto e tornar a solução factível e eficiente novamente.

1.3 Contribuições

A formulação do problema apresentado é, per si, uma contribuição original deste trabalho, que define e formaliza o planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis. Durante as buscas por referências, a autora não encontrou na literatura nenhum trabalho que lide com o mesmo problema ou com um problema suficientemente parecido a ponto de permitir uma comparação direta entre soluções e/ou algoritmos.

Tendo como base o ineditismo do problema, é possível conjecturar que os algoritmos de otimização desenvolvidos também sejam originais. Apesar de se enquadrarem em paradigmas já consolidados, como heurísticas de busca local e meta-heurísticas, os algoritmos foram construídos com base nas características específicas do problema e utilizam métodos originais para explorar o espaço de soluções. Como os algoritmos e métodos foram testados em instâncias reais de uma planta industrial atualmente em operação, é ampliada a relevância prática deste trabalho. A descrição do problema de movimentação de bobinas, as heurísticas desenvolvidas, assim como os resultados alcançados, além de apresentados neste trabalho também foram publicados em dois artigos [27] [26].

Por fim, é importante destacar que a otimização do planejamento de rotas para o deslocamento de bobinas se encaixa no contexto de indústria 4.0, que é um conceito muito popular aplicado na digitalização e automatização das indústrias em geral. Em vez da tomada de decisão manual para construção das rotas, susceptíveis a erros humanos, é possível obter as condições atuais da fábrica, gerar planos de produção otimizados e fornecer os resultados em tempo real e de forma automática, caso os algoritmos propostos sejam integrados a outros softwares da fábrica.

1.4 Estrutura do texto

Este documento tem sete capítulos, incluindo esta introdução que apresenta as motivações do trabalho, objetivos específicos e as contribuições esperadas.

Apesar de não ter encontrado referências prévias para o problema de mover e posicionar bobinas em uma fábrica de tubos flexíveis, o segundo capítulo apresenta referências bibliográficas de problemas com uma ou mais características coincidentes, como a movimentação de blocos ou *containers* com restrições de armazenamento ou deslocamento.

O terceiro capítulo descreve e exemplifica a configuração da fábrica deste estudo de caso e como deve ser elaborado o plano de movimentação das bobinas para os problemas de planejamento e replanejamento. Também são definidos alguns termos muito utilizados e importantes para a compreensão deste trabalho como *submovimento*, *movimento* e *operação*, além das restrições e das funções objetivo do problema.

O plano de movimentação determina como as bobinas são deslocadas na fábrica de uma maneira eficiente. Como discutido no quarto capítulo, esse deslocamento eficiente é baseado na definição de distância entre as posições, definição esta que permite construir caminhos mínimos. Esse conceito é utilizado pelas heurísticas para planejamento VNS–Heur (*Variable Neighborhood Search Heuristic*) e ILP–Heur (*Integer Linear Programming Heuristic*) e pela heurística para replanejamento LS–Heur (*Local Search Heuristic*) especificadas no quinto capítulo.

Os resultados encontrados por todas as estratégias propostas no quinto capítulo são

descritos no sexto capítulo, após teste em 13 instâncias reais de uma fábrica de tubos flexíveis. O capítulo final sintetiza o trabalho com uma análise crítica dos resultados.

2 Revisão Bibliográfica

Na revisão bibliográfica realizada para este trabalho, a autora não encontrou referências prévias para o problema específico de mover e posicionar bobinas em uma fábrica de tubos flexíveis. Entretanto, apesar deste problema específico possivelmente não ter sido documentado ainda, ele tem conexões com alguns estudos que incluem a movimentação de bobinas em fábricas/depósitos de bobinas de aço, de placas de aço ou de *containers* em terminais portuários. Esses estudos pertencem a classes conhecidas de problemas de pesquisa operacional, que compartilham uma ou mais das seguintes características:

- Movimentos de objetos em fábricas, onde esses objetos são armazenados temporariamente e, em algum momento, são retirados.
- Restrições em posições de armazenamento e no espaço para movimentação de objetos dentro da fábrica. Por essa razão os objetos posicionados dentro da instalação são realocados para permitir o deslocamento de outros objetos.
- Unidades de movimentação (exemplo: pontes rolantes, guindastes) como recursos limitados, cujo uso deve ser programado a fim de melhorar o desempenho da fábrica.

A referência [22] aborda o problema NP-difícil de agendamento de ponte rolante em um almoxarifado de bobinas de aço onde a ponte rolante é responsável por armazenar, retirar e reorganizar as bobinas. É necessário determinar a sequência dessas ações, assim como as posições destino de cada bobina durante a reorganização. São definidas estratégias de redução de variáveis para acelerar a solução do modelo e a solução ótima do problema reduzido é obtida a partir de um *solver MIP* para pequenas e médias instâncias. Para instâncias grandes é definida uma abordagem exata e outra aproximada, ambas com programação dinâmica. Essa referência é semelhante à fábrica de tubos flexíveis, uma vez que dada uma necessidade de movimentação das bobinas com uso de ponte rolante (seja para deslocar até uma posição de máquina, na fábrica de tubos flexíveis, seja para movimentar até um caminhão, no almoxarifado de bobinas de aço), caso não exista um caminho direto, é necessário planejar movimentos de retirada que facilitem os deslocamentos futuros e que reduzam o atraso de conclusão da tarefa corrente. Apesar de serem bobinas diferentes nos dois trabalhos, com regras de negócio diferentes (por exemplo, na fábrica de tubos flexíveis não é possível armazenar uma bobina por cima de outra e existe mais de uma ponte rolante), é possível assumir que este trabalho de movimentação das bobinas em uma fábrica de tubos flexíveis também é NP-difícil.

O estudo [21] retrata o mesmo problema da referência anterior, mas propõe um algoritmo genético (AG) como meio para solução. O trabalho [16] também apresenta o problema de

agendamento de ponte rolante, mas em um cenário onde existem duas pontes rolantes que são submetidas a restrições de não interferência. Nesse estudo não são consideradas nem a entrada de bobinas no depósito, nem a operação de reorganizá-las. Assim como na referência anterior, o problema é tratado com algoritmos genéticos. Apesar da fábrica de tubos flexíveis também considerar mais de uma ponte rolante, nela não é necessário definir restrições de não interferência entre as unidades de movimentação, uma vez que elas operam em espaços separados, de forma totalmente independente. Em acréscimo, esse trabalho é diferente por não acompanhar temporalmente o posicionamento das bobinas.

Um estudo sobre o problema decorrente do planejamento de armazenamento de placas de aço na produção de aço integrado é apresentado em [12]. Esse estudo otimiza a ordem de armazenamento das placas de aço em pilhas. Existem restrições que limitam o espaço físico para mover as placas, o número de unidades de movimentação disponíveis e as janelas de tempo em que as operações devem ser realizadas. Devido a diferença entre a ordem de produção e a de empilhamento, existem pilhas auxiliares onde as placas podem ser posicionadas temporariamente para permitir o reordenamento das placas conforme a ordem de produção. É apresentado que até uma versão simples do problema é PSPACE-completo. Por essa razão, é proposta uma heurística baseada em busca em grafos. Algumas diferenças relevantes desse problema em relação ao que é considerado neste trabalho são: as unidades de movimentação podem seguir rotas flexíveis enquanto as pontes rolantes são movimentadas em caminhos específicos; é possível carregar mais de uma placa simultaneamente mas apenas uma bobina por vez; as operações são sempre realizadas por um único meio de transporte enquanto na fábrica de tubos flexíveis existe mais de uma unidade de movimentação.

As referências [24, 25] estudam o problema de mover *containers* em terminais portuários. Esses artigos propõem um processo para pré-posicionar os *containers* em pilhas que são ordenadas de forma que os itens no topo tem maior probabilidade de serem entregues em relação aos itens na parte inferior. Esse pré-posicionamento é executado utilizando o tempo ocioso dos guindastes. No primeiro estudo é proposta uma heurística de busca local, e no segundo são definidas duas heurísticas, uma customizada com menor tempo de execução e outra baseada no algoritmo *branch-and-bound*, mais lenta que a primeira, mas com resultados melhores. O algoritmo *branch-and-bound* também é aplicado diretamente mas o tempo de execução não é factível para instâncias grandes. A ideia geral de executar o reposicionamento dos blocos para facilitar operações futuras é aplicável à fábrica de tubos flexíveis; entretanto, a estrutura das restrições é bastante diferente nesses dois casos.

O problema de realocação de blocos (BRP), que é uma abstração geral do problema de retirada de *containers*, é retratado pelas referências [1, 4, 5, 18, 19, 20]. O BRP é definido a partir de n blocos, numerados de 1 a n. Esses blocos são empilhados diretamente um em cima do outro em um terminal portuário que contém s pilhas *last-in-first-out* (LIFO). Os blocos devem ser retirados do terminal de acordo com a sequência $1, 2, 3, \ldots, n$. Blocos que não tenham sido

retirados devem continuar em uma das pilhas até que o seu respectivo momento de retirada seja alcançado. O objetivo é retirar todos os blocos usando o menor número de movimentos, seja uma movimentação direta, em que o bloco é permanentemente removido, seja uma realocação, em que o bloco é movido de um pilha para outra. É considerado também que um bloco só pode ser movido quando nenhum outro bloco está acima dele e que o número máximo de blocos empilhados é restrito.

A referência [5] apresenta uma heurística mista, denominada método de programação matemática exata, para solucionar porções específicas do problema. A heurística busca encontrar o padrão de realocação que minimiza o número total de movimentos necessários para aplicar uma dada sequência de retirada dos blocos. Em [4], os autores inicialmente provam que o BRP é NP-difícil, e então propõem procedimentos exatos e heurísticos para instâncias pequenas e grandes, respectivamente. Os autores em [18] introduzem uma formulação matemática do BRP que envolve menos variáveis de decisões do que em modelos anteriores. A referência [19] considera duas variantes do BRP: com blocos duplicados e com blocos únicos. É proposto um limite mais rígido para o número de realocações de blocos, permitindo a construção de um algoritmo de *branch-and-bound* mais rápido. Os autores em [20] propõem diversos operadores heurísticos para o BRP, que são utilizados em uma metaheurística construtiva para fornecer soluções de alta qualidade. Por fim, [1] considera o BPC restrito, em que apenas os blocos que estão acima do próximo a ser retirado podem ser movimentados. Nesse trabalho, uma formulação exata, com programação linear inteira, é proposta para o problema.

Todos os estudos mencionados acima lidam com problemas de carregar, descarregar e organizar pilhas. Problemas desses tipos aparecem em uma série de aplicações práticas como terminais de *containers* e na fabricação de aço. A referência [14] sugere uma classificação para esses tipos de problema. Diversas características apresentadas estão presentes no caso do planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis. Em particular, o uso de operadores heurísticos, assim como aplicado nas referências acima, pode ser transportado para o problema de deslocamento de bobinas descrito aqui.

Outra característica importante do caso de estudo desde trabalho é a necessidade de lidar com o problema de agendamento de unidades de movimentação que deslocam as bobinas dentro da fábrica. Essa questão não é o foco na maior parte dos trabalhos mencionados acima.

As referências [7, 8, 13] tratam do problema de programação de guindastes. A referência [7] estuda sobre o carregamento de navios em portos utilizando guindastes, sem restrições relacionadas ao movimento dos guindastes ou das cargas. O modelo resultante torna-se semelhante aos problemas de programação de máquina. Os trabalhos [8] e [13] estudam o caso específico de operação de guindastes em terminais de *containers* portuários. Em ambos os modelos, existe uma restrição que impede os guindastes de cruzarem uns com os outros. Existem também restrições de precedência entre as operações de descarga e carga e restrições temporais para a disponibilidade de cada guindaste para os navios. As contribuições específicas de [13] são a modelagem de

guindastes não uniformes, a inclusão de restrições de precedência arbitrárias entre tarefas e a inclusão da possibilidade de mover guindastes em diferentes direções. A contribuição de [8] está relacionada à inclusão do tempo necessário para movimentar os guindastes entre diferentes posições no modelo. Ambos os trabalhos desenvolvem métodos exatos (programação dinâmica ou *branch-and-bound*) para pequenas instâncias do problema e métodos heurísticos para grandes instâncias.

As referências [11, 17, 23] consideram o conjunto de problemas de programação de guindastes e de gerenciamento de *containers* em terminais. O trabalho [17] lida com a programação de guindastes em pátio de *containers* em que múltiplos guindastes dividem uma única via de movimentação bidirecional no pátio. É considerado o objetivo de minimizar o tempo de espera de caminhões que são carregados e descarregados pelos guindastes. Em [23], os autores estudam o mesmo problema, todavia considerando o conjunto: otimização da programação dos guindastes e as posições de estacionamento dos veículos. A referência [11] considera a atribuição e programação de guindastes no cais e no pátio, e a programação de caminhões no pátio. A otimização combinada de diversas unidades de movimentação e de itens de armazenamento, considerando janelas de tempo, é uma característica também presente no planejamento de deslocamento de bobinas em uma fábrica de tubos flexíveis.

Como o problema específico estudado nesta tese possivelmente ainda não foi abordado em outro trabalho para comparação de resultados, é esperado que estratégias similares às utilizadas pelas das referências deste capítulo também possam ser aplicadas ao planejamento do deslocamento de bobinas, alcançando soluções eficientes. Nesse sentido, assim como os trabalhos apresentados, este trabalho é constituído majoritariamente pelo desenvolvimento de métodos heurísticos, incluindo estratégias com operadores de busca local, modelos matemáticos e meta-heurísticas. Essas estratégias são consolidadas em algoritmos com diferentes limites de tempo de execução que podem ser utilizados em diferentes momentos na fábrica de tubos flexíveis.

3 Descrição do Problema

Desde o pedido do cliente até a entrega, a produção de tubos flexíveis demanda diversas etapas de planejamento para garantir a eficiência operacional. Antes mesmo do começo da produção é necessário alinhar qual a data esperada de entrega de cada tubo flexível e quando eles devem ser produzidos. A construção dos tubos flexíveis é realizada em múltiplas etapas e as camadas são adicionadas de acordo com a especificação do cliente. Essas camadas têm objetivos e composições diversas e, por essa razão, são confeccionadas em diferentes máquinas. A definição de quando o tubo será produzido inclui o agendamento de todas as máquinas envolvidas no processo.

Desde a primeira camada os tubos são armazenados enrolados em bobinas, que são trocadas ao decorrer da produção. As bobinas disponíveis têm diferentes dimensões, suportando tubos flexíveis apenas até um limite de peso e espessura. Assim, após o planejamento da confecção de cada camada é necessário avaliar quais bobinas são capazes de armazenar o tubo flexível com suas novas características e definir a alocação das bobinas de acordo com o agendamento das máquinas.

A transição dos tubos flexíveis entre máquinas demanda movimentação das bobinas pela fábrica e está sujeita a limitação de recursos e espaço físico. Por essa razão, essa movimentação também deve ser planejada a fim de garantir que não existam atrasos para cumprir o agendamento das máquinas e o principal objetivo deste trabalho é encontrar soluções factíveis e eficientes para esta etapa de planejamento.

Uma das limitações para o deslocamento das bobinas, para garantir a integridade dos tubos flexíveis, é a restrição de que os tubos não podem ser armazenados em ambientes externos, expostos a condições climáticas como chuva e sol, até a finalização da sua produção. A fábrica considerada neste trabalho abrange uma área interna e outra externa. A área interna é coberta e contém máquinas para confecção dos tubos flexíveis, bobinas com tubos flexíveis em progresso, bobinas vazias, duas pontes rolantes e dois carros para o deslocamento das bobinas. Esse espaço é adequado para minimizar a exposição dos tubos flexíveis às condições climáticas com exceção das posições dos dois carros, que são apenas parcialmente cobertas. Por essa razão, as bobinas não podem ser armazenadas nos carros, que devem ser utilizados apenas durante o transporte dentro da área interna ou entre as áreas interna e externa. A área externa é responsável por armazenar bobinas vazias e com tubos flexíveis já finalizados, ao aguardo da entrega ao cliente. Essa área é descoberta e a movimentação das bobinas é realizada por meio de guindastes.

A área interna é dividida em três regiões. Duas regiões laterais definem o espaço de trabalho de duas pontes rolantes independentes e a central é o local onde a maior parte das máquinas está instalada. Essa divisão é significativa pois ela representa o modo de produção

dos tubos flexíveis. Para a confecção de cada camada, a bobina que contém o tubo flexível em progresso é posicionada em um local específico de uma região lateral e na outra, também em um local específico, é posicionada uma bobina vazia. Entre as duas posições, na região central, está a máquina responsável por construir a nova camada e, durante o processo produtivo, a bobina com tubo flexível é desenrolada, o tubo se movimenta pela máquina e ao final é enrolado na bobina vazia do outro lado. Um dos carros é destinado à movimentação de bobinas entre as duas regiões laterais.

Outras limitações para o deslocamento das bobinas são:

- Dimensão da área interna: devido à altura da área interna, as pontes rolantes não são capazes de movimentar uma bobina por cima de outra nem posicionar duas bobinas no mesmo local (uma por cima da outra);
- Restrição de movimentação das unidades de movimentação (pontes rolantes e carros):
  - Podem se deslocar de maneiras específicas e por locais pré-definidos;
  - Não estão automaticamente integradas ao plano de movimentação e, portanto, são comandadas manualmente por um operador;
  - Podem deslocar apenas uma bobina por vez;
  - São independentes mas restritas à uma região (não trabalham simultaneamente na mesma região).

Para melhor visualização, um diagrama representando a parte interna da fábrica considerada neste trabalho é apresentado na Figura 3. O diagrama também inclui a representação da localização de 44 bobinas em um dado instante de tempo, como um exemplo de uma configuração possível.

Na figura:

- São definidas as duas regiões laterais (identificadas como Região A e Região B) e a região central (Região C). As pontes rolantes independentes de cada região lateral serão referenciadas como Ponte Rolante A e Ponte Rolante B ao longo do trabalho.
- Os vértices marcados com círculos, quadrados, e losango (vértices 1–23 e 26–59) representam as posições da fábrica. Existem três tipos diferentes de posições:
  - Posição de armazenamento (vértices indicados com círculos): espaço livre de movimentação ou para armazenar bobina.
  - Posição de máquina (vértices 21–23 e 41–50, indicados com quadrados): é o espaço onde as bobinas devem estar posicionadas para fornecer ou receber o tubo flexível da máquina. Dado que a máquina não esteja em uso, esse local pode ser utilizado

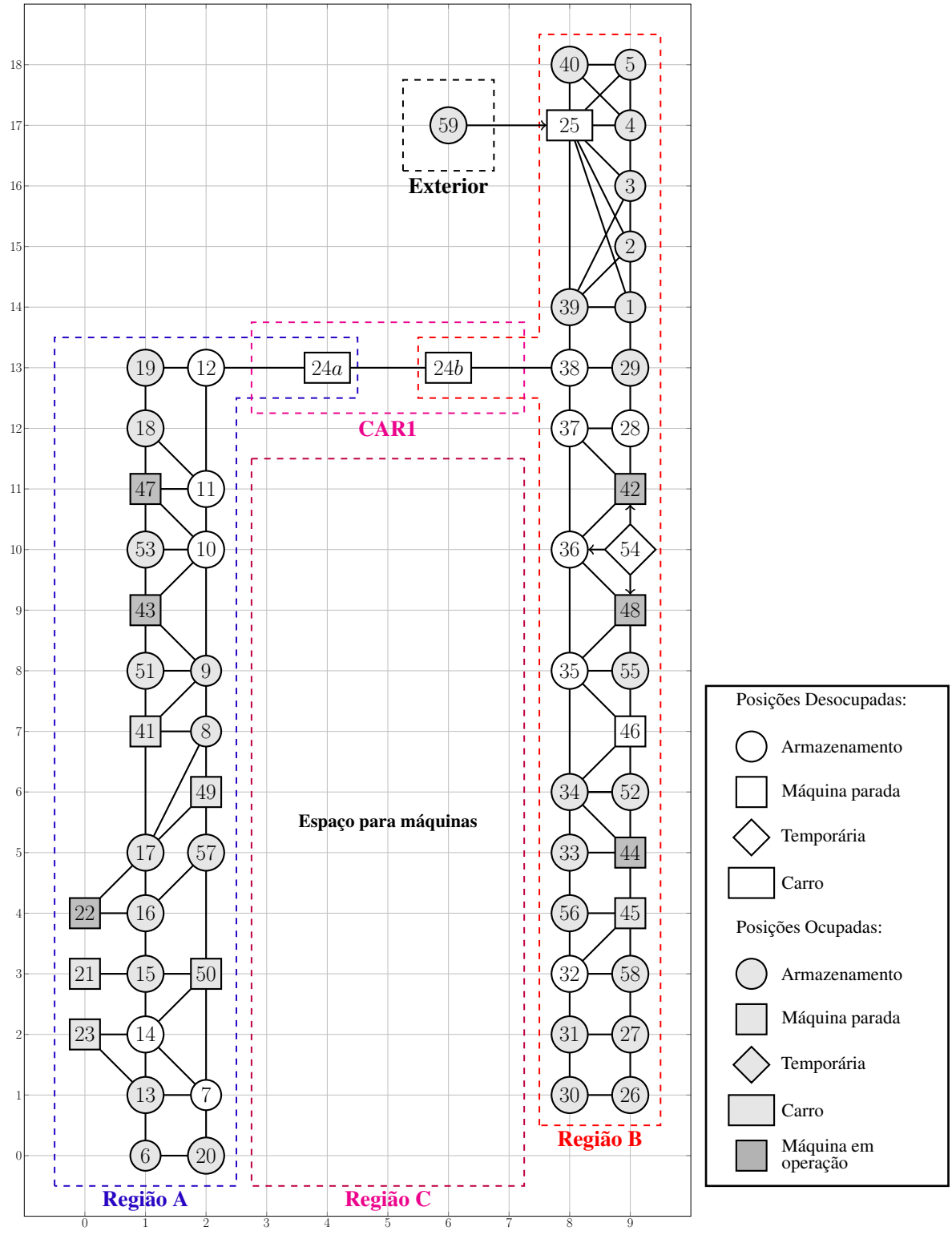


Figura 3 – Exemplo de uma configuração da fábrica. Posições em cinza claro estão ocupadas por alguma bobina (máquinas em cinza claro estão apenas armazenando a bobina). Máquinas em cinza escuro estão ocupadas (processando uma bobina); suas posições podem ser acessadas apenas ao fim da operação.

tanto para armazenar bobina quanto para espaço livre de movimentação. Por outro lado, caso a máquina esteja em uso, a posição fica bloqueada durante todo o intervalo de operação da máquina, sem possibilidade de retirada da bobina (uma posição bloqueada fica indisponível tanto para armazenamento quanto para movimentação).

– Posição temporária (vértice 54, indicado com losango): espaço reservado para a operação das máquinas, com condições específicas de uso. Para simplificação deste trabalho, considera-se que essa posição nunca pode ser utilizada nem para armazenamento nem para construção de caminho (pode ser utilizada manualmente pelos operadores e é possível neste trabalho apenas retirar bobina desta posição).

- As arestas de conexão entre os vértices determinam os deslocamentos permitidos para as unidades de movimentação entre duas posições adjacentes.

- As posições 21, 22 e 23 são casos especiais de posição de máquina. Elas representam as máquinas que constroem a primeira camada do tubo flexível e, por essa razão, não precisam da região central e de duas posições de máquina (uma com o tubo flexível e outra com a bobina vazia) para operar. Em acréscimo, para disponibilizar as bobinas para a Ponte Rolante A, essas posições de máquina se deslocam horizontalmente até às posições 15, $16/17$ ou $13/14$. Nesse deslocamento as posições correspondentes precisam estar desocupadas.

- As regiões A e B são conectadas a partir de um carro (CAR1), que se movimenta entre as posições 12 e 38. CAR1 é acessado pela Ponte Rolante A a partir da Região A e pela Ponte Rolante B a partir da Região B. Ele é representado por duas posições individuais ($24a$ e $24b$), uma vez que ele pode estar tanto próximo da Região A quanto da Região B, após finalizar a movimentação. Todavia, o carro não pode estar na duas posições simultaneamente.

- As posições 59 e 25 representam a conexão entre o exterior e o interior da fábrica. A posição 59 indica a primeira posição da área externa e a integração é realizada pelo carro CAR2, representado pelo vértice 25. Na prática a movimentação entre as duas áreas pode ser realizada em qualquer direção, respeitando o princípio de não armazenar tubos em progresso na área externa. Todavia, neste trabalho, como a informação se o tubo já foi concluído não está disponível, as bobinas podem ser movimentadas apenas na direção exterior$\rightarrow$interior, para fornecer bobinas vazias às máquinas.

- Posições brancas não contêm bobinas, enquanto qualquer posição em cinza claro representa um local ocupado por uma bobina que não está sendo processada. Posições de máquina em cinza escuro representam máquinas que estão em operação, processando as bobinas presentes nessas posições. Essa diferença de cores é relevante pois ela faz referência à uma importante regra para a movimentação das bobinas: posições desocupadas podem

ser utilizadas diretamente, ocupadas requerem movimentos de retirada para construção de caminhos que as utilizem e em operação não podem ser utilizadas enquanto a máquina correspondente não finalizar o processamento da bobina.

Um plano de fundo no formato de grade está incluso no diagrama da fábrica (Figura 3) para ajudar o leitor a identificar a localização das posições e a entender os exemplos. A notação $\#|T(X,Y)$ é usada para referenciar as posições, da seguinte maneira:

- $\#$: identificador inteiro da posição;
- T: tipo da posição, que pode ser A (armazenamento), M (máquina), T (temporária), ou C (carro);
- X e Y: coordenadas horizontal e vertical da posição.

Conforme essa notação, a posição 45, por exemplo, é referenciada como $45|M(9,4)$, dado que ela é uma posição de máquina localizada na coordenada horizontal 9 e na coordenada vertical 4.

A distribuição das 44 bobinas posicionadas na fábrica no exemplo de configuração da Figura 3 pode ser encontrada na Tabela 1.

Tabela 1 – Exemplo de distribuição das bobinas entre posições da fábrica.

ID bobina	ID posição	ID bobina	ID posição	ID bobina	ID posição	ID bobina	ID posição
$B1$	$1\|A(9,14)$	$B2$	$2\|A(9,15)$	$B3$	$3\|A(9,16)$	$B4$	$4\|A(9,17)$
$B5$	$5\|A(9,18)$	$B6$	$6\|A(1,0)$	$B7$	$8\|A(2,7)$	$B8$	$9\|A(2,8)$
$B9$	$13\|A(1,1)$	$B10$	$15\|A(1,3)$	$B11$	$16\|A(1,4)$	$B12$	$17\|A(1,5)$
$B13$	$18\|A(1,12)$	$B14$	$19\|A(1,13)$	$B15$	$20\|A(2,0)$	$B16$	$21\|A(0,3)$
$B17$	$22\|A(0,4)$	$B18$	$23\|A(0,2)$	$B19$	$26\|A(9,1)$	$B20$	$27\|A(9,2)$
$B21$	$29\|A(9,13)$	$B22$	$30\|A(8,1)$	$B23$	$31\|A(8,2)$	$B24$	$33\|A(8,5)$
$B25$	$34\|A(8,6)$	$B26$	$39\|A(8,14)$	$B27$	$40\|A(8,18)$	$B28$	$41\|M(1,7)$
$B29$	$42\|M(9,11)$	$B30$	$43\|M(1,9)$	$B31$	$44\|M(9,5)$	$B32$	$45\|M(9,4)$
$B33$	$47\|M(1,11)$	$B34$	$48\|M(9,9)$	$B35$	$49\|M(2,6)$	$B36$	$50\|M(2,3)$
$B37$	$51\|A(1,8)$	$B38$	$52\|A(9,6)$	$B39$	$53\|A(1,10)$	$B40$	$55\|A(9,8)$
$B41$	$56\|A(8,4)$	$B42$	$57\|A(2,5)$	$B43$	$58\|A(9,3)$	$B44$	$59\|A(6,17)$

Um exemplo sobre a construção de caminhos e sobre a regra de que não é possível passar uma bobina por cima de uma posição ocupada, é a movimentação da bobina da posição $8|A(2,7)$ para a posição $46|M(9,7)$. Esse movimento não pode ser executado diretamente porque não existe um caminho livre entre essas posições. A maneira mais rápida de tornar esse movimento factível seria primeiramente executar uma operação de retirada deslocando a bobina em $9|A(2,8)$ para a posição $28|A(9,12)$.

O planejamento de movimentação das bobinas deve ser realizado a partir do agendamento das máquinas e da definição de quais bobinas serão utilizadas por cada máquina, para

alguns dias de produção. Na organização atual da fábrica considerada neste trabalho (antes da implantação dos algoritmos desenvolvidos), o planejamento citado deveria ser realizado manualmente, diretamente no chão de fábrica. Na prática é costume que o planejamento de movimentação das bobinas simplesmente não seja executado. Os operadores apenas definem em tempo real a trajetória das bobinas, baseado na sua intuição e experiência e, no geral, a tomada de decisão para deslocamento das bobinas considera apenas o agendamento da máquina corrente e, no máximo, os próximos imediatos. Isso acontece porque realizar o planejamento manualmente demandaria horas de trabalho, levando à necessidade de ter um funcionário quase exclusivo para essa atividade.

Sem planejamento e sem considerar um horizonte de tempo de alguns dias, a movimentação é amplamente susceptível a erros e não é possível garantir que o deslocamento das bobinas é eficiente, nem prever se será viável concretizar o agendamento de máquinas e bobinas sem atrasos. Alguns movimentos sem uma análise cautelosa podem levar a um deadlock (quando não existe a possibilidade de abrir um caminho da bobina até a posição de máquina destino sem a retirada de bobinas da parte interna da fábrica) e outros podem distanciar a bobina de uma posição de máquina que será utilizada em breve.

Uma exceção da necessidade de planejamento de movimentação das bobinas é o deslocamento dos tubos finalizados da área interna para a externa. Como os tubos irão apenas aguardar a entrega ao cliente e não existem restrições de tempo relevantes nesse caso, o movimento das pontes rolantes para disponibilizar os tubos no CAR2 pode ser realizado em momentos livres, quando as pontes rolantes não estão sendo utilizadas para mover os tubos não finalizados entre as posições de máquinas. É importante destacar todavia que, quanto antes a bobina for deslocada para a área externa, melhor será para o planejamento interno, que contará com mais uma posição disponível.

Um ponto importante é que em algumas situações é necessário realizar o replanejamento da movimentação das bobinas. Mesmo quando o planejamento é realizado pode ser necessário revê-lo emergencialmente quando o agendamento das máquinas sofre alguma alteração urgente, quando as máquinas ou pontes rolantes sofrem pane ou algum defeito, quando os funcionários necessários para a operação não estão disponíveis ou quando o planejamento prévio de movimentação das bobinas não é executado conforme estipulado.

As limitações e objetivos apresentados neste capítulo são válidos tanto para o planejamento quando o replanejamento. Essas atividades se diferenciam apenas pelo momento em que são executadas. O planejamento deveria ser realizado diariamente, a partir da definição mais atualizada de agendamento das máquinas e bobinas alocadas, enquanto o replanejamento pode ser necessário a qualquer momento, quando existe alguma alteração na execução do planejamento que invalida os próximos movimentos planejados ou quando é demandado um ajuste emergencial do agendamento das máquinas.

A fim de melhorar a eficiência e reduzir atrasos na operação, a otimização do plane-

jamento e do replanejamento de movimentação das bobinas é proposta neste estudo, a partir de algoritmos que serão discutidos nos próximos capítulos. Para explicar os algoritmos e a modelagem do problema, as seguintes definições serão importantes e empregadas ao longo deste trabalho:

Submovimento: o ato de mover uma bobina de uma posição para outra vizinha.

Movimento: o ato de transportar uma bobina de uma posição para outra, através de um caminho livre, sem interrupções, utilizando a mesma unidade de movimentação (U.M.) (ponte rolante ou carro). Um movimento é composto por um ou mais submovimentos.

Operação: o ato de pegar uma bobina em uma posição e colocar em outra, independentemente da U.M. utilizada, através de um caminho livre, sem interrupções. Uma operação é composta por um ou mais movimentos.

Subtarefa: consiste em mover uma bobina específica para uma posição de máquina específica, como definido pela tarefa pai.

Tarefa: é um conjunto de subtarefas que devem ser executadas para permitir o processamento da bobina. Cada tarefa tem, em geral, duas subtarefas mas, em casos específicos, ela pode ser composta de uma única subtarefa.

Processamento da bobina: quando as subtarefas associadas a uma dada tarefa são completadas (i.e., as duas bobinas são posicionadas nas posições de máquinas corretas), então a máquina pode começar a processar as bobinas para construir uma camada do tubo. Esse processamento é referenciado como o *processamento da bobina*.

Horário de chegada: horário em que uma dada bobina alcança a posição de máquina destino, no fim da subtarefa.

Plano de produção: a programação que estabelece os horários de referência em que cada processamento de bobina deve começar e encerrar. Ele contém informações referentes ao agendamento de máquinas e também da alocação de bobinas, que são etapas de planejamento anteriores ao deslocamento das bobinas.

Horário inicial: horário em que cada processamento da bobina deve idealmente começar, de acordo com o plano de produção. A diferença entre o horário inicial real e o ideal é tratada como uma função objetivo.

Horário final: horário máximo em que cada processamento de bobina deve terminar, de acordo com o plano de produção. Essa especificação é fixa (independentemente do horário inicial, o horário final é sempre respeitado) e é uma simplificação do modelo; caso contrário, qualquer atraso ou adiantamento no horário de chegada impactaria em uma modificação no horário final.

As definições apresentadas são descritas no Exemplo 1 e, em seguida, esses conceitos são reforçados no Exemplo 2 que indica uma solução candidata para o planejamento do deslocamento de bobinas para uma tarefa.

Exemplo 1 *Exemplos de operação, movimento, submovimento, tarefa, subtarefa e plano de produção são fornecidos utilizando a configuração apresentada na Figura 3 e as tabelas auxiliares (Tabela 2 e Tabela 3):*

Operação: *um exemplo de operação é o deslocamento da bobina* $B8$*, da posição* $9|A(2,8)$*, para a posição* $46|M(9,7)$*, utilizando o caminho* $9|A(2,8) \rightarrow 10|A(2,10) \rightarrow 11|A(2,11) \rightarrow 12|A(2,13) \rightarrow 24a|C(4,13) \rightarrow 24b|C(6,13) \rightarrow 38|A(8,13) \rightarrow 37|A(8,12) \rightarrow 36|A(8,10) \rightarrow 35|A(8,8) \rightarrow 46|M(9,7)$.

Movimento: *a operação apresentada é composta de três movimentos:* $9|A(2,8) \rightarrow 10|A(2,10) \rightarrow 11|A(2,11) \rightarrow 12|A(2,13) \rightarrow 24a|C(4,13)$ *utilizando a Ponte Rolante A,* $24a|C(4,13) \rightarrow 24b|C(6,13)$ *com o CAR1 (representa a movimentação do carro da Região A para a Região B), e* $24b|C(6,13) \rightarrow 38|A(8,13) \rightarrow 37|A(8,12) \rightarrow 36|A(8,10) \rightarrow 35|A(8,8) \rightarrow 46|M(9,7)$ *utilizando a Ponte Rolante B.*

Submovimento: *O primeiro movimento apresentado é composto de quatro submovimentos:* $9|A(2,8) \rightarrow 10|A(2,10)$*,* $10|A(2,10) \rightarrow 11|A(2,11)$*,* $11|A(2,11) \rightarrow 12|A(2,13)$*,* $12|A(2,13) \rightarrow 24a|C(4,13)$.

Tarefa e subtarefa: *Um exemplo de tarefa com duas subtarefas pode ser encontrado na Tabela 2. Essa tarefa indica um processamento de bobina que deve começar em 18 unidades de tempo (u.t.) e finalizará em 300 u.t. (não existe uma relação direta entre u.t. e unidades reais para preservar os dados da empresa). A primeira subtarefa envolve mover a bobina* $B26$ *para a posição* $41|M(1,7)$ *e na segunda é necessário mover a bobina* $B1$ *para a posição* $46|M(9,7)$.

Tabela 2 – Exemplo de uma tarefa com duas subtarefas.

ID da Tarefa	**Início** (u.t.)	**Fim** (u.t.)	**Bobina1**	**Bobina2**	**Posição1**	**Posição2**
1	18	300	$B26$	$B1$	$41\|M(1,7)$	$46\|M(9,7)$

em que:
ID da Tarefa é o identificador (ID) da tarefa;
Início (u.t.) é o horário inicial desejado para o processamento da bobina (em u.t.);
Fim (u.t.) é o horário definido para finalizar o processamento da bobina (em u.t.);
Bobina1 e **Bobina2** são as bobinas designadas para as subtarefas 1 e 2, respectivamente;
Posição1 e **Posição2** são as posições destino das subtarefas 1 e 2, respectivamente.

Plano de produção: *Um plano de produção é composto de diversas tarefas, de acordo com a quantidade de dias considerada no agendamento de máquinas. Podem existir tarefas*

para diferentes máquinas, assim como múltiplas tarefas para a mesma máquina, como apresentado na Tabela 3. Neste plano, que contempla um horizonte de planejamento de aproximadamente 3000 *u.t., as duplas de posição de máquina* $45|M(9,4)$ *e* $50|M(2,3)$*,* $49|M(2,6)$ *e* $44|M(9,5)$*,* $42|M(9,11)$ *e* $47|M(1,11)$*,* $43|M(1,9)$ *e* $48|M(9,9)$ *foram utilizadas entre* 2 *ou* 3 *vezes. Também é possível perceber que a duração de operação das máquinas (diferença entre Horário Inicial e Horário Final) não é constante, variando entre máquinas e até entre execuções na mesma máquina. Como exemplo, enquanto a operação da máquina da tarefa* 1 *tem duração planejada de* 1200 *u.t., na tarefa* 2 *esse tempo é de* 471 *u.t..*

Tabela 3 – Exemplo de uma plano de produção com onze tarefas.

ID da Tarefa	**Início** (u.t.)	**Fim** (u.t.)	**Bobina1**	**Bobina2**	**Posição1**	**Posição2**
1	3	1203	$B19$	$B40$	$45\|M(9,4)$	$50\|M(2,3)$
2	6	477	$B16$	$B25$	$49\|M(2,6)$	$44\|M(9,5)$
3	21	1041	$B22$	$B36$	$42\|M(9,11)$	$47\|M(1,11)$
4	300	597	$B18$	$B8$	$43\|M(1,9)$	$48\|M(9,9)$
5	483	1743	$B26$	$B6$	$44\|M(9,5)$	$49\|M(2,6)$
6	603	1047	$B1$	$B30$	$43\|M(1,9)$	$48\|M(9,9)$
7	1044	1338	$B30$	$B22$	$42\|M(9,11)$	$47\|M(1,11)$
8	1209	2268	$B33$	$B14$	$45\|M(9,4)$	$50\|M(2,3)$
9	1341	1710	$B22$	$B30$	$42\|M(9,11)$	$47\|M(1,11)$
10	1749	2349	$B2$	$B7$	$49\|M(2,6)$	$44\|M(9,5)$
11	2274	3069	$B33$	$B15$	$45\|M(9,4)$	$50\|M(2,3)$

Exemplo 2 *Como um exemplo, assuma a configuração da fábrica e as posições das bobinas apresentadas na Figura 3 e o planejamento de uma única tarefa da Tabela 2. Nessa configuração, as bobinas* $B26$ *e* $B1$ *estão localizadas nas posições* $39|A(8,14)$ *e* $1|A(9,14)$*, respectivamente. Uma solução candidata para o planejamento do deslocamento de bobinas é apresentada na Tabela 4 (cada linha representa um submovimento).*

É possível notar que o planejamento de uma única tarefa é executada com 8 operações, 10 movimentos e 25 submovimentos. Seis dessas operações são necessárias para liberar o caminho para completar as subtarefas.

Na Tabela 2 é definido que o horário inicial do processamento da bobina asssociado à tarefa é igual a 18 u.t. Como a subtarefa 1 é finalizada em 21 u.t., existe atraso de três u.t., significando que o processamento da bobina precisará ser atrasado até que as duas subtarefas sejam completadas, em 21 u.t.. Por outro lado, como a subtarefa 2 é finalizada em 16 u.t., existe um adiantamento de duas u.t. nessa subtarefa e a U.M. fica livre duas u.t. antes do fim da tarefa, ficando disponível para uso em outra tarefa. Essa situação ilustra que uma tarefa pode ter atraso e adiantamento simultaneamente.

Tabela 4 – Exemplo de solução candidata.

tarefa	subtarefa	operação	movimento	submovimento	bobina	posição inicial	posição final	tempo inicial	tempo final	U.M.
-	-	1	1	1	$B10$	$15\|A(1,3)$	$14\|A(1,2)$	0	3	A
-	-	1	1	2	$B10$	$14\|A(1,2)$	$7\|A(2,1)$	3	3	A
-	-	2	1	1	$B11$	$16\|A(1,4)$	$15\|A(1,3)$	3	6	A
-	-	2	1	2	$B11$	$15\|A(1,3)$	$14\|A(1,2)$	6	6	A
-	-	3	1	1	$B12$	$17\|A(1,5)$	$16\|A(1,4)$	6	9	A
-	-	3	1	2	$B12$	$16\|A(1,4)$	$15\|A(1,3)$	9	9	A
-	-	4	1	1	$B7$	$8\|A(2,7)$	$17\|A(1,5)$	9	12	A
-	-	4	1	2	$B7$	$17\|A(1,5)$	$16\|A(1,4)$	12	12	A
-	-	5	1	1	$B8$	$9\|A(2,8)$	$8\|A(2,7)$	12	15	A
-	-	5	1	2	$B8$	$8\|A(2,7)$	$17\|A(1,5)$	15	15	A
-	-	6	1	1	$B28$	$41\|M(1,7)$	$8\|A(2,7)$	15	18	A
1	1	7	1	1	$B26$	$39\|A(8,14)$	$38\|A(8,13)$	10	13	B
1	1	7	1	2	$B26$	$38\|A(8,13)$	$24b\|C(6,13)$	13	13	B
1	1	7	2	1	$B26$	$24b\|C(6,13)$	$24a\|C(4,13)$	13	18	CAR1
1	1	7	3	1	$B26$	$24a\|C(4,13)$	$12\|A(2,13)$	18	21	A
1	1	7	3	2	$B26$	$12\|A(2,13)$	$11\|A(2,11)$	21	21	A
1	1	7	3	3	$B26$	$11\|A(2,11)$	$10\|A(2,10)$	21	21	A
1	1	7	3	4	$B26$	$10\|A(2,10)$	$9\|A(2,8)$	21	21	A
1	1	7	3	5	$B26$	$9\|A(2,8)$	$41\|M(1,7)$	21	21	A
1	2	8	1	1	$B1$	$1\|A(9,14)$	$39\|A(8,14)$	13	16	B
1	2	8	1	2	$B1$	$39\|A(8,14)$	$38\|A(8,13)$	16	16	B
1	2	8	1	3	$B1$	$38\|A(8,13)$	$37\|A(8,12)$	16	16	B
1	2	8	1	4	$B1$	$37\|A(8,12)$	$36\|A(8,10)$	16	16	B
1	2	8	1	5	$B1$	$36\|A(8,10)$	$35\|A(8,8)$	16	16	B
1	2	8	1	6	$B1$	$35\|A(8,8)$	$46\|M(9,7)$	16	16	B

Algumas características/restrições importantes devem ser consideradas na modelagem do problema:

- Cada movimento envolve içar, mover por um caminho livre e descer uma bobina na posição destino. Quase todo tempo necessário para executar um movimento é gasto pelas ações de içar e descer a bobina. Isso significa que o tempo de deslocamento de uma ponte rolante é quase constante para qualquer movimento executado na mesma região da fábrica. Por exemplo, na Figura 3, o tempo necessário para mover uma bobina da posição $39|A(8,14)$ para a posição $38|A(8,13)$ pode ser considerado igual ao tempo para mover a mesma bobina até a posição $46|M(9,7)$.

- O tempo total gasto em um movimento é atribuído ao seu primeiro submovimento. Essa simplificação é possível porque o tempo de deslocamento da U.M. em um movimento é muito menor do que os tempos de içar e descer a bobina. Por essa razão, o tempo de deslocamento é desconsiderado e a representação do tempo restante é concentrada no primeiro submovimento, sem que o modelo perca fidelidade. Por convenção, os intervalos

de tempo do movimento são modelados incluindo o horário inicial e excluindo o final (i.e., $[\text{início}, \text{fim}[$).

- Durante a elaboração do plano de produção em geral é considerada uma alta alocação das máquinas. Como apresentado no exemplo da Tabela 3, a dupla de posições de máquina $45|M(9,4)$ e $50|M(2,3)$ é utilizada por 1200 u.t. na Tarefa 1 e, até a próxima utilização, na Tarefa 8, existe um intervalo de seis u.t.. O intervalo é suficiente apenas para retirar as bobinas que estavam em uso previamente e posicionar as novas. Essa característica, que se repete entre outras tarefas, demonstra a necessidade de um plano de movimentação das bobinas eficiente (sendo necessário em alguns casos até o pré-posicionamento das bobinas), para garantir que o deslocamento final das bobinas seja realizado em poucas unidades de tempo e não atrase as tarefas em sequência.

- Os movimentos executados por uma dada U.M. devem ser executados sequencialmente. Todavia, duas (ou mais) U.M.s podem ser utilizadas em paralelo.

- Nenhuma posição pode conter mais de uma bobina por vez, assim como nenhuma U.M. pode transportar mais de uma bobina por vez.

- O tempo de deslocamento dos carros CAR1 e CAR2 pode ser assumido como zero u.t. quando eles não estão transportando uma bobina.

- Alguns movimentos envolvendo máquinas podem bloquear posições durante sua execução (caso especial das posições 22 e 23):

  - Os movimentos $23|M(0,2) \rightarrow 13|A(1,1)$ ou $13|A(1,1) \rightarrow 23|M(0,2)$ só podem ser realizados se a posição $14|A(1,2)$ está livre.
  - Os movimentos $23|M(0,2) \rightarrow 14|A(1,2)$ ou $14|A(1,2) \rightarrow 23|M(0,2)$ só podem ser realizados se a posição $13|A(1,1)$ está livre.
  - Os movimentos $22|M(0,4) \rightarrow 16|A(1,4)$ ou $16|A(1,4) \rightarrow 22|M(0,4)$ só podem ser realizados se a posição $17|A(1,5)$ está livre.
  - Os movimentos $22|M(0,4) \rightarrow 17|A(1,5)$ ou $17|A(1,5) \rightarrow 22|M(0,4)$ só podem ser realizados se a posição $16|A(1,4)$ está livre.

A eficiência operacional mencionada como objetivo do planejamento e do replanejamento, pode ser definida por múltiplos objetivos, organizados por prioridade. Por essa razão, o problema de otimização considerado aqui é definido como uma otimização multiobjetivo com cinco funções objetivo combinadas em uma composição lexicográfica. Existem diversas maneiras de definir e modelar problemas multiníveis, mas a escolha pela composição lexicográfica pode ser explicada pela organização de prioridade dos objetivos, já que, neste trabalho, o aprimoramento do primeiro objetivo é sempre preferencial em relação ao segundo e assim por diante.

A organização de prioridade e a explicação de cada uma das funções objetivo são definidas a seguir, considerando F_1 como objetivo mais importante e F_5 o menos importante.

$$F_1 \rightarrow F_2 \rightarrow F_3 \rightarrow F_4 \rightarrow F_5 \tag{3.1}$$

F_1 – minimizar o número de subtarefas incompletas: eventualmente, um conjunto de movimentos pode levar a uma situação de *deadlock*, em que não é possível finalizar parte das subtarefas porque as operações necessárias para completá-las se tornam impossíveis. Nesses casos, o operador deve mover algumas bobinas para o exterior da fábrica. Como o algoritmo proposto não contempla movimentações da área interna para a externa, ele para quando um *deadlock* é alcançado. Para evitar essa situação, o primeiro objetivo a ser otimizado é a quantidade restante de subtarefas a finalizar.

F_2 – minimizar o número de operações cujo destino final é CAR1(posições $24a|C(4,13)$ ou $24b|C(6,13)$) ou CAR2 (posição $25|C(8,17)$): a longa exposição a condições climáticas desfavoráveis (sol excessivo, chuva, etc) pode afetar o conteúdo da bobina (o tubo flexível). Como os carros estão localizados em áreas parcialmente cobertas da fábrica, operações com destino final em um carro, apesar de possíveis, não são permitidas. Por padrão, o algoritmo proposto também é finalizado caso seja necessário utilizar algum carro como destino final de uma operação.

As duas primeiras funções objetivo dizem respeito à factibilidade do problema. Caso elas não sejam atendidas assume-me que a otimização falhou e que é necessário alterar as condições iniciais, seja modificando o agendamento das máquinas / alocação das bobinas, seja movendo algumas bobinas vazias que não serão utilizadas para a área externa.

Em algumas situações específicas quando a fábrica está com alta lotação de bobinas, é possível retirar a condição de factibilidade da segunda função de objetivo. Neste caso, é desejado apenas encontrar o número mínimo de operações cujo destino final é CAR1 ou CAR2 e o operador deve ficar atento e avaliar quais tubos flexíveis e por quanto tempo estão armazenados nos carros.

F_3 – minimizar o atraso total: uma tarefa é dita atrasada quando pelo menos uma subtarefa associada tem horário de chegada superior ao horário inicial do processamento da bobina, já que, nesse caso, o processo de produção do tubo necessariamente começará depois do horário inicial desejado. O atraso é avaliado como a diferença máxima entre o horário de chegada das subtarefas e o horário inicial do processamento da bobina, assim como apresentado na equação (3.2). O objetivo F_3 é a soma do atraso associado à todas tarefas, como definido na equação (3.3).

$$At(i) = \max\{0, [\max(C(i,1), C(i,2)) - I(i)]\} \tag{3.2}$$

$$F_3 = \sum_{i \in \mathcal{T}} At(i) \tag{3.3}$$

em que:

- $At(i)$ é o atraso associado à tarefa i.
- $C(i,j)$ é o horário de chegada da subtarefa j (i.e., o horário em que a bobina da subtarefa j alcançou a posição de destino), incluída na tarefa i.
- $I(i)$ é o horário de início do processamento da bobina associado à tarefa i.
- $\mathcal{T}$ é o conjunto de tarefas.

F_4 – maximizar o adiantamento total: se a diferença entre o horário de início da tarefa e o horário de conclusão da subtarefa é maior do que zero, define-se que essa subtarefa está adiantada (ver equação (3.4)). O objetivo F_4 é a soma do adiantamento associado a todas as subtarefas, como apresentado na equação (3.5), em que $Ad(i,j)$ é o adiantamento associado a subtarefa j da tarefa i. O efeito desse objetivo consiste em alcançar soluções que liberem a U.M. antecipadamente, permitindo adiantar a execução do próximo movimento que está agendado para essa unidade.

$$Ad(i,j) = \max\{0, [I(i) - C(i,j)]\} \tag{3.4}$$

$$F_4 = \sum_{i \in \mathcal{T}} \sum_{j \in \{1,2\}} Ad(i,j) \tag{3.5}$$

O horário inicial do plano de produção é tratado como um valor de referência, uma vez que o horário de chegada da bobina (conclusão da subtarefa) pode ser anterior ou posterior. Já o horário final do plano de produção é sempre igual ao do processamento da bobina. Assim, o atraso e o adiantamento são avaliados em relação ao horário inicial do plano de produção. Uma posição de máquina envolvida em um processamento de bobina continua bloqueada desde o horário de chegada da bobina nessa posição até o horário final do processamento da bobina.

F_5 – minimizar o número de operações executadas: é interessante reduzir a quantidade de operações necessária para completar um conjunto de tarefas, uma vez que essas operações demandam tempo e causam desgaste da U.M. ao longo do tempo (i.e., o número de manutenções inesperadas e preventivas tendem a crescer se mais operações são executadas).

Os três últimos critérios são relacionados ao desempenho da solução e estão condicionados aos dois primeiros objetivos. É sempre desejável obter o menor atraso possível (F_3), seguido do maior adiantamento (F_4) e, por último, no menor número de operações (F_5), assim como definido na composição lexicográfica, desde que os dois primeiros objetivos sejam iguais à zero.

As variáveis de decisão do problema proposto são os submovimentos / movimentos / operações a serem executados e a janela de tempo em que eles estão programados.

Como o planejamento manual de movimentação das bobinas para a produção de tubos flexíveis é muito trabalhoso, é estabelecida a necessidade de construção de um algoritmo capaz de planejar o deslocamento das bobinas de forma automatizada. Este algoritmo deve considerar todas as limitações e restrições apresentadas neste capítulo, assim como as funções objetivo com a respectiva ordem de prioridade.

Existem duas demandas para o tempo de execução do algoritmo. A primeira, que diz respeito ao planejamento diário, é limitada a algo em torno de quatro horas de execução. Já a segunda, é referente ao replanejamento da movimentação e, por isso, é restrita a apenas cinco minutos. Com base na diferença de tempo de execução nesses dois casos, são propostos dois algoritmos para tratar o problema, considerando os mesmos objetivos e restrições, mas variando o tempo necessário para completar sua execução. Esses algoritmo serão apresentados nos próximos capítulos.

4 Problema de Distância Específica

Problemas de otimização são definidos como a minimização de funções em algum espaço de variáveis de decisão. A formulação de estratégias significativas para procurar soluções no espaço de variáveis de decisão não depende apenas das propriedades da própria função objetivo, mas também da existência de alguma estrutura no espaço de variáveis que, de alguma maneira, estabeleça correlações entre os valores de função objetivo de diferentes pontos no espaço. Na otimização de funções de variáveis contínuas, as propriedades usuais do espaço $\mathbb{R}^n$ são suficientes. Entretanto, em problemas combinatórios, normalmente é necessário definir um *design* explícito do espaço das variáveis de decisão, assim como de operadores que são utilizados na busca, de forma que o algoritmo seja capaz de decidir quais novos pontos devem ser avaliados baseando-se na informação adquirida em pontos avaliados anteriormente. O requisito mínimo normalmente necessário é a existência de alguma noção de *distância* que permite a definição de uma *vizinhança* em que os operadores de *busca local* trabalham. Uma discussão sobre essas questões é apresentada em [2, 3, 15].

No problema específico considerado aqui, alguns dos blocos de construção mais óbvios para definir a medida de distância não funcionariam. Por exemplo, mover uma bobina de uma posição para outra que é fisicamente perto não constitui necessariamente um movimento dentro da vizinhança porque, muito frequentemente, a posição destino pode estar ocupada, requisitando complexos movimentos prévios para liberá-la. Em acréscimo, o caminho para uma posição próxima da posição atual da bobina pode estar obstruido, enquanto o caminho para outra posição que é fisicamente mais distante pode estar livre. Além disso, como os componentes principais da quantidade de tempo necessária para mover uma bobina estão relacionados com os tempos de pegar/subir/descer/soltar, as distâncias físicas entre origem e destino têm pouca relevância como variáveis suporte para organizar a busca pela função objetivo mínima.

Outra medida de distância frequentemente empregada em problemas de otimização combinatória é a distância de Manhattan, que representa o número de trocas elementares nas variáveis de decisão que são necessárias para mover de um ponto no espaço de variáveis de decisão para outro ponto. Essa distância também não funcionaria neste problema, devido à heterogeneidade das diferentes ações de submovimentos. Novamente, se um submovimento envolver ações de pegar/subir/descer/soltar, ele tem um custo muito mais relevante do que submovimentos que não requisitam essas ações. Também existe diferença entre movimentos que são diretamente parte de uma tarefa e movimentos que são executados apenas para desbloquear um caminho para outro movimento.

Um procedimento adequado para medir distâncias neste trabalho é definido de acordo com uma variação do algoritmo de Dijkstra para encontrar caminhos mínimos em grafos [6]. A

distância a ser medida refere-se ao esforço para deslocar uma bobina do vértice origem O para o vértice destino D, com O e D indicando posições no grafo que representa a planta (por exemplo Figura 3). As principais etapas desse procedimento, também denominado *ReelFastestPath*, são:

ReelFastestPath

1. Defina $custo\_caminho(O) = 0$ e $pai(O) = 0$.
2. Para cada vértice i restante, defina $custo\_caminho(i) = \infty$ e $pai(i) = \infty$.
3. Defina todos os vértices como não visitados.
4. Enquanto D não é visitado:
   a) Defina $U = \arg\min_{i \in \mathcal{U}} custo\_caminho(i)$, em que $\mathcal{U}$ é o conjunto de vértices não visitados.
   b) Visite U.
   c) Para cada vértice $V \in \mathcal{U}$:
      i. Avalie $d(V)$ como:
      $$d(V) = custo\_caminho(U) + custo(U, V) \tag{4.1}$$
      ii. Se $d(V) < custo\_caminho(V)$, defina $custo\_caminho(V) = d(V)$ e $pai(V) = U$.
5. Recupere o caminho mínimo entre O e D por meio do array pai.

Nesse procedimento:

- $custo\_caminho(P)$ é o custo para alcançar o vértice P, a partir de O.
- $pai(U)$ é o pai do vértice U no caminho mínimo.
- $custo(U, V)$ é o custo da conexão direta entre U e V.

No método clássico de Dijkstra, o custo de incluir um arco que conecta os nós U e V é uma constante previamente conhecida $custo(U, V)$ e, se esse arco não existe, então $custo(U, V)$ é infinito. Na distância definida aqui, $custo(U, V)$ é a soma de três fatores:

$$custo(U, V) = tempo_{MOV}(U, V) + tempo_{LIMPEZA}(V) + tempo_{ARCO}(U, V) \tag{4.2}$$

em que:

- $tempo_{MOV}(U, V)$ é o tempo gasto pelo submovimento entre as posições U e V:
  - Caso o submovimento $U \rightarrow V$ corresponda ao início de um movimento (primeiro submovimento), o tempo necessário para mover uma bobina de U para V é o tempo gasto pela respectiva U.M para executar um movimento: $tempo_{MOV}(U, V) = UM_{tempo}(U, V)$.
  - Caso contrário: $tempo_{MOV}(U, V) = 0$.
- $tempo_{LIMPEZA}(V)$ é o tempo necessário para desocupar a posição V, se ela está ocupada nesse momento por alguma bobina. Esse tempo é calculado usando o procedimento *PositionClearTimeEstimation* descrito na seção 4.1.
- $tempo_{ARCO}(U, V)$ é um pequeno valor que é empregado para prevenir a construção de caminhos excessivamente longos. Em todos os experimentos assume-se que $tempo_{ARCO}(U, V) = 0.001$ u.t. para cada arco no grafo da fábrica.
- $UM_{tempo}(U, V)$ é o tempo necessário pela U.M. designada para mover uma bobina da posição U para a posição V, através de um caminho livre (i.e., o tempo que a U.M. gasta para executar o movimento).

Essa diferença em como os custos dos caminhos são avaliados determina algumas características únicas para o problema de caminho mínimo solucionado por *ReelFastestPath*:

- O custo do caminho depende da sequência de posições uma vez que o custo de um novo submovimento depende da U.M. que será utilizada nesse submovimento e da U.M utilizada no movimento anterior.
- O posicionamento corrente das bobinas na fábrica afeta o caminho construído, porque o tempo necessário para desocupar cada posição ocupada exerce um papel importante no custo do caminho.
- Dado que o tempo necessário para mover uma bobina na mesma região da fábrica é sempre o mesmo, independentemente das posições de origem e destino, não existe diferença (em termos do tempo do movimento) em construir um caminho com 1 ou 10 submovimentos. O termo $tempo_{ARCO}(U, V)$ é empregado para evitar a geração de caminhos excessivamente longos. Essa condição define que, dado dois caminhos que necessitam do mesmo tempo para serem finalizados, aquele com menos submovimento sempre é priorizado.

4.1 Estimativa de tempo de limpeza de posição

Como discutido, na maioria dos casos não existe caminho livre entre as posições inicial (O) e final (D). Portanto, o procedimento *ReelFastestPath* deve levar em consideração não apenas

o custo da operação $O \rightarrow D$, mas também o custo das operações necessárias para desocupar o caminho entre O e D. Um novo procedimento que estima o tempo para as operações de desocupar caminhos, em um custo computacional razoável, é proposto aqui. Esse operador, que é nomeado *PositionClearTimeEstimation*, é executado conforme as etapas a seguir e exemplificado no Exemplo 3.

PositionClearTimeEstimation

- Dado que $\mathcal{F}$ define o conjunto de posições desocupadas e disponíveis (i.e., as posições que estão desocupadas e que não serão utilizadas no caminho mínimo corrente entre O e D).

1. Para cada posição $K \in \mathcal{F}$:
   a) Defina K como a posição desocupada e disponível corrente.
   b) Encontre o caminho mínimo entre K e todas as posições restantes na fábrica. Nessa etapa, o termo $custo(U, V)$ da equação (4.1) é calculado como apresentado na equação (4.3).

$$custo(U,V) = \begin{cases} 0 & \text{se a posição } V \text{ está desocupada} \\ UM_{tempo}(U,V) & \text{caso contrário} \end{cases} \tag{4.3}$$

2. Considere que $cm(K, P)$ estabelece o caminho mínimo entre as posições K e P. Para cada posição i, defina $tempo_{LIMPEZA}(i) = \min_{j \in \mathcal{F}} cm(j, i)$.
3. Para cada posição i que está atualmente bloqueada, defina $tempo_{LIMPEZA}(i) = \infty$.
4. Para cada posição i que está atuamente desocupada, defina $tempo_{LIMPEZA}(i) = 0$.

Exemplo 3 *Como um exemplo, a execução de PositionClearTimeEstimation para a configuração da fábrica na Figura 3 determinaria que:* (i) *a posição* $15|A(1,3)$ *pode ser desocupada por uma única operação da Ponte Rolante A (a bobina* $B10$ *na posição* $15|A(1,3)$ *pode ser deslocada para a posição* $14|A(1,2)$*);* (ii) *a posição* $16|A(1,4)$ *pode ser desocupada com duas operações da Ponte Rolante A (uma operação necessária para desocupar* $15|A(1,3)$ *e uma operação para mover* $B11$ *de* $16|A(1,4)$ *para* $15|A(1,3)$*);* (iii) *a posição* $17|A(1,5)$ *pode ser desocupada com três operações da Ponte Rolante A (duas necessárias para desocupar* $16|A(1,4)$ *e uma operação para mover a bobina* $B12$ *de* $17|A(1,5)$ *para* $16|A(1,4)$*); e asssim por diante.*

Esse procedimento estima o tempo necessário para desocupar uma dada posição, levando em consideração a posição mais próxima desocupada e disponível. O tempo estimado para

desocupar a posição é válido para um único movimento, o que significa que essa operação deve ser re-executada depois de cada movimento ser executado.

É importante destacar que o procedimento executado na etapa 1b pode ser realizado com uma única execução do algoritmo de Dijkstra simplesmente substuindo seu critério de parada para "pare apenas quando todos os vértices forem visitados". Portanto, se $|\mathcal{F}|$ posições desocupadas estão disponíveis, então o algoritmo de Dijkstra é executado $|\mathcal{F}|$ vezes. Em acréscimo, as posições desocupadas e bloqueadas têm $tempo_{LIMPEZA}$ igual a 0 e infinito, respectivamente.

Os procedimentos *ReelFastestPath*, correspondente à uma variação do algoritmo de Dijkstra, e *PositionClearTimeEstimation* serão utilizados pelas heurísticas propostas neste trabalho para a otimização do planejamento do deslocamento de bobinas.

5 Heurísticas Propostas

São propostas duas estratégias de otimização para o planejamento do deslocamento de bobinas em uma planta de produção de tubos flexíveis. A primeira é dedicada ao planejamento diário e pode ser executada com limite de quatro horas, enquanto a segunda, voltada para o replanejamento, deve ser finalizada em pouco minutos e é utilizada quando é necessário ajustar repentinamente o planejamento devido à algum imprevisto. Ambas estratégias compartilham componentes que foram concebidos para explorar as característica principais do problema, usando os cálculos de distância como definido no Capítulo 4. Por essa razão este capítulo primeiramente apresenta os componentes compartilhados na seção 5.1, seguidos pelas heurísticas para planejamento na seção 5.2 e pelas heurísticas para replanejamento na seção 5.3. A síntese das heurísticas propostas é apresentada ao fim do capítulo na seção 5.4.

5.1 Componentes Compartilhados

Os componentes definidos para explorar as características principais do problema são divididos em quatro classes principais: Operadores Básicos, Construção da Solução Inicial, Vizinhanças de Busca Local Rápidas, e Vizinhanças de Busca Local Caras. Estes componentes são apresentados nesta seção e utilizados pelos algoritmos das heurísticas de planejamento e replanejamento.

5.1.1 Operadores Básicos

Os Operadores Básicos são empregados para construir um plano de movimentação das bobinas capaz de completar uma dada subtarefa. A partir das posições de origem e destino da subtarefa e da alocação atual da fábrica, define-se o caminho mínimo da bobina localizada na posição de origem até o destino, incluindo a trajetória de outras bobinas para liberar o caminho mínimo. Três operadores propostos, nomeados *BuildMoveListA*, *BuildMoveListB*, e *BuildMoveListC*, utilizam o procedimento *ReelFastestPath* (apresentado no Capítulo 4) e complementam o resultado com os movimentos necessários para limpar o caminho mínimo, com pequenas variações entre si.

5.1.1.1 BuildMoveListA

O primeiro operador gerador de movimentos, nomeado *BuildMoveListA*, considera que o caminho mínimo deve ser executado em uma única operação. Por essa razão, todos os movimentos de limpeza são executados primeiramente para permitir, em seguida, o deslocamento da bobina alvo da posição de origem até a posição de destino. Após obter a lista completa de

submovimentos, este operador executa duas heurísticas na tentativa de reorganizar a lista e reduzir a quantidade total de operações da subtarefa. Estas etapas são descritas a seguir:

BuildMoveListA

- Dado B o identificador da bobina a ser movimentada, O o identificador da posição corrente de B, e $D \neq O$ o identificador da posição de máquina que deve receber a bobina a fim de permitir o processamento da bobina.

1. Estimar o caminho mínimo entre O e D usando o procedimento *ReelFastestPath*.
2. Baseado no caminho mínimo estimado no passo 1, identificar as posições que devem ser desocupadas antes de mover B de O para D. Ordenar essa lista em ordem crescente de distância até D.
3. Definir a lista de operações pendentes como vazia.
4. Enquanto a lista de posições a desocupar não é vazia:
   a) Definir P_1 como a primeira posição da lista de posições a serem desocupadas.
   b) Identificar a posição P_2 que receberá a bobina da posição P_1 de forma que o custo de mover a bobina para P_2 é mínimo (usar *ReelFastestPath* para avaliar esse custo). Neste passo não é possível desocupar o caminho entre P_1 e P_2, podendo, no máximo, P_2 estar ocupada.
   c) Se P_2 está ocupada, então adicionar P_2 na lista de posições a desocupar e incluir $P_1 \rightarrow P_2$ na lista de operações pendentes.
   d) Caso contrário, mover a bobina de P_1 para P_2, executar todas as operações pendentes, e atualizar a lista de posições a desocupar.
5. Mover B de O para D.
6. Definir a *flag* de aprimoramento como verdadeira.
7. Enquanto a *flag* de aprimoramento for verdadeira:
   a) Executar heurística para operação de comprimir.
   b) Executar heurística para operação de adiantar/atrasar/comprimir.
   c) Se a solução não melhorou, definir a *flag* de aprimoramento como falsa.
8. Atribuir janela de tempo para cada operação baseando nas disponiblidades de posição, bobina e U.M.

Exemplo 4 *Para ilustrar o procedimento descrito acima e as duas heurísticas de aprimoramento nos passos 7a e 7b considere a fábrica na condição da Figura 4. Assuma que a subtarefa a ser processada por essas heurísticas consiste em mover a bobina* $B4$*, previamente na posição* $4|A(9,17)$*, para posição* $50|M(2,3)$*.*

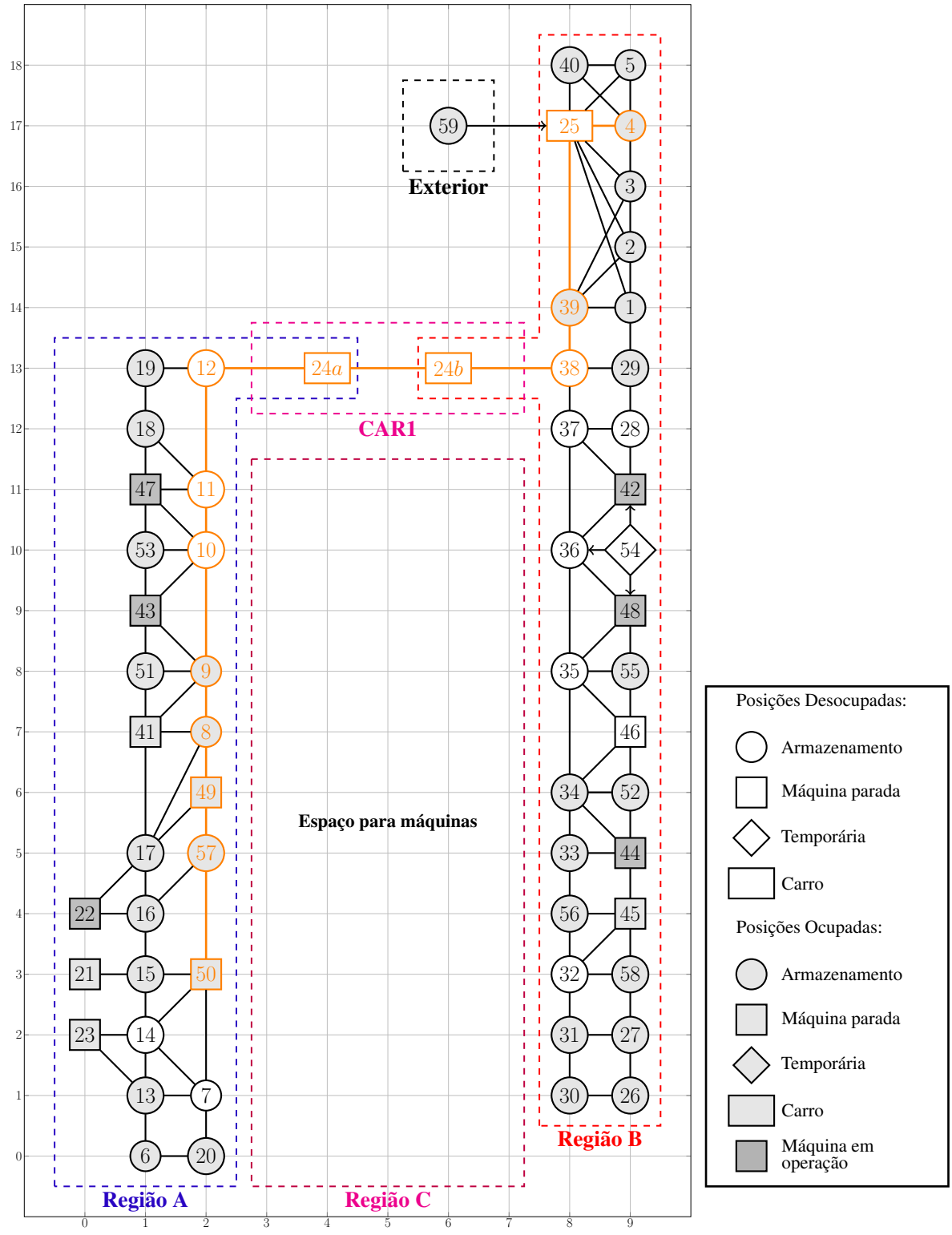


Figura 4 – Caminho mínimo conforme Exemplo 4

Nesse caso o operador BuildMoveListA funcionaria da seguinte maneira:

- *No passo 1, é identificado que o caminho mínimo entre* $4|A(9,17)$ *e* $50|M(2,3)$ *é* $4|A(9,17) \rightarrow 25|C(8,17) \rightarrow 39|A(8,14) \rightarrow 38|A(8,13) \rightarrow 24b|C(6,13) \rightarrow 24a|C(4,13) \rightarrow 12|A(2,13) \rightarrow 11|A(2,11) \rightarrow 10|A(2,10) \rightarrow 9|A(2,8) \rightarrow 8|A(2,7) \rightarrow 49|M(2,6) \rightarrow 57|A(2,5) \rightarrow 50|M(2,3)$. *Essa rota mínima é representada em laranja na Figura 4.*

- *As posições a serem desocupadas são* $50|M(2,3)$, $57|A(2,5)$, $49|M(2,6)$, $8|A(2,7)$, $9|A(2,8)$, *e* $39|A(8,14)$, *nessa ordem (passo 2).*

- *Os passos 4 e 5 resultam em 14 operações, que são apresentadas na Tabela 5.*

Tabela 5 – Exemplo de execução do operador *BuildMoveListA* – passos 1 até 5.

operação	bobina	posição inicial	posição final	operação	bobina	posição inicial	posição final
1	$B36$	$50\|M(2,3)$	$7\|A(2,1)$	2	$B42$	$57\|A(2,5)$	$14\|A(1,2)$
3	$B8$	$9\|A(2,8)$	$37\|A(8,12)$	4	$B7$	$8\|A(2,7)$	$9\|A(2,8)$
5	$B35$	$49\|A(2,6)$	$8\|A(2,7)$	6	$B8$	$37\|A(8,12)$	$28\|A(9,12)$
7	$B7$	$9\|A(2,8)$	$37\|A(8,12)$	8	$B35$	$8\|A(2,7)$	$9\|A(2,8)$
9	$B7$	$37\|A(8,12)$	$36\|A(8,10)$	10	$B35$	$9\|A(2,8)$	$37\|A(8,12)$
11	$B7$	$36\|A(8,10)$	$35\|A(8,8)$	12	$B35$	$37\|A(8,12)$	$36\|A(8,10)$
13	$B26$	$39\|A(8,14)$	$37\|A(8,12)$	14	$B4$	$4\|A(9,17)$	$50\|M(2,3)$

A partir da Tabela 5, é possível notar que o conjunto de operações obtido pode ser reduzido agrupando algumas operações. Esse é o principal papel da duas heurísticas de aprimoramento propostas:

- *A heurística para operação de comprimir avalia se duas operações de uma dada bobina podem ser agrupadas em uma única sem bloquear as operações que ocorrem entre elas. Por exemplo, as operações 3 e 6 (na Tabela 5) podem ser comprimidas sem afetar as operações 4 e 5.*

- *Em muitos casos, não é possível comprimir diretamente duas operações da mesma bobina em uma única porque a operação resultante bloquearia as operações que acontecem entre elas. A heurística para operação de adiantar/atrasar/comprimir tenta contornar essa situação experimentando avançar ou atrasar as operações entre aquelas que seriam comprimidas. Por exemplo, as operações 4 e 7 podem ser comprimidas se a operação 6 avançar para antes da 4.*

A execução completa do ciclo no passo 7, para as operações da Tabela 5, reduz a solução de 14 para 7 operações, como apresentado na Tabela 6. É possível notar que nenhuma bobina é submetida a mais de uma operação nessa tabela.

Por fim, uma janela de tempo é atribuida para cada operação no passo 8. Essas janelas de tempo são estimadas baseado na disponibilidade das bobinas, posições e U.M.s, seguindo a filosofia do "quanto antes". Essa escolha é justificada pela relevância de minimizar o atraso e maximizar o adiantamento no problema.

Tabela 6 – Exemplo de execução do operador *BuildMoveListA* – depois do passo 7.

operação	bobina	posição inicial	posição final	operação	bobina	posição inicial	posição final
1	$B36$	$50\|M(2,3)$	$7\|A(2,1)$	2	$B42$	$57\|A(2,5)$	$14\|A(1,2)$
3	$B8$	$9\|A(2,8)$	$28\|A(9,12)$	4	$B7$	$8\|A(2,7)$	$35\|A(8,8)$
5	$B35$	$49\|A(2,6)$	$36\|A(8,10)$	6	$B26$	$39\|A(8,14)$	$37\|A(8,12)$
7	$B4$	$4\|A(9,17)$	$50\|M(2,3)$				

5.1.1.2 BuildMoveListB e BuildMoveListC

O operador *BuildMoveListA* assume que o movimento da bobina alvo para a posição de destino em uma subtarefa é executado em uma única operação (por exemplo, o movimento da bobina $B4$ para posição $50|M(2,3)$ na Tabela 6). Portanto, para uma dada subtarefa, o operador gera algumas operações para desocupar o caminho entre a bobina e a posição de destino e, quando o caminho está livre, uma operação final é executada para completar a subtarefa. Essa suposição pode levar a situações infactíveis em alguns casos, quando o caminho mínimo é muito longo e/ou a fábrica está com muitas posições ocupadas naquele momento, como apresentado no próximo exemplo.

Os operadores *BuildMoveListB* e *BuildMoveListC* foram propostos para contornar essa limitação. Utilizando diferentes estratégias, os operadores quebram a operação final da subtarefa em duas operações, de maneira que apenas parte das posições do caminho mínimo entre O e D devem se manter desocupadas durante cada passo. A ação de quebrar a operação final é realizada múltiplas vezes, a fim de encontrar aquela com melhor resultado. Em *BuildMoveListB* para cada posição intermediária do caminho mínimo, a operação final é dividida em duas. Enquanto *BuildMoveListC* quebra a operação principal em duas utilizando posições de fora do caminho mínimo.

Exemplo 5 *Considerando o diagrama da Figura 5, assuma que a bobina $B22$, que está na posição $30|A(8,1)$, deve ser deslocada para a posição de máquina $50|M(2,3)$. O menor caminho entre $30|A(8,1)$ e $50|M(2,3)$ é $30|A(8,1) \rightarrow 31|A(8,2) \rightarrow 32|A(8,3) \rightarrow 56|A(8,4) \rightarrow 33|A(8,5) \rightarrow 34|A(8,6) \rightarrow 35|A(8,8) \rightarrow 36|A(8,10) \rightarrow 37|A(8,12) \rightarrow 38|A(8,13) \rightarrow 24a|C(6,13) \rightarrow 24b|C(8,13) \rightarrow 12|A(2,13) \rightarrow 11|A(2,11) \rightarrow 10|A(2,10) \rightarrow 9|A(2,8) \rightarrow 8|A(2,7) \rightarrow 49|A(2,6) \rightarrow 57|A(2,5) \rightarrow 50|A(2,3)$ (representado em laranja).*

Levando em consideração que as posições no caminho devem permanecer livres para a operação final da subtarefa, conforme BuildMoveListA, é possível verificar que a operação direta $30|A(8,1) \rightarrow 50|M(2,3)$ é infactível: nove posições devem ser desocupadas ($31|A(8,2)$,

$56|A(8,4)$, $33|A(8,5)$, $34|A(8,6)$, $9|A(2,8)$, $8|A(2,7)$, $49|A(2,6)$, $57|A(2,5)$, *e* $50|A(2,3)$*) e existem apenas quatro posições permitidas e desocupadas fora desse caminho (*$46|M(9,7)$, $28|A(9,12)$, $14|A(1,2)$, *e* $7|A(3,1)$*).*

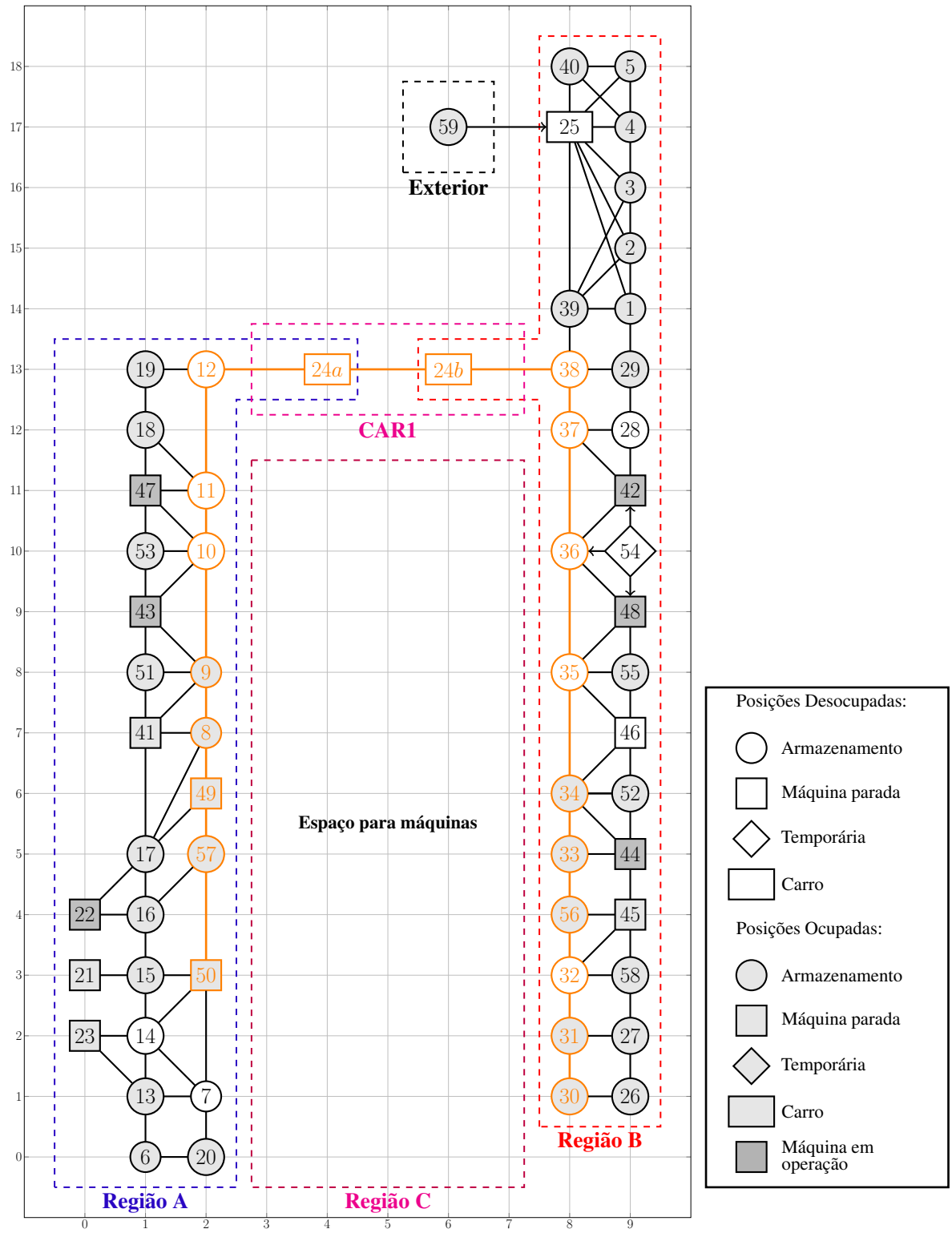


Figura 5 – Caminho mínimo conforme Exemplo 5

As seguintes etapas são necessárias para executar *BuildMoveListB*:

BuildMoveListB

1. Definir a melhor solução corrente como vazia e o melhor tempo para completar a subtarefa corrente como infinito.
2. Estimar o caminho mínimo entre O e D usando *ReelFastestPath*.
3. Para cada posição I entre a segunda e a penúltima posições do caminho mínimo (excluindo as posições CAR1 e CAR2):
   a) Usar *BuildMoveListA* para planejar o movimento de B da posição O para I e construir o conjunto $\mathcal{P}_1$ com as operações resultantes.
   b) Usar *BuildMoveListA* para planejar o movimento de B da posição I para D e construir o conjunto $\mathcal{P}_2$ com as operações resultantes.
   c) Combinar $\mathcal{P}_1$ e $\mathcal{P}_2$ em $\mathcal{P}$.
   d) Se o tempo para completar a subtarefa utilizando $\mathcal{P}$ for melhor do que a melhor solução corrente, então atualizar a melhor solução corrente e o melhor tempo para completar a subtarefa.

O operador *BuildMoveListC* segue uma estratégia similar e com passos semelhantes a *BuildMoveListB*, exceto pela tentativa de quebrar a operação principal utilizando posições de fora do caminho mínimo entre O e D (veja exemplo abaixo).

Como *BuildMoveListA* é executado múltiplas vezes dentro de *BuildMoveListB* e *BuildMoveListC* (2 vezes por posição intermediária ou fora do caminho mínimo), esses operadores consomem muito mais tempo do que *BuildMoveListA*. Além disso, *BuildMoveListC* tende a precisar de mais tempo do que *BuildMoveListB* dado que o número de posições fora do caminho mínimo tendem a ser maior do que o número de posições ao longo do caminho.

Exemplo 6 *O operador BuildMoveListC pode completar a subtarefa citada acima (mover a bobina* $B22$ *para posição* $50|M(2,3)$*), como apresentado na Tabela 7. Nessa solução, a posição* $28|A(9,12)$ *é usada para quebrar a operação principal: na operação 5,* $B31$ *é deslocada para posição* $28|A(9,12)$ *e, na operação 14,* $B31$ *é finalmente movida para a posição* $50|M(2,3)$*. As operações 1–4 e 6–13 são necessárias para desocupar os caminhos* $30|A(8,1) \rightarrow 31|A(8,2) \rightarrow 32|A(8,3) \rightarrow 56|A(8,4) \rightarrow 33|A(8,5) \rightarrow 34|A(8,6) \rightarrow 35|A(8,8) \rightarrow 36|A(8,10) \rightarrow 37|A(8,12) \rightarrow 28|A(9,12)$ *e* $28|A(9,12) \rightarrow 37|A(8,12) \rightarrow 38|A(8,13) \rightarrow 24a|C(6,13) \rightarrow 24b|C(8,13) \rightarrow 12|A(2,13) \rightarrow 11|A(2,11) \rightarrow 10|A(2,10) \rightarrow 9|A(2,8) \rightarrow 8|A(2,7) \rightarrow 49|A(2,6) \rightarrow 57|A(2,5) \rightarrow 50|A(2,3)$*, respectivamente.*

Tabela 7 – Exemplo de execução de *BuildMoveListC*.

operação	bobina	posição inicial	posição final	operação	bobina	posição inicial	posição final
1	$B25$	$34\|A(8,6)$	$46\|M(9,7)$	2	$B24$	$33\|A(8,5)$	$11\|A(2,11)$
3	$B41$	$56\|A(8,4)$	$12\|A(2,13)$	4	$B23$	$31\|A(8,2)$	$38\|A(8,13)$
5	$\mathbf{B22}$	$\mathbf{30}\|\mathbf{A}(\mathbf{8},\mathbf{1})$	$\mathbf{28}\|\mathbf{A}(\mathbf{9},\mathbf{12})$	6	$B36$	$50\|M(2,3)$	$7\|A(2,1)$
7	$B42$	$57\|A(2,5)$	$14\|A(1,2)$	8	$B23$	$38\|A(8,13)$	$32\|A(8,3)$
9	$B41$	$12\|A(2,13)$	$56\|A(8,4)$	10	$B24$	$11\|A(2,11)$	$33\|A(8,5)$
11	$B8$	$9\|A(2,8)$	$34\|A(8,6)$	12	$B7$	$8\|A(2,7)$	$35\|A(8,8)$
13	$B35$	$49\|M(2,6)$	$36\|A(8,10)$	**14**	$\mathbf{B22}$	$\mathbf{28}\|\mathbf{A}(\mathbf{9},\mathbf{12})$	$\mathbf{50}\|\mathbf{M}(\mathbf{2},\mathbf{3})$

5.1.2 Construção da Solução Inicial

A fim de gerar uma solução inicial para o plano de produção, a etapa de Construção da Solução Inicial utiliza os Operadores Básicos para cada uma das subtarefas contidas no plano e, ao final, concatena as operações resultantes. O principal objetivo deste componente é gerar rapidamente uma solução factível e razoavelmente boa (já que ela foi gerada a partir de caminhos mínimos), que poderá ser aprimorada pelos componentes subsequentes.

As subtarefas do plano de produção podem ser organizadas em múltiplas ordenações para construção da solução inicial e por isso, essa é uma variável de entrada do componente de construção da solução inicial. São propostas três regras para gerar a sequência de subtarefas:

Sequência A: As tarefas são ordenadas baseadas no seu tempo de início. Para cada tarefa, as duas subtarefas correspondentes são ordenadas baseando-se na disponibilidade da bobina/posição no instante zero.

Sequência B: Todas as subtarefas são ordenadas baseando-se na disponibilidade da bobina/posição no instante zero.

Sequência C: Assim como na Sequência A, as tarefas são ordenadas baseadas no seu tempo de início, mas nessa regra, para cada tarefa, a segunda subtarefa sempre é realizada antes da primeira. Essa sequência foi gerada após varias análises experimentais, que mostraram que na maior parte dos casos na Sequência A a primeira subtarefa é posicionada antes da segunda.

O componente apresentado também recebe como variável de entrada a ordem em que os operadores básicos são organizados. Essa ordenação é definida a partir de dois operadores: *BuildMoveListCombinedA* e *BuildMoveListCombinedB*.

A fim de obter a solução para cada subtarefa, *BuildMoveListCombinedA* executa primeiramente *BuildMoveListA*. Caso não seja possível encontrar uma solução, a ação é repetida utilizando *BuildMoveListB* e o mesmo acontece para *BuildMoveListC*, assim como apresentado nas etapas abaixo:

BuildMoveListCombinedA

- Dada a subtarefa corrente a ser realizada:

1. Definir as posições de carro como indisponíveis.
2. Tentar completar a subtarefa usando *BuildMoveListA*.
3. Se a subtarefa não foi concluida, tentar completar a subtarefa corrente usando *BuildMoveListB*.
4. Se a subtarefa não foi concluída, tentar completar a subtarefa corrente usando *BuildMoveListC*.
5. Se a subtarefa não foi concluída:
   a) definir as posições de carro como disponíveis, e
   b) tentar completar a subtarefa corrente usando *BuildMoveListA*.
6. Se a subtarefa não foi concluida, tentar completar a subtarefa corrente usando *BuildMoveListB*.
7. Se a subtarefa não foi concluída, tentar completar a subtarefa corrente usando *BuildMoveListC*.

BuildMoveListCombinedB segue a mesma estratégia de *BuildMoveListCombinedA*, apenas invertendo a ordem entre *BuildMoveListA* e *BuildMoveListB*.

A priorização de *BuildMoveListA* em relação a *BuildMoveListB*, e *BuildMoveListB* em relação a *BuildMoveListC* é justificada principalmente pelo tempo necessário para a execução desses operadores. Dado que o tempo de execução é um aspecto crítico para o problema deste trabalho, dar prioridade para os operadores menos caros é recomendado.

Apesar de ser uma opção mais cara, *BuildMoveListCombinedB* gera soluções mais dinâmicas para manipulações posteriores, uma vez que *BuildMoveListB* é utilizado prioritariamente. Como *BuildMoveListB* divide a operação do caminho mínimo em duas, os próximos componentes tem uma maior quantidade de operações para tentar reorganizar e aprimorar.

Outra característica interessante dos procedimentos mencionados acima é como eles lidam com os carros. Como discutido no Capítulo 3, operações cujo destino final é um carro devem ser evitadas uma vez que, nessas operações, a bobina pode ficar exposta a condições climáticas adversas. Seguindo esse princípio, os operadores *BuildMoveListCombined* foram

elaborados de forma que operações cujo destino final é um carro são permitidas apenas se não é possível completar a subtarefa sem usar os carros com o propósito de armazenamento.

A fim de gerar uma solução inicial completa para o plano de produção, os operadores *BuildMoveListCombined* são utilizados em todas as subtarefas, seguindo a sequência pré-estabelecida, conforme o procedimento descrito abaixo:

BuildInitialSolution

- Dado uma lista com a ordem em que as subtarefas devem ser avaliadas e qual operador *BuildMoveListCombined* deve ser utilizado (A ou B):

1. Definir a *flag* de parada como falsa.
2. Enquanto *flag* de parada for falsa:
   a) Aplicar o operador *BuildMoveListCombined* informado para executar a subtarefa corrente.
   b) Se *BuildMoveListCombined* obteve uma solução viável, então passar para a próxima subtarefa, considerando a nova configuração da planta, a partir dos movimentos gerados para a subtarefa anterior.
   c) Caso contrário, definir a *flag* de parada como verdadeira.
   d) Se todas as subtarefas estão concluídas, então definir a *flag* de parada como verdadeira.
3. Avaliar a solução gerada usando as funções F_1 até F_5.

5.1.3 Vizinhanças de Busca Local Rápidas

Dois conjuntos de vizinhanças de busca local são propostos neste trabalho. O primeiro conjunto, denominado Vizinhanças de Busca Local Rápidas (ou simplesmente FLSN - *Fast Local Search Neighborhood*), é abordado nesta subseção. As FLSN visam aprimorar a solução inicial encontrada pelo componente de Construção da Solução Inicial utilizando métodos de baixo custo computacional que executam pequenas alterações na solução corrente para explorar novas possibilidades de solução, conforme apresentado a seguir:

PreMoveUnfeasibleSolutions: eventualmente, os operadores podem levar a soluções em que algumas das subtarefas não estão completas e a solução inicial é infactível. O objetivo desta vizinhança é aprimorar a factibilidade da solução, por meio dos seguintes passos:

- Dada uma solução e uma subtarefa que não está concluída na solução fornecida:

1. Identificar a última operação da bobina envolvida na subtarefa incompleta.
2. Cancelar todas operações depois daquela identificada no passo 1, e ela inclusive.
3. Tentar mover a bobina diretamente para a posição de destino da subtarefa (ou para uma posição próxima caso o destino esteja bloqueado ou seja inalcançável nesse momento).
4. Tentar executar as subtarefas pendentes usando o operador *BuildMoveListCombined*.

TaskCompression: essa vizinhança é similar à heurística de compressão empregada no operador *BuildMoveListA*, cujo objetivo é avaliar se duas operações de uma dada bobina podem ser agrupadas em uma única sem bloquear as operações que ocorrem entre elas. As principais diferenças entre os dois métodos são: (i) a vizinhança *TaskCompression* considera todas as operações da solução em vez das operações de uma única subtarefa e (ii) a decisão entre soluções na vizinhança *TaskCompression* é tomada baseada na função lexicográfica definida pelos objetivos F_1 a F_5 (a nova solução só é mantida caso o seu valor objetivo seja melhor do que o da solução incumbente) em vez de sempre manter o novo resultado.

Advance/Postpone/Compression: essa vizinhança é similar à heurística para operação de adiantar/atrasar/comprimir usada no operador *BuildMoveListA*, cujo objetivo é agrupar operações de uma mesma bobina a partir do adiantamento ou atraso de outras operações. As principais diferenças entre a vizinhança e a heurística são as mesmas observadas para a vizinhança *TaskCompression*: ela é aplicada a soluções completas e toma decisões baseada na função lexicográfica.

Essas três vizinhanças são combinadas em um único operador de busca local, que funciona como apresentado a seguir:

FLSN

- Dada uma solução inicial:

1. Se a solução é infactível:
   a) Definir a *flag* de aprimoramento de factibilidade como verdadeira.
   b) Enquanto a *flag* de aprimoramento de factibilidade for verdadeira:
      i. Aplicar a busca pela vizinhança *PreMoveUnfeasibleSolutions* para aprimorar a factibilidade.
      ii. Se a factibilidade não foi aperfeiçoada na iteração corrente da vizinhança ou se a nova solução é factível, então definir a *flag* de aprimoramento de factibilidade como falsa.

2. Definir a *flag* de aprimoramento da solução como verdadeira.
3. Enquanto a *flag* de aprimoramento da solução for verdadeira:
    a) Definir a *flag* de aprimoramento da compressão como verdadeira.
    b) Enquanto a *flag* de aprimoramento da compressão for verdadeira:
        i. Aplicar a busca pela vizinhança *TaskCompression*.
        ii. Se a solução não foi aperfeiçoada na iteração corrente da vizinhança, então definir a *flag* de aprimoramento da compressão como falsa.
    c) Definir a *flag* de aprimoramento de avanço/atraso como verdadeira.
    d) Enquanto a *flag* de aprimoramento de avanço/atraso for verdadeira:
        i. Aplicar a busca pela vizinhança *Advance/Postpone/Compression*.
        ii. Se a solução não foi aperfeiçoada na iteração corrente da vizinhança, então definir a *flag* de avanço/atraso como falsa.
    e) Se a solução não foi aperfeiçoada na iteração corrente da busca local, então definir a *flag* de aprimoramento da solução como falsa.

5.1.4 Vizinhanças de Busca Local Caras

O segundo conjunto de vizinhanças de busca local, denominado Vizinhanças de Busca Local Caras (ou ELSN - *Expensive Local Search Neighborhood*), é abordado nesta subseção. As ELSN visam aprimorar a solução encontrada por FLSN utilizando até sete vizinhanças que exploram o espaço de busca de forma mais ampla e demandam mais tempo para serem processadas.

Os métodos incluem tentativas diversas de modificar ou gerar uma nova solução com regras obtidas a partir do conhecimento gerado pela descrição do problema. Alguns métodos têm uma abordagem mais agressiva que outros e/ou são adequados para diferentes tipos de heurísticas. Por essa razão, é importante notar as características de cada um no momento de selecionar e ordenar as vizinhanças para construção do algoritmo, a fim de obter uma boa solução final e no tempo desejado, de acordo com o propósito do algoritmo.

As sete vizinhanças são apresentadas a seguir e é possível notar que, no geral, cada solução candidata intermediária, uma vez obtida, é submetida a FLSN. Isso significa que as buscas locais rápidas são executadas várias vezes e explica, portanto, a diferença de nomenclatura entre as vizinhanças rápidas e caras.

SubtaskSequence: a vizinhança *SubtaskSequence* tenta aprimorar o valor objetivo da solução incumbente trocando a sequência em que as subtarefas são avaliadas e, em seguida, gerando uma nova solução. A sequência considerada é extraída a partir da lista de movimentos da solução incumbente e ela não é necessariamente a mesma do componente de Construção de Solução Inicial, uma vez que FLSN pode alterar a ordem das operações.

A lógica desta vizinhança é explorar pequenas modificações na sequência que, por si só, não geram alternativas para a construção da solução inicial. São propostos três métodos capazes de alterar a sequência das subtarefas (*SubtaskSequenceA*, *SubtaskSequenceB* e *SubtaskSequenceC*), com diferentes regras e quantidade de trocas executadas, conforme apresentado a seguir:

SubtaskSequenceA

1. Salvar a solução incumbente como a melhor solução.
2. Para cada subtarefa i na sequência de subtarefas corrente:
   a) Trocar as subtarefas i e $i+1$ na sequência de subtarefas corrente, caso as bobinas das subtarefas sejam diferentes.
   b) Para essa nova sequência, construir uma nova solução utilizando o mesmo procedimento de construção de solução descrito na subseção 5.1.2.
   c) Tentar aperfeiçoar a nova solução usando *FLSN*.
   d) Atualizar a melhor solução, caso o resultado seja superior.

SubtaskSequenceB

1. Salvar a solução incumbente como a melhor solução.
2. Para cada subtarefa i na sequência de subtarefas corrente:
   a) Mover a subtarefa i para a posição $i+2$ na sequência de subtarefas corrente, caso as bobinas das subtarefas sejam diferentes.
   b) Para essa nova sequência, construir uma nova solução utilizando o mesmo procedimento de construção de solução descrito na subseção 5.1.2.
   c) Tentar aperfeiçoar a nova solução usando *FLSN*.
   d) Atualizar a melhor solução, caso o resultado seja superior.

SubtaskSequenceC

1. Salvar a solução incumbente como a melhor solução.
2. Para cada subtarefa i na sequência de subtarefas corrente:
3. Para cada subtarefa j na sequência de $i + 1$ até $i + 5$:
   a) Trocar as subtarefas i e j, caso as bobinas das subtarefas sejam diferentes.
   b) Para essa nova sequência, construir uma nova solução utilizando o mesmo procedimento de construção de solução descrito na subseção 5.1.2.
   c) Tentar aperfeiçoar a nova solução usando *FLSN*.
   d) Atualizar a melhor solução, caso o resultado seja superior.

Considerando uma sequência com 20 subtarefas, *SubtaskSequenceA* realiza aproximadamente 19 trocas (essa quantidade pode alterar de acordo com as bobinas de cada subtarefa) e cada troca executa FLSN uma vez. *SubtaskSequenceB*, por sua vez, realiza aproximadamente 18 movimentações e *SubtaskSequenceC* 85 trocas. A vizinhança pode ser composta pelos métodos apresentados, em qualquer quantidade e organização, conforme a relação custo benefício almejada. Os métodos escolhidos são executados em sequência.

OperationSplit: a vizinhança *OperationSplit* divide operações para identificar soluções potencialmente melhores. Essa estratégia pode ser interessante nos casos em que alguma operação é atrasada porque a posição de destino está bloqueada durante algum intervalo de tempo. Assim, quando a operação é quebrada em duas ou mais, é possível antecipar alguns submovimentos e, eventualmente, adiantar a subtarefa corrente ou outras.

São propostos dois métodos não complementares para divisão das operações (*OperationSplitA* e *OperationSplitB*). A depender do algoritmo, é possível escolher entre uma versão em que todas as operações são quebradas simultaneamente, gerando uma única nova solução ao final, ou uma versão que realiza apenas uma divisão por vez, gerando múltiplas soluções a serem comparadas, como apresentado a seguir:

OperationSplitA

1. Para cada operação i na solução corrente:
   a) Cada submovimento j da operação i se torna uma operação, com exceção dos submovimentos que envolvem movimentação para os carros.
2. Tentar aperfeiçoar a solução usando *FLSN*.

3. A nova solução é mantida independentemente de aperfeiçoar ou não a solução inicial.

OperationSplitB

1. Para cada operação i na solução corrente:
   a) Para cada submovimento j da operação i, começando do segundo:
      i. Dividir a operação i em duas, com a primeira incluindo os submovimentos 1 até $j-1$ e, na segunda, os submovimentos j até o último.
      ii. Tentar aperfeiçoar a solução usando *FLSN*.
      iii. Atualizar a melhor solução, caso o resultado seja superior.

OperationSplitA é especialmente interessante para ser aplicado quando a solução incumbente pode ter valores ruins. Ao dividir todas as operações e manter essa nova solução com mais operações, *OperationSplitA* gera uma maior capacidade de aperfeiçoamento para as vizinhanças seguintes. *OperationSplitB*, ao contrário, é sugerido quando não se deseja perder os resultados da solução incumbente, que normalmente já tem valores bons.

RandomSearch: apesar de a ELSN conter outras seis vizinhanças, não é possível garantir que o espaço de busca está sendo totalmente explorado. *RandomSearch* é uma vizinhança estocástica que realiza modificações na solução corrente que potencialmente não seriam abordadas pelas regras das demais vizinhanças.

São propostos dois métodos não complementares para esta vizinhança (*RandomSearchA* e *RandomSearchB*). Ambos são executados durante um tempo pré-definido fornecido como parâmetro mas, a depender do algoritmo, é possível escolher entre uma versão que só fornece novos resultados caso a solução incumbente seja aperfeiçoada ou uma versão que sempre fornece o melhor resultado da vizinhança, independentemente de ser melhor ou pior que a solução incumbente. Os dois métodos são apresentados a seguir:

RandomSearchA

1. Salvar a solução incumbente como a melhor solução.
2. Dada a lista de submovimentos correspondente ao plano de produção completo e um tempo de execução, enquanto o tempo de execução não for alcançado:
   a) Sortear um número i limitado ao número de submovimentos da lista.

b) Cortar a lista na posição i. Esta posição pode estar contida em uma operação ou movimento.

c) A solução é mantida do primeiro submovimento até o submovimento i.

d) Gerar os submovimentos necessários para concluir a subtarefa do submovimento i.

e) Tentar executar as subtarefas pendentes usando o operador *BuildMoveListCombined*.

f) Tentar aprimorar essa nova solução usando *FLSN*.

g) Atualizar a melhor solução, caso o resultado seja superior.

RandomSearchB

1. Definir a *flag* de primeira solução como falsa.
2. Dada a lista de submovimentos correspondente ao plano de produção completo e um tempo de execução. Enquanto o tempo de execução não for alcançado:
   a) Sortear um número i limitado ao número de submovimentos da lista.
   b) Cortar a lista na posição i. Esta posição pode estar contida em uma operação ou movimento.
   c) A solução é mantida do primeiro submovimento até o submovimento i.
   d) Gerar os submovimentos necessários para concluir a subtarefa do submovimento i.
   e) Tentar executar as subtarefas pendentes usando o operador *BuildMoveListCombined*.
   f) Tentar aprimorar essa nova solução usando *FLSN*.
   g) Caso a *flag* de primeira solução seja falsa: salvar a solução corrente como melhor solução.
   h) Caso contrário: atualizar a melhor solução, caso o resultado seja superior.

O tempo de execução pode ser configurado conforme as necessidades do algoritmo. *RandomSearchA* é sugerido para um algoritmo de apenas uma iteração, em que as vizinhanças só atualizam a solução corrente em caso de aprimoramento, enquanto *RandomSearchB* é recomendado para um algoritmo de múltiplas iterações, onde é necessário aplicar uma perturbação na solução corrente entre as iterações.

PreMoveOperation: a vizinhança *PreMoveOperation* tenta reduzir o número de operações necessárias para completar o plano de produção unindo operações da mesma bobina, com o objetivo de adiantar operações futuras e potencialmente adiantar as subtarefas correspondentes. Ela é diferente da vizinhança *Advance/Postpone/Compression* uma vez que ela reconstrói uma parte da solução incumbente ao tentar unir as operações em vez de apenas reorganizar. Em acréscimo, outra diferença é o fato das múltiplas tentativas de união gerarem várias possíveis soluções, assim como apresentado a seguir:

PreMoveOperation

1. Salvar a solução incumbente como a melhor solução.
2. Para cada operação i que não está atribuída a uma subtarefa na solução inicial:
   a) Se a bobina envolvida na operação i tem alguma operação subsequente:
      i. Identificar a posição destino da operação imediatamente subsequente com a mesma bobina.
      ii. Cancelar a operação i e todas as operações subsequentes.
      iii. Tentar mover a bobina diretamente para a posição destino identificada no passo 2(a)i.
      iv. Tentar executar as subtarefas pendentes usando o operador *BuildMoveList-Combined*.
      v. Tentar aprimorar essa nova solução usando *FLSN*.
      vi. Atualizar a melhor solução, caso o resultado seja superior.

A operação atribuída a uma subtarefa é aquela que completa a subtarefa, i.e. a operação que move a bobina correta para a posição de destino especificada. Por exemplo, na Tabela 4, as operações 7 e 8 são aquelas atribuídas às subtarefas 1-1 e 1-2, respectivamente. As outras seis operações restantes não estão atribuídas a nenhuma subtarefa uma vez que eles não levam a nenhum processamento de bobina. O passo 2, portanto, seleciona as operações que não completam a subtarefa.

PreMoveTask: assim como *PreMoveOperation*, a vizinhança *PreMoveTask* também tenta reduzir o número de operações necessárias para completar o plano de produção unindo operações da mesma bobina. A diferença está no fato de *PreMoveTask* avaliar o conjunto de operações atribuídas em vez do conjunto das desatribuídas. Dessa maneira, esta vizinhança tenta antecipar diretamente a operação de conclusão da subtarefa unindo com uma outra operação prévia da mesma bobina, com o objetivo de adiantar a subtarefa, como apresentado a seguir:

PreMoveTask

1. Salvar a solução incumbente como a melhor solução.
2. Para cada operação i que está atribuída a uma subtarefa na solução inicial:
   a) Se a bobina envolvida na operação i tem alguma operação anterior:
      i. Cancelar a operação imediatamente anterior e também todas as operações posteriores.
      ii. Tentar mover a bobina diretamente para a posição destino especificada na subtarefa atribuída (ou para uma posição perto se o destino está bloqueado ou inalcançável no momento).
      iii. Tentar executar as subtarefas pendentes usando o operador *BuildMoveList-Combined*.
      iv. Tentar aperfeiçoar a nova solução usando *FLSN*.
      v. Atualizar a melhor solução, caso o resultado seja superior.

AnticipateSubtask: a vizinhança *AnticipateSubtTask* tenta antecipar subtarefas atrasadas, na intenção de encontrar algum momento em que seria possível encaixá-la, diminuindo seu atraso, sem piorar globalmente a solução. Esse adiantamento é realizado da seguinte maneira:

AnticipateSubtask

1. Salvar a solução incumbente como a melhor solução.
2. Para cada subtarefa i atrasada na solução incumbente:
3. Para cada subtarefa j adiantada e que termina antes de i:
   a) Caso a bobina de j seja diferente da bobina de i, cancelar todas as operações de j e também todas as operações posteriores.
   b) Tentar concluir a subtarefa i logo após as operações remanescentes da solução incumbente.
   c) Tentar executar as subtarefas pendentes usando o operador *BuildMoveListCombined*.
   d) Tentar aprimorar a nova solução usando *FLSN*.
   e) Atualizar a melhor solução, caso o resultado seja superior.

OperationSwap: a vizinhança *OperationSwap* troca duas operações sequenciais e independentes. A ideia nesse caso com a alteração da ordenação das operações é verificar se na nova configuração FLSN consegue agrupar mais operações, gerando uma possível redução do número de operações ou até algum adiantamento na solução completa. Os passos para execução de *OperationSwap* são descritos abaixo:

OperationSwap

1. Salvar a solução incumbente como a melhor solução.
2. Para cada operação i na solução incumbente:
   a) Se possível, trocar as operações i e $i + 1$ na solução incumbente corrente.
   b) Tentar aperfeiçoar a nova solução usando *FLSN*.
   c) Atualizar a melhor solução, caso o resultado seja superior.

5.2 Heurísticas para Planejamento

São propostas duas heurísticas para planejamento que utilizam como base os componentes compartilhados. Ambas são destinadas ao planejamento diário das movimentações, com flexibilidade para o tempo de execução e com horizonte de planejamento de cinco a dez dias. A primeira, nomeada *Heurística Baseada em Busca Local com VNS para o Deslocamento de Bobinas*, é referenciada como VNS–Heur no restante do trabalho. Nessa proposta, a execução dos componentes compartilhados é repetida em múltiplas iterações conforme a metaheurística *Variable Neighborhood Seach* (VNS), como apresentado na subseção 5.2.1. Apesar do critério de parada de quatro horas para o planejamento, essa heurística tenta encontrar bons resultados mais rapidamente e, portanto, foi definido um tempo de execução fixo de uma hora, sem análise de convergência da solução. A segunda, nomeada *Heurística Baseada em Programação Linear para o Deslocamento de Bobinas*, é abreviada por ILP–Heur. É definido um tempo de execução máximo igual a quatro horas e, apesar do foco na programação linear, os componentes compartilhados também são utilizados, mas com acréscimo de uma etapa com modelos exatos, como explicado na subseção 5.2.2.

5.2.1 VNS–Heur

VNS–Heur é composta por três etapas principais que foram construídas a partir dos componentes compartilhados: geração da solução inicial, busca local e perturbação. A partir da execução das etapas de geração da solução inicial e busca local é esperado obter uma solução

factível e eficiente mas que exploram apenas determinadas regiões do espaço de variáveis de decisão. Ao alterar aleatoriamente a solução na etapa de perturbação, espera-se explorar novas regiões e, eventualmente, encontrar soluções ainda melhores. Nesta heurística o critério de parada é determinado unicamente pelo tempo de execução, sempre igual a uma hora.

5.2.1.1 Geração da Solução Inicial

O componente compartilhado de construção da solução inicial (*BuildInitialSolution*) é utilizado múltiplas vezes, na intenção de gerar múltiplas soluções iniciais. São utilizadas as três regras de geração de sequência de subtarefas da subseção 5.1.2 mas, caso alguma regra forneça uma sequência repetida, ela é descartada. Em acréscimo, para cada sequência, também são aplicados os dois operadores *BuildMoveListCombined*, resultando em até seis soluções iniciais.

O operador de busca local *FLSN*, apresentado na subseção 5.1.3, é aplicado em todas as candidatas à solução inicial no intuito de aperfeiçoá-las pela primeira vez, com métodos de baixo custo computacional. Ainda como parte da etapa de geração da solução inicial, o melhor resultado é selecionado e aplicado à vizinhança de busca local *SubtaskSequence*, que extrai a ordenação de subtarefas da solução incumbente, experimenta algumas ordenações similares e sugere uma nova solução inicial alternativa.

Como as heurísticas de planejamento têm maior flexibilidade no tempo de execução, a VNS–Heur utiliza os três métodos de *SubtaskSequence* (*SubtaskSequenceA*, *SubtaskSequenceB* e *SubtaskSequenceC*) e caso um resultado mais eficiente seja encontrado, a solução inicial é atualizada e disponibilizada para a etapa de busca local. O fluxograma completo da geração da solução inicial é apresentado na Figura 6.

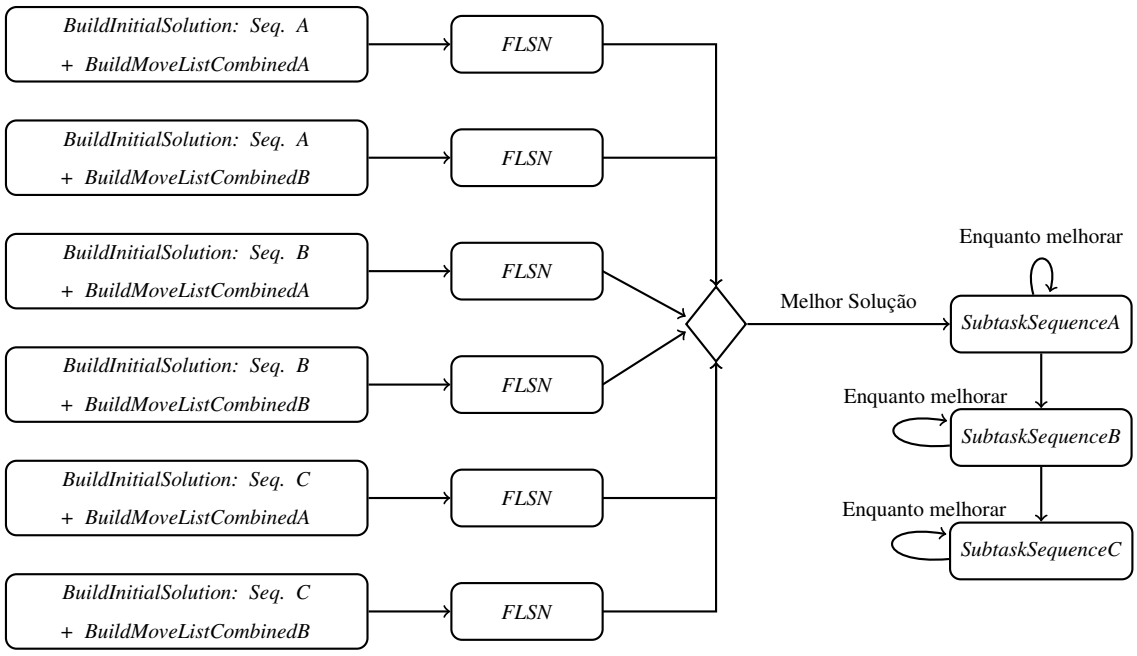


Figura 6 – Fluxograma da Geração da Solução Inicial em VNS–Heur

5.2.1.2 Busca Local

Para aperfeiçoar a solução inicial são aplicadas em sequência cinco vizinhanças de busca local apresentadas na subseção 5.1.4: *OperationSplit*, *PreMoveOperation*, *PreMoveTask*, *AnticipateSubtask* e *OperationSwap*. A sequência de vizinhanças foi definida empiricamente, experimentando a melhor ordenação em diferentes casos de teste. Vale ressaltar que *OperationSplit* é utilizada na versão A, onde todas as operações são divididas e a nova solução é sempre mantida, uma vez que a solução incumbente vindo da etapa de perturbação pode não ser tão boa. Em acréscimo, as demais vizinhanças são repetidas enquanto a nova solução for melhor do que a solução inicial.

O fluxograma que representa a etapa de busca local é apresentado na Figura 7.

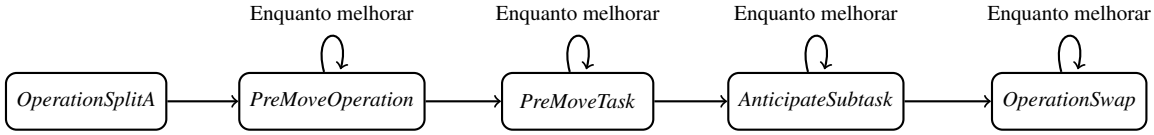


Figura 7 – Fluxograma da etapa de busca local em VNS–Heur

5.2.1.3 Perturbação

A vizinhança *RandomSearch*, apresentada na subseção 5.1.4, é aplicada como etapa de perturbação de VNS–Heur. Ela é executada durante um minuto, utilizando a variante *RandomSearchB*, em que a melhor solução da vizinhança é retornada, independentemente de ser melhor ou pior do que a solução incumbente. Em acréscimo, a solução incumbente desta vizinhança é sempre a melhor solução já encontrada para a heurística.

A etapa de perturbação é executada logo após a etapa de busca local e a nova solução é fornecida à vizinhança *OperationSplit*, retornando a execução para a busca local. Essas iterações são repetidas enquanto o critério de parada não for alcançado, como apresentado na Figura 8.

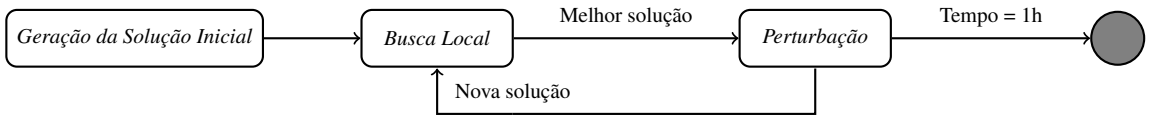


Figura 8 – Fluxograma completo de VNS–Heur

5.2.2 ILP–Heur

Com a possibilidade de propor um algoritmo que possa ser executado em algumas horas, é lógico pensar em uma abordagem exata, capaz de encontrar a solução ótima. Entretanto, o

problema de planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis tem alta dimensão, uma vez que ele é indexado por bobinas e tempo, com duração do plano de produção de até dez dias. Com essas características e dimensões, é coerente afirmar que apenas um modelo exato não é capaz de encontrar o ótimo global em algumas horas.

Uma opção para incluir uma abordagem exata é aplicar um modelo de programação linear em apenas uma parte do algoritmo. De acordo com esta ideia a heurística ILP–Heur é proposta, misturando procedimentos desenvolvidos a partir de programação linear com vizinhanças de busca local. A heurística é dividida em duas etapas principais: geração da solução inicial e busca local; e a abordagem exata é aplicada apenas durante a geração da solução inicial. O tempo máximo de execução de quatro horas é aplicado à primeira etapa diretamente, uma vez que a busca local, mesmo sem critério de parada, é finalizada em poucos minutos.

Ainda visando reduzir o tempo de execução da abordagem exata, é definido que a etapa de geração da solução inicial deve primeiramente ser capaz de gerar uma rota por subtarefa em vez de diretamente gerar uma lista completa de movimentos para o plano de produção. Em acréscimo, para reduzir o tamanho do modelo ainda mais, o modelo de programação linear também deve receber uma lista de submovimentos por subtarefa como parâmetro de entrada, estabelecendo um limite superior ao modelo matemático. Duas versões são propostas: uma lista de submovimentos vazia e o resultado fornecido pelo operador compartilhado *BuildMoveListCombined*.

Na primeira versão o modelo matemático é responsável inicialmente por gerar completamente a rota ótima por subtarefa. Já na segunda versão ele deve aperfeiçoar até o ótimo partindo do resultado de *BuildMoveListCombined*. É válido ressaltar que, mesmo sendo importante otimizar o problema como um todo para garantir a eficiência dos movimentos durante todo o plano de produção, o ótimo almejado nesse momento é no contexto da subtarefa e não do plano de produção completo, para que seja possível encontrar uma solução exata no tempo limitado especificado.

A fim de experimentar diferentes abordagens, são ainda propostas duas outras versões relativas ao modelo matemático: modelo temporal e modelo em estágios. O modelo temporal aplica um único modelo exato para gerar o caminho mínimo de cada subtarefa e em seguida apenas concatena os resultados para gerar uma solução completa para o plano de produção. O modelo em estágios divide as responsabilidades de encontrar os submovimentos de cada subtarefa e a melhor organização temporal dos submovimentos do plano de produção completo em dois submodelos sequenciais.

As versões referentes à lista inicial de submovimentos por subtarefa são aplicadas aos dois modelos, convergindo para duas heurísticas com duas versões internas: ILP–Heur Temporal e ILP–Heur em Estágios. A primeira é apresentada na subseção 5.2.2.1, enquanto a segunda é descrita na subseção 5.2.2.2. Em ambas heurísticas a construção da solução inicial com uso do modelo exato é seguida da mesma etapa de busca local para otimização da solução.

5.2.2.1 ILP–Heur Temporal

ILP–Heur Temporal é composta por duas etapas principais: geração da solução inicial e busca local. A primeira etapa é responsável por gerar o caminho mínimo para cada subtarefa com uso do modelo temporal e ela pode ser executada em duas versões, conforme a lista de submovimentos recebida como parâmetro de entrada. São realizadas múltiplas execuções do modelo matemático em sequência, uma para cada subtarefa. A segunda etapa otimiza a solução incumbente de forma global utilizando vizinhanças de busca local já apresentadas nos componentes compartilhados.

5.2.2.1.1 Geração da Solução Inicial

A fim de gerar uma solução inicial para o plano de produção, o modelo temporal utiliza uma modelagem linear inteira mista para construir o caminho mínimo de cada subtarefa. Esse caminho mínimo inclui todos os submovimentos da rota, assim como a ordenação e a atribuição temporal dos submovimentos. Essa assertiva é relevante, e define o nome do modelo, uma vez que a relação entre os submovimentos e a atribuição temporal é o que define a quantidade de movimentos e a duração do caminho mínimo (dois submovimentos com a mesma bobina executados separadamente exigem que a ponte rolante execute a ação de içar duas vezes, gerando dois movimentos).

A execução da primeira subtarefa considera a configuração real de posicionamento das bobinas na fábrica e as subtarefas em sequência consideram o posicionamento e o tempo obtido ao fim da otimização anterior. A lista de movimentos completa, correspondente à solução inicial, é gerada a partir da concatenação do caminho mínimo de cada subtarefa.

Vale ressaltar um novo conceito introduzido neste modelo: o instante de movimentação. O instante de movimentação é definido como a ação de uma ou mais U.M.s executarem um submovimento em um mesmo instante. Como as quatro U.M.s (duas pontes rolantes e dois carros) podem ser operadas em paralelo, um instante de movimentação pode abranger até quatro submovimentos, desde que sejam em U.M.s distintas. O instante de movimentação é uma alternativa para não indexar o posicionamento das bobinas pelo tempo, que é dependente do horizonte de planejamento do plano de produção.

Na versão com lista inicial de submovimentos vazia, para reduzir a dimensão do modelo, é definida uma quantidade máxima de 15 instantes de movimentação para cada subtarefa. Caso não seja possível alcançar a posição de destino da subtarefa com essa quantidade, o modelo é executado novamente para a subtarefa em uma segunda tentativa com 26 instantes de movimentação. Essas quantidades foram definidas experimentalmente. Com 15 instantes de movimentação é possível otimizar várias subtarefas com baixo tempo de execução, mas para alguns casos essa quantidade não é suficiente. Com 26 instantes são incluídos mais alguns casos, e é factível encontrar uma solução dentro de um tempo restrito. Acima de 26 instantes de

movimentação, o tamanho do modelo gera um aumento do tempo de execução não condizente com os limites estabelecidos.

Na versão que utiliza o resultado do operador *BuildMoveListCombined* inicialmente é realizado um processamento para identificar a quantidade de instantes de movimentação e, em seguida, o modelo é executado com essa quantidade mais uma unidade. Caso a quantidade seja maior que 70, assume-se que a heurística falhou, pois a dimensão do modelo é muito grande e normalmente existe falha por falta de recursos computacionais para sua execução. Essa versão tem grande potencial para ser melhor do que a com lista de submovimentos vazia. Além de fornecer uma solução inicial melhor, a quantidade de instantes de movimentação é personalizada para cada execução. Dessa forma, é possível executar o modelo sempre uma única vez com poucos instantes quando possível e com uma quantidade maior que 26 quando necessário.

Os conjuntos de variáveis e parâmetros utilizados pelo modelo temporal para geração do caminho mínimo de cada subtarefa são descritos abaixo.

- $pos_{i,b,m}$: variável binária que indica se a bobina b está na posição i durante um instante de movimentação m.
- $mov_{b,i,j,q,m}$: variável binária que indica se a bobina b é deslocada da posição i para a posição j no instante de movimentação m, usando a U.M. q. Essa variável é correspondente ao submovimento, uma vez que o deslocamento em cada instante de movimentação é restrito a posições vizinhas.
- t_m variável inteira que representa a quantidade de u.t. necessária até um determinado instante de movimentação m.
- $\mathcal{P}$: conjunto de posições.
- $\mathcal{Q}$: conjunto de U.M.s; a Ponte Rolante A é representada pelo número 1, a Ponte Rolante B por 2, o CAR1 por 3 e CAR2 por 4.
- $\mathcal{B}$: conjunto de bobinas.
- $\mathcal{M}$: conjunto de instantes de movimentação permitidos (a quantidade de instantes de movimentação é limitada; o modelo pode usar a quantidade necessária de instantes de movimentação desde que esta não ultrapasse o limite; o último instante de movimentação é dito m_f).
- t_i: horário mínimo para iniciar o plano de movimentação; esse valor considera quando a bobina e a posição da subtarefa estarão disponíveis e quando o plano de movimentação da subtarefa anterior foi encerrado.
- t_s: horário inicial da subtarefa s no plano de produção.

- t_p: quantidade de u.t. definida para movimentação das pontes rolantes.
- t_c: quantidade de u.t. definida para movimentação dos carros.
- b_s: bobina da subtarefa s.
- p_s: posição destino da subtarefa s.
- $c_{i,j,q}$: parâmetro binário que estabelece se a U.M. q conecta diretamente as posições i e j; i.e., se a U.M. q está na mesma região de i e j, se existe um caminho válido entre essas posições e se elas são vizinhas.
- $p_{i,b}$: parâmetro binário que indica se, inicialmente, a bobina b está alocada na posição i.

Em teoria o modelo temporal deveria avaliar exatamente os critérios apresentados no Capítulo 3 e as respectivas funções objetivo. Na prática, as cinco funções objetivo são aplicadas na etapa de busca local, mas para a geração da solução inicial é necessário realizar alguns ajustes:

- Como a otimização é realizada por subtarefa, a função objetivo F_1 (minimizar o número de subtarefas incompletas) não é mais aplicável.
- O conceito de operação não existe no modelo temporal. Por essa razão o objetivo F_2 (minimizar o número de operações cujo destino final é um carro) é transformado em uma restrição onde o carro não pode estar ocupado ao fim da subtarefa.
- Ainda pela falta do conceito de operação, o objetivo F_5 é alterado de "minimizar o número de operações executadas" para "minimizar a quantidade de submovimentos executados". O conceito de movimento não é utilizado nesse objetivo porque não existe uma variável para movimento no modelo, ele é apenas utilizado implicitamente durante o cálculo da variável que representa o tempo.
- Como o modelo é aplicado por subtarefa não é possível calcular nem o atraso total nem o por tarefa, como definido em F_3.
- Como o modelo é aplicado por subtarefa a maximização do adiantamento total especificada em F_4 é substituída apenas pelo adiantamento da subtarefa.
- Na modelagem do adiantamento, não existe diferença se uma subtarefa não é completada ou se é completada com atraso. Por essa razão, apesar de não mensurar o atraso, a função objetivo do modelo temporal penaliza, a cada instante de movimentação, se a bobina não está na posição de destino.

Em resumo, a função objetivo do modelo temporal contempla dois critérios e uma penalidade: a maximização do adiantamento da subtarefa, a penalidade relativa ao atraso e a

minimização da quantidade de submovimentos, como apresentado na função (5.1). Assim como definido na composição lexicográfica, existe uma definição de prioridade entre os termos da função objetivo representada pelos fatores n_1, n_2 e n_3.

$$\max[(n_1 \cdot \sum_{m \in \mathcal{M}} \max(0, (t_s \cdot pos_{p_s,b_s,m}) - t_m)) - (n_2 \cdot \sum_{m \in \mathcal{M}} \max(0, (t_s - t_s \cdot pos_{p_s,b_s,m}))) - (n_3 \cdot \sum_{m \in \mathcal{M}} \sum_{q \in \mathcal{Q}} \sum_{b \in \mathcal{B}} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{P}} mov_{b,i,j,q,m})] \tag{5.1}$$

Para satisfazer as características do problema e construir uma lista de submovimentos factível para a subtarefa são definidas diversas restrições para o modelo temporal, representadas pelas equações (5.2) a (5.18). A equação (5.2) define a restrição de fluxo, indicando que se uma posição recebeu ou forneceu uma bobina, também deve haver um submovimento relacionado. Em sequência, (5.3) restringe que uma bobina sempre deve estar em uma única posição e (5.4) estabelece que cada posição pode conter no máximo uma bobina em cada instante de movimentação.

$$pos_{i,b,m} - pos_{i,b,m+1} = \sum_{q \in \mathcal{Q}} \sum_{j \in \mathcal{P}} mov_{b,i,j,q,m+1} - \sum_{q \in \mathcal{Q}} \sum_{j \in \mathcal{P}} mov_{b,j,i,q,m+1} \quad \forall\ m \in \mathcal{M}, b \in \mathcal{B}, i \in \mathcal{P} \tag{5.2}$$

$$\sum_{i \in \mathcal{P}} pos_{i,b,m} = 1 \quad \forall\ m \in \mathcal{M}, b \in \mathcal{B} \tag{5.3}$$

$$\sum_{b \in \mathcal{B}} pos_{i,b,m} \leq 1 \quad \forall\ m \in \mathcal{M}, i \in \mathcal{P} \tag{5.4}$$

As restrições em (5.5) definem que um submovimento só pode ser realizado com uma determinada U.M. se existir um caminho válido entre as respectivas posições com essa U.M.. Em acréscimo, um único submovimento simultâneo pode ser executado por U.M., conforme indicado na equação (5.6). Essa regra permite que as pontes rolantes e carros operem em paralelo, mas não define condições específicas na interseção entre as pontes rolantes e carros. Essas regras específicas, apresentadas em (5.7) e (5.8), indicam que em um mesmo submovimento apenas uma de três situações deve ocorrer: $i)$ uma bobina é adicionada no carro, ou; $ii)$ o carro se desloca, ou; $iii)$ a bobina é retirada do carro. Elas são necessárias uma vez que os carros não

conseguem estar fisicamente em duas posições ao mesmo tempo. A numeração das posições nas fórmulas segue a configuração da Figura 3.

$$mov_{b,i,j,q,m} \leq c_{i,j,q} \quad \forall\ m \in \mathcal{M}, b \in \mathcal{B}, q \in \mathcal{Q}, i \in \mathcal{P}, j \in \mathcal{P} \tag{5.5}$$

$$\sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,m} \leq 1 \quad \forall\ m \in \mathcal{M}, q \in \mathcal{Q} \tag{5.6}$$

$$\begin{aligned}\sum_{b\in\mathcal{B}} mov_{b,12,24a,1,m} + \sum_{b\in\mathcal{B}} mov_{b,24a,12,1,m} + \sum_{b\in\mathcal{B}} mov_{b,24a,24b,3,m} \\ + \sum_{b\in\mathcal{B}} mov_{b,24b,24a,3,m} + \sum_{b\in\mathcal{B}} mov_{b,24b,38,2,m} + \sum_{b\in\mathcal{B}} mov_{b,38,24b,2,m} \quad \leq 1 \quad \forall\ m \in \mathcal{M}\end{aligned} \tag{5.7}$$

$$\begin{aligned}\sum_{b\in\mathcal{B}} mov_{b,59,25,4,m} + \sum_{b\in\mathcal{B}} mov_{b,25,59,4,m} \\ + \sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{P}} mov_{b,i,25,2,m} + \sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{P}} mov_{b,25,i,2,m} \quad \leq 1 \quad \forall\ m \in \mathcal{M}\end{aligned} \tag{5.8}$$

Para garantir que não existam simultaneamente duas bobinas nas duas posições de CAR1 ($24a$ e $24b$), é estipulada a equação (5.9). Ainda sobre os carros, como não existe o conceito de operação no modelo temporal, o objetivo F_2 apresentado no Capítulo 3 é transformado nas restrições (5.10) e (5.11), indicando que ao final do plano de movimentação da subtarefa não pode existir bobina nos carros.

$$\sum_{b\in\mathcal{B}} pos_{24a,b,m} + \sum_{b\in\mathcal{B}} pos_{24b,b,m} \leq 1 \quad \forall\ m \in \mathcal{M} \tag{5.9}$$

$$pos_{24a,b,m_f} + pos_{24b,b,m_f} = 0 \quad \forall\ b \in \mathcal{B} \tag{5.10}$$

$$pos_{59,b,m_f} = 0 \quad \forall\ b \in \mathcal{B} \tag{5.11}$$

A equação (5.12) define a distribuição inicial das bobinas entre as posições da fábrica. Sobre a organização temporal do modelo, (5.13) representa o tempo após o primeiro instante de movimentação, considerando o horário inicial t_i fornecido como parâmetro e se existe algum submovimento naquele instante (caso exista algum submovimento são contabilizadas t_p ou t_c unidades de tempo, dependendo da U.M. utilizada). Movimentos com U.M.s diferentes podem ser executados em paralelo, o que é garantido por (5.14), que calcula o tempo nos demais

instantes de movimentação, sempre somando a u.t. anterior com a atual. Essa equação difere da anterior porque submovimentos em sequência com a mesma bobina e U.M. são contabilizados em uma única u.t.. Logo, é verificado se existe algum submovimento começando naquele instante para somar as unidades de tempo. Caso o deslocamento de uma mesma bobina esteja sendo finalizado ou ainda esteja em execução, não existe acréscimo de tempo.

$$pos_{i,b,1} = p_{i,b} \quad \forall\ b \in \mathcal{B}, i \in \mathcal{P} \tag{5.12}$$

$$t_1 = t_i + \max[\min(t_p, \sum_{q\in\mathcal{Q}}\sum_{b\in\mathcal{B}} t_p \cdot \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,1}), \min(t_c, \sum_{q\in\mathcal{Q}}\sum_{b\in\mathcal{B}} t_c \cdot \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,1})] \tag{5.13}$$

$$\begin{aligned} t_q =& t_{q-1} + \max[\min(t_p, \sum_{q\in\mathcal{Q}}\sum_{b\in\mathcal{B}} t_p \cdot \max(0, \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,m} - \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,m-1}))), \\ & \min(t_c, \sum_{q\in\mathcal{Q}}\sum_{b\in\mathcal{B}} t_c \cdot \max(0, \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,m} - \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} mov_{b,i,j,q,m-1})))] \quad \forall\ m \in \mathcal{M} \end{aligned} \tag{5.14}$$

Os últimos conjuntos de restrições são representados pelas equações (5.15) a (5.18) e dizem respeito às posições $13|A(1,1)$, $14|A(1,2)$, $16|A(1,4)$, $17|A(1,5)$, $22|M(0,4)$ e $23|M(0,2)$, que podem ser bloqueadas por movimentos específicos como explicado no Capítulo 3 (regras para as máquinas que constroem a primeira camada do tubo flexível, em que as posições de máquina se deslocam para disponibilizar as bobinas para a ponte rolante).

$$\begin{aligned} \sum_{m\in\mathcal{M}}\sum_{b\in\mathcal{B}} pos_{16,b,m} + mov_{b,16,22,1,m+1} + mov_{b,22,16,1,m+1} \\ + mov_{b,17,22,1,m+1} + mov_{b,22,17,1,m+1} \quad \leq 1 \end{aligned} \tag{5.15}$$

$$\begin{aligned} \sum_{m\in\mathcal{M}}\sum_{b\in\mathcal{B}} pos_{17,b,m} + mov_{b,16,22,1,m+1} + mov_{b,22,16,1,m+1} \\ + mov_{b,17,22,1,m+1} + mov_{b,22,17,1,m+1} \quad \leq 1 \end{aligned} \tag{5.16}$$

$$\begin{aligned} \sum_{m\in\mathcal{M}}\sum_{b\in\mathcal{B}} pos_{13,b,m} + mov_{b,13,23,1,m+1} + mov_{b,23,13,1,m+1} \\ + mov_{b,14,23,1,m+1} + mov_{b,23,14,1,m+1} \quad \leq 1 \end{aligned} \tag{5.17}$$

$$\begin{aligned} \sum_{m\in\mathcal{M}}\sum_{b\in\mathcal{B}} pos_{14,b,m} + mov_{b,13,23,1,m+1} + mov_{b,23,13,1,m+1} \\ + mov_{b,14,23,1,m+1} + mov_{b,23,14,1,m+1} \quad \leq 1 \end{aligned} \tag{5.18}$$

Para agilizar o modelo temporal deve ser realizado um filtro prévio nos parâmetros de entrada para diminuir a quantidade de posições e bobinas avaliadas. Devem ser removidas as posições de máquina em operação no horário inicial de cada subtarefa e as respectivas bobinas em uso. Também devem ser removidas outras bobinas e posições que potencialmente não serão utilizadas na criação do caminho mínimo, conforme as regras abaixo (para referência de coordenadas e regiões, vide Figura 3):

- Se a bobina e a posição de destino estão na mesma região e a coordenada horizontal de ambas é inferior ao número 5, são desconsideradas as posições e as respectivas bobinas da outra região.
- Se a bobina e a posição de destino estão na mesma região e a coordenada horizontal de alguma é superior ao número 5, são desconsideradas as posições e as respectivas bobinas cuja coordenada horizontal é inferior ao número 10 da outra região. Neste caso, é necessário considerar uma parte da outra região porque é provável que seja necessário transportar alguma bobina de uma região para a outra a fim de gerar caminho livre.
- Se a bobina e a posição de destino estão em regiões opostas da fábrica, são desconsideradas as posições e as respectivas bobinas cuja coordenada horizontal é inferior à coordenada da posição que contém a bobina inicialmente menos 5, em uma região, e da mesma forma, na outra região, com a coordenada da posição de destino menos 5.
- As posições com coordenada horizontal igual a zero raramente são usadas pelo plano de movimentação. Por essa razão, elas só são incluídas caso a bobina da subtarefa esteja em uma dessas posições.

Para tentar respeitar o critério de parada de quatro horas, o tempo de execução do modelo temporal deve ser limitado à quarenta minutos para cada subtarefa. Caso o ótimo não tenha sido encontrado em quarenta minutos, deve ser retornada a melhor solução candidata obtida até esse instante. Quando o modelo recebe como parâmetro de entrada uma lista de submovimentos vazia, caso uma outra solução candidata não vazia não seja encontrada, considera-se que o algoritmo não foi capaz de solucionar o problema e ele é finalizado. Na segunda versão, mesmo que ao fim dos quarenta minutos o resultado ótimo não tenha sido encontrado, nem nenhuma outra solução candidata, a solução proveniente do operador compartilhado *BuildMoveListCombined* é retornada. Caso todas as subtarefas não sejam finalizadas em quatro horas, também deve ser considerado que o algoritmo não conseguiu solucionar o caso.

Após concatenar as soluções de cada subtarefa, a heurística aplica a vizinhança *SubtaskSequence* com os três métodos na mesma configuração de VNS–Heur (primeiramente *SubtaskSequenceA*, seguido de *SubtaskSequenceB* e por fim *SubtaskSequenceC*). O fluxo para geração da solução inicial em ILP–Heur Temporal da versão com lista inicial de submovimentos

vazia pode ser visualizado na Figura 9. O fluxo da versão que utiliza o operador compartilhado *BuildMoveListCombined* é apresentado na Figura 10.

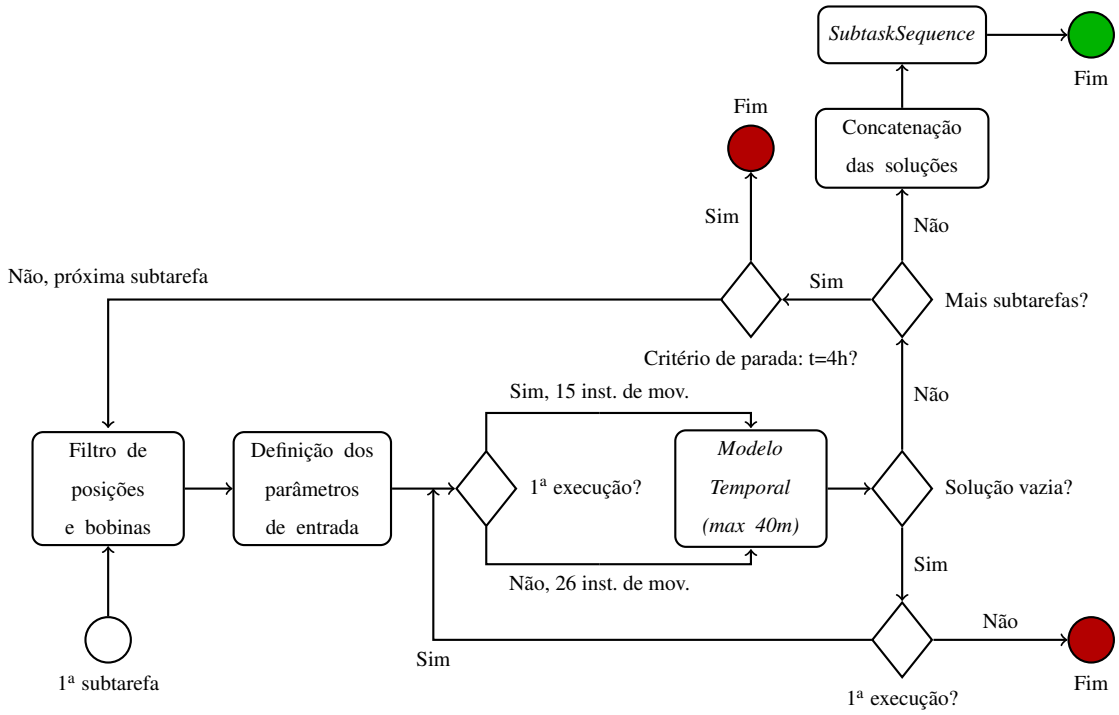


Figura 9 – Fluxograma Geração da Solução Inicial em ILP–Heur Temporal - Lista vazia

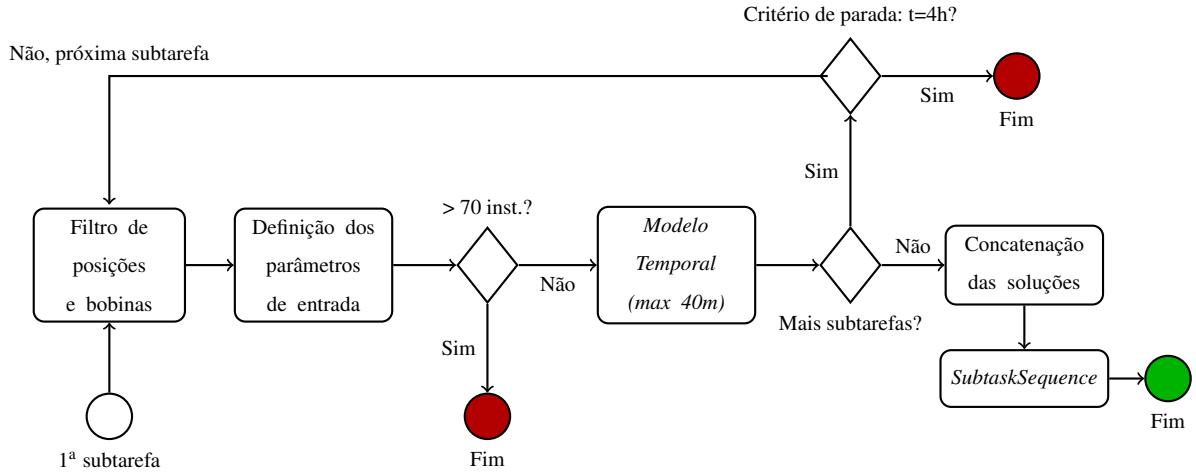


Figura 10 – Fluxograma Geração da Solução Inicial em ILP–Heur Temporal - *BuildMoveListCombined*

5.2.2.1.2 Busca Local

Para aperfeiçoar a solução inicial são aplicadas em sequência cinco vizinhanças de busca local apresentadas na subseção 5.1.4: *OperationSplit*, *PreMoveOperation*, *PreMoveTask*, *AnticipateSubtask* e *OperationSwap*. Como o modelo matemático não considera exatamente as mesmas funções objetivo da busca local, a solução incumbente da busca local pode ter um valor com grande potencial de aperfeiçoamento. Por essa razão, *OperationSplit* é utilizada na versão A, onde todas as operações são divididas e a nova solução é sempre mantida. Em acréscimo, as demais vizinhanças são repetidas enquanto a nova solução for melhor do que a solução inicial e não existe um critério de parada específico, pois o tempo de execução é normalmente de poucos minutos.

O fluxograma que representa a etapa de busca local é apresentado na Figura 11.

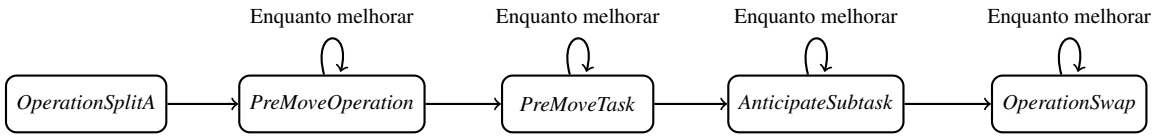


Figura 11 – Fluxograma da etapa de busca local em ILP–Heur Temporal

5.2.2.2 ILP–Heur em Estágios

Dado que submovimentos sequenciais com a mesma bobina e U.M. são consolidados em um único movimento e que independentemente da quantidade de submovimentos o tempo atribuído ao movimento é sempre o mesmo, o Modelo Temporal pode ter que comparar diversas ordenações de submovimentos ao construir o plano de movimentação para garantir o adiantamento máximo da subtarefa.

Por essa razão, apesar de ILP–Heur em Estágios também ser composta pelas etapas de geração da solução inicial e busca local como ILP–Heur Temporal, ILP–Heur em Estágios altera a etapa de geração da solução inicial para não incluir o conceito de movimento em um primeiro momento. Os objetivos de encontrar rotas por subtarefa com a menor quantidade de submovimentos e identificar a melhor ordenação de todos os submovimentos do plano de produção são desacoplados em dois submodelos. Com essa divisão, ILP–Heur em Estágios ganha uma habilidade extra, em comparação a ILP–Heur temporal, de otimizar a atribuição temporal no contexto de todo o plano de produção e não apenas de cada subtarefa.

Essa proposta é baseada na hipótese de que, ao dividir a abordagem em dois estágios, os submodelos são mais simples e possivelmente tem um melhor desempenho levando em consideração o tempo de execução e a qualidade da solução (já que o segundo submodelo otimiza o plano de produção completo). É importante ressaltar que, apesar do segundo submodelo

otimizar o plano de produção como um todo, o resultado não é um ótimo global para o problema já que a lista de submovimentos é fixa e gerada no primeiro submodelo para o contexto de cada subtarefa.

5.2.2.2.1 Geração da Solução Inicial

A fim de gerar uma solução inicial para o plano de produção, o modelo em estágios utiliza dois submodelos sequenciais: atemporal e encaixe temporal. O submodelo atemporal é responsável por gerar a menor quantidade de submovimentos necessária para cada subtarefa, enquanto o encaixe temporal deve ordenar e atribuir tempo aos submovimentos de todo o plano de produção.

Assim como no modelo temporal, a execução da primeira subtarefa no submodelo atemporal considera a configuração real de posicionamento das bobinas na fábrica e as subtarefas em sequência consideram o posicionamento obtido ao fim da otimização anterior. A lista de submovimentos completa, fornecida para o encaixe temporal, é gerada a partir da concatenação do resultado de cada subtarefa.

Como o submodelo atemporal não é capaz de gerar os movimentos e os seus respectivos tempos, a avaliação das funções objetivo é ainda mais restrita do que no modelo temporal. Não é possível penalizar o atraso e o adiantamento é calculado pela quantidade de instantes de movimentação em que a bobina da subtarefa já está na posição de destino, sem relação direta com tempo.

Em resumo, a função objetivo do submodelo atemporal contempla dois critérios: a maximização do adiantamento da subtarefa e a minimização da quantidade de submovimentos, como apresentado na função (5.19). Assim como definido na composição lexicográfica, existe uma definição de prioridade entre os termos da função objetivo representada pelos fatores n_1 e n_2. Nessa função o primeiro termo é relativo ao adiantamento, seguido pela quantidade de submovimentos.

$$\max[(n_1 \cdot \sum_{m \in \mathcal{M}} pos_{p_s,b_s,m}) - (n_2 \cdot \sum_{m \in \mathcal{M}} \sum_{q \in \mathcal{Q}} \sum_{b \in \mathcal{B}} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{P}} mov_{b,i,j,q,m})] \tag{5.19}$$

Para satisfazer as características do problema e construir um conjunto de submovimentos factível para a subtarefa, o submodelo atemporal utiliza as mesmas restrições do modelo temporal, com exceção das equações 5.13 e 5.14. Essas equações definem o tempo de planejamento a partir da ordenação das bobinas movimentadas e, portanto, deixam de existir.

Apesar de ILP–Heur em Estágios conter dois submodelos menores que ILP–Heur Temporal, ainda é importante reduzir o tamanho do modelo. Por essa razão, assim como o modelo temporal, o submodelo atemporal também tem duas versões conforme a lista de submovimentos

por subtarefa definida como parâmetro de entrada (lista de submovimentos vazia ou a partir do operador *BuildMoveListCombined*). As regras aplicadas para a quantidade de instantes de movimentação em ambas versões são as mesmas do modelo temporal. Da mesma forma, o filtro prévio nos parâmetros de entrada para diminuir a quantidade de posições e bobinas avaliadas também é aplicado ao submodelo atemporal e cada subtarefa pode ser executada por no máximo quarenta minutos.

O submodelo de encaixe temporal é responsável pela ordenação temporal de todos os movimentos obtidos no estágio anterior. Para reduzir o tamanho do modelo o encaixe temporal lida com movimentos em vez de submovimentos. Submovimentos em sequência com mesma bobina e U.M. gerados pelo submodelo atemporal devem ser agrupados em movimentos antes do início do encaixe temporal.

É importante destacar que o encaixe temporal não modifica os movimentos definidos anteriormente, apenas organiza e atribui os instantes de tempo em que cada movimento deve ser executado. Dessa forma, é possível adiantar e/ou agrupar alguns movimentos, desde que as regras de precedência não sejam quebradas (i.e., não é possível executar o movimento de retirada de uma bobina de uma determinada posição antes que o movimento de colocar a bobina nessa posição seja realizado).

Os conjuntos de variáveis e parâmetros do submodelo de encaixe temporal são descritos abaixo.

- $x_{j,t}$: variável binária que indica se o movimento j é executado no instante de tempo t.
- $y_{b,q,t}$: variável binária que indica se a bobina b é movimentada pela U.M. q no instante de tempo t.
- t_{max}: tempo máximo do plano de produção, que é igual ao horário de término da última tarefa.
- $\mathcal{T} = \{1, 2, \ldots, t_{max}\}$: conjunto de unidades de tempo.
- $\mathcal{B}$: conjunto de bobinas.
- $\mathcal{S}$: conjunto de subtarefas.
- $\mathcal{K}$: conjunto de tarefas.
- $\mathcal{Q}$: conjunto de U.M.s.
- j_{max}: quantidade total de movimentos.
- $\mathcal{J} = \{1, 2, \ldots, j_{max}\}$: conjunto de movimentos.
- $\mathcal{J}_q$: conjunto de movimentos executado pela U.M. q.

- j_{k1}: último movimento da subtarefa 1 da tarefa k.
- j_{k2}: último movimento da subtarefa 2 da tarefa k.
- $\mathcal{P}_j$: conjunto de posições usadas no movimento j.
- bob_j: bobina deslocada no movimento j.
- $bob_{b,j}$: parâmetro binário que indica se a bobina b é deslocada no movimento j.
- T_u: início da tarefa u conforme o plano de produção.
- C_{Kq}: quantidade de u.t. definida para movimentação de cada U.M..
- t_{i_j}: horário inicial mínimo para o movimento j (é diferente de zero caso alguma bobina ou posição envolvida no movimento já tenha sido utilizada em uma tarefa anterior).

O encaixe temporal também não consegue avaliar exatamente os critérios apresentados no Capítulo 3 e as respectivas funções objetivo. A função objetivo F_1 (minimizar o número de subtarefas incompletas) não é aplicável, uma vez que as subtarefas já foram processadas individualmente no submodelo atemporal. Como não existe o conceito de operação, o objetivo F_2 (minimizar o número de operações cujo destino final é um carro) também não é avaliado. Em compensação, como o submodelo atemporal planeja a retirada de bobina no carro ao final da subtarefa, esse movimento também existirá no encaixe temporal em algum momento.

O objetivo F_3 (minimizar o atraso total) é aplicado sem alterações, mas F_4 é alterado de "maximizar o adiantamento total" para "minimizar a antecipação total". A antecipação é definida apenas a partir do horário do último movimento de cada subtarefa, sem referência ao horário de início da tarefa conforme o plano de produção. Por fim, o submodelo atemporal fornece um conjunto de movimentos para o encaixe temporal, mas é desejado agrupar e reduzir essa quantidade de movimentos. Por essa razão, não é possível gerar um objetivo a partir da variável de movimentos e o objetivo F_5 é alterado de "minimizar o número de operações executadas" para "minimizar a quantidade de bobinas se deslocando".

Em resumo, a função objetivo do encaixe temporal contempla três critérios: a minimização do atraso total, a minimização da antecipação total e a minimização da quantidade de bobinas se deslocando, como apresentado na função (5.20). Assim como definido na composição lexicográfica, existe uma definição de prioridade entre os termos da função objetivo, representado pelos fatores n_1, n_2 e n_3. Nessa função, o primeiro termo é relativo ao atraso, seguido pelo adiantamento e pela quantidade de bobinas se deslocando.

$$\begin{aligned}\min f(x) =&(n_1 \cdot \sum_{k\in\mathcal{K}} \max(0, \sum_{t\in\mathcal{T}} t \cdot x_{j_{k1},t} - T_{task_k}, \sum_{t\in\mathcal{T}} t \cdot x_{j_{k2},t} - T_{task_k})) \\ &+ (n_2 \cdot \sum_{k\in\mathcal{K}} \sum_{t\in\mathcal{T}} t \cdot x_{j_{k1},t} + t \cdot x_{j_{k2},t}) + (n_3 \cdot \sum_{b\in\mathcal{B}} \sum_{q\in\mathcal{Q}} \sum_{t\in\mathcal{T}} y_{b,q,t})\end{aligned} \tag{5.20}$$

Para satisfazer as características do problema e construir uma lista de movimentos factível para o plano de produção, são definidas diversas restrições para o submodelo de encaixe temporal representadas pelas equações (5.21) a (5.27).

A equação (5.21) define que cada movimento deve ser executado uma única vez. Por sua vez, (5.22) restringe que cada ponte rolante/carro só pode movimentar uma bobina por vez e que o somatório de tempo é igual ao tempo de movimentação da U.M.. O conjunto de restrições (5.23) estabelece a associação entre bobina e movimento. Se um determinado movimento é executado em um dado instante de tempo, então a variável que indica o movimento da bobina correspondente àquele instante também deve ter valor unitário. Essa associação permite que os movimentos gerados no submodelo anterior sejam agrupados para redução do tempo de deslocamento, uma vez que mais de um movimento com a mesma bobina pode ser executado em um mesmo instante de tempo/U.M..

$$\sum_{t \in \mathcal{T}} x_{j,t} = 1 \quad \forall\ j \in \mathcal{J} \tag{5.21}$$

$$\sum_{b \in \mathcal{B}} \sum_{s=t}^{t+C_{Kq}-1} y_{b,q,s} \leq 1 \quad \forall\ q \in \mathcal{Q}, t \in 1..t_{max} - C_{Kq} + 1 \tag{5.22}$$

$$\min(1, \sum_{j \in \mathcal{J}_{\text{II}}} bob_{b,j} \cdot x_{j,t}) = y_{b,q,t} \quad \forall\ q \in \mathcal{Q}, t \in \mathcal{T}, b \in B \tag{5.23}$$

A equação (5.24) estabelece um horário inicial mínimo para cada movimento, baseado no plano de produção. Caso a bobina e/ou a posição de destino do movimento sejam utilizadas em uma tarefa anterior, o movimento só pode iniciar após a finalização da respectiva tarefa.

$$\sum_{t \in \mathcal{T}} t \cdot x_{j,t} \geq t_{i_j} \quad \forall\ j \in \mathcal{J} \tag{5.24}$$

As restrições em (5.25) e (5.26) definem que, se uma bobina estava deslocando em uma U.M., só pode haver um próximo movimento para ela em outra U.M. após a finalização do anterior. (5.25) indica a restrição no sentido carro - ponte rolante, enquanto (5.26) indica a restrição no sentido ponte rolante - carro. Nesta última equação, apenas o tempo de uma das pontes rolantes é utilizado, uma vez que o tempo de movimentação é igual para ambas. Um ponto

importante é que (5.26) não é válida para o carro 3, uma vez que não são permitidos movimentos da parte interna da fábrica para a parte externa.

$$bob_i = bob_j \Rightarrow \sum_{t \in \mathcal{T}} t \cdot x_{i,t} \geq \sum_{t \in \mathcal{T}} t \cdot x_{j,t} + C_{K4} \quad \forall\ i \in \mathcal{J}_1 \cup \mathcal{J}_2, j \in \mathcal{J}_3 \cup \in \mathcal{J}_4, i > j \tag{5.25}$$

$$bob_i = bob_j \Rightarrow \sum_{t \in \mathcal{T}} t \cdot x_{j,t} \geq \sum_{t \in \mathcal{T}} t \cdot x_{i,t} + C_{K1} \quad \forall\ i \in \mathcal{J}_1 \cup \mathcal{J}_2, j \in \mathcal{J}_4, j > i \tag{5.26}$$

Para finalizar, as restrições em (5.27) definem, para cada U.M., que se um movimento utiliza alguma posição de um movimento anterior, ele deve acontecer depois.

$$\mathcal{P}_i \cap \mathcal{P}_j \geq 1 \Rightarrow \sum_{t \in \mathcal{T}} t \cdot x_{i,t} \geq \sum_{t \in \mathcal{T}} t \cdot x_{j,t} + C_{Kq} \quad \forall\ q \in \mathcal{Q}, i, j \in \mathcal{J}_q, i > j \tag{5.27}$$

No submodelo de encaixe temporal as restrições de precedência representadas pelas equações (5.25) a (5.27) são construídas dinamicamente a partir da lista de movimentos proveniente do submodelo atemporal. Portanto, antes de executar o encaixe temporal é necessário avaliar a lista de movimentos para construir a relação de movimentos de diferentes U.M.s que usam a mesma bobina e a relação de movimentos que usam a mesma posição.

O conjunto de unidades de tempo fornecido para o encaixe temporal abrange todo o horizonte de tempo do plano de produção, que pode durar até 10 dias. Por essa razão, a dimensão do modelo pode crescer muito e, assim como em ILP–Heur Temporal, é necessário encontrar alternativas que diminuam o tamanho do modelo e melhorem seu desempenho. Para agilizar o encaixe temporal deve ser realizada uma redução nas unidades de tempo conforme a seguinte regra: se o intervalo entre o início de duas subtarefas é maior do que 100 u.t. é possível supor que existe tempo ocioso e devem ser consideradas apenas 50 u.t. após o início da primeira subtarefa e 50 u.t. antes do início da segunda. Por exemplo, se o horário inicial de uma subtarefa é 60 u.t. e o horário inicial da subtarefa seguinte é 200 u.t., o conjunto de unidades de tempo considerado entre essas duas subtarefas é igual a $]60, 110[\ \cup\ ]150, 200[$.

Como o tempo de execução completo de ILP–Heur em Estágios deve ser no máximo quatro horas, o tempo limite de execução do submodelo de encaixe temporal é igual à diferença entre o critério de parada de quatro horas e o tempo utilizado pelo primeiro submodelo. É considerado que a heurística completa não foi capaz de solucionar o problema caso o encaixe temporal não encontre nenhum resultado no tempo disponível.

Por fim, caso o encaixe temporal tenha encontrado uma solução dentro do tempo permitido, a heurística aplica a vizinhança *SubtaskSequence* com os três métodos na mesma configuração de VNS–Heur (primeiramente *SubtaskSequenceA*, seguido de *SubtaskSequenceB* e por fim *SubtaskSequenceC*).

O fluxo genérico para geração da solução inicial em ILP–Heur em Estágios pode ser visualizado na Figura 12 e na Figura 13, considerando o círculo verde com a letra X como a conexão entre as figuras. A primeira figura tem foco no submodelo atemporal e a segunda descreve o uso do submodelo de encaixe temporal. A etapa *Submodelo Atemporal* na primeira figura não foi subdividida, mas ela considera os diferentes fluxos e validações para solução vazia e quantidade de instantes de movimentação apresentados na Figura 9 e na Figura 10 conforme a solução inicial definida como parâmetro de entrada.

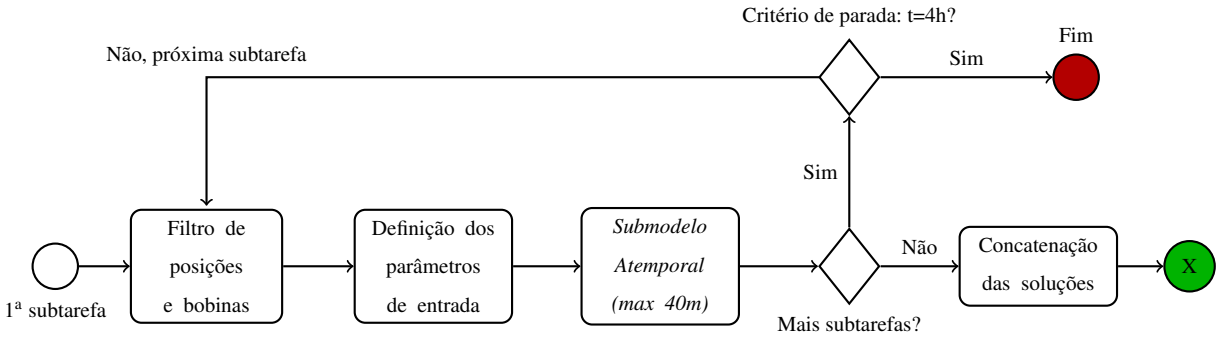


Figura 12 – Fluxograma do submodelo atemporal em ILP–Heur em Estágios

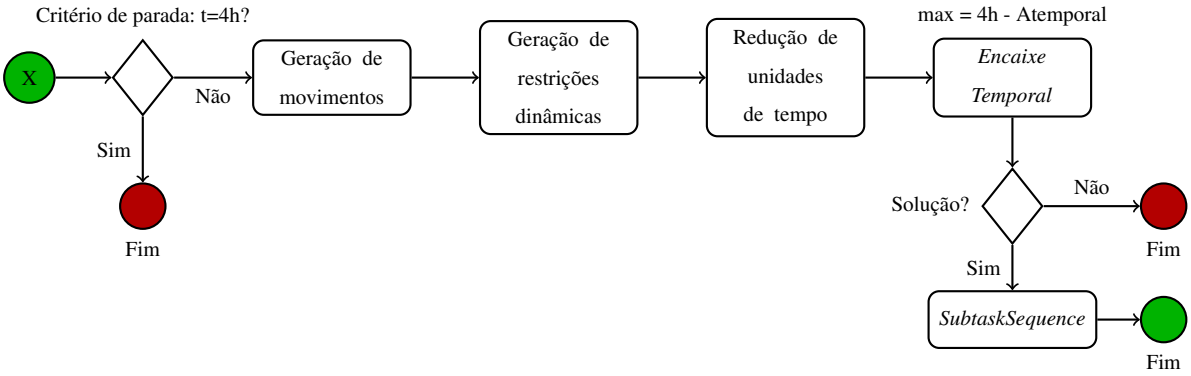


Figura 13 – Fluxograma do submodelo encaixe temporal em ILP–Heur em Estágios

5.2.2.2.2 Busca Local

Para aperfeiçoar a solução inicial ILP–Heur em Estágios utiliza exatamente a mesma configuração de busca local de ILP–Heur Temporal. São aplicadas em sequência cinco vizinhanças de busca local apresentadas na subseção 5.1.4: *OperationSplitA*, *PreMoveOperation*, *PreMoveTask*, *AnticipateSubtask* e *OperationSwap*. Neste momento não é considerado critério de parada, pois o tempo de execução é normalmente de poucos minutos.

5.3 Heurística para Replanejamento

O plano de produção contendo as tarefas a serem executadas é enviado para a operação diariamente. Esse plano contém as tarefas que estão em operação neste momento (tarefas cujo horário inicial de processamento da bobina é zero) e novas tarefas para um período entre cinco e dez dias, de forma a permitir o "planejamento a longo prazo" dos movimentos. Entretanto, modificações nesse plano podem acontecer ao longo do dia, devido a eventuais atrasos, adiantamentos ou manutenções corretivas a serem realizadas. Nesses casos é necessário otimizar o planejamento de novo, para o mesmo horizonte de tempo, a fim de gerar uma lista de movimentos atualizada. Durante a execução do algoritmo a movimentação das bobinas deve ser pausada, porque as posições correntes das bobinas na fábrica são entradas para o algoritmo.

Dada essa necessidade de replanejamento, a *Heurística Baseada em Busca Local para o Deslocamento de Bobinas* (LS–Heur) é proposta na subseção 5.3.1.1 para ajustes no plano de produção ao longo do dia, com tempo de execução máximo de cinco minutos (300 segundos).

5.3.1 LS–Heur

Dentro do conceito de replanejamento, a heurística de busca local LS–Heur é proposta para alcançar resultados de plano de movimentação de forma rápida utilizando os componentes compartilhados anteriormente apresentados. Conforme as vizinhanças selecionadas, a heurística pode ter um comportamento determinístico ou estocástico. Nesse sentido, LS–Heur é subdividida em duas versões conforme a abordagem utilizada: a primeira, nomeada *LS–Heur Determinística*, é apresentada na subseção 5.3.1.1 e a segunda, nomeada *LS–Heur Estocástica* é descrita na subseção 5.3.1.2.

5.3.1.1 LS–Heur Determinística

LS–Heur Determinística é composta por duas etapas principais que foram construídas a partir dos componentes compartilhados: geração da solução inicial e busca local. Ela é muito semelhante a VNS–Heur, mas como o próprio nome indica, essa versão utiliza uma abordagem determinística, em que para um mesmo conjunto de variáveis de entrada, o resultado é sempre igual. Dessa forma, a heurística abre mão da etapa de perturbação, que é estocástica e quando

aplicada em múltiplas iterações permite explorar de forma mais abrangente o espaço de variáveis de decisão, para garantir um resultado dentro do critério de parada de cinco minutos. O principal desafio nesse caso é garantir que, mesmo com um tempo reduzido e em apenas uma iteração, a solução seja eficiente e de acordo com as expectativas da operação da fábrica.

A etapa de geração da solução inicial de LS–Heur Determinística é semelhante à mesma etapa da heurística VNS–Heur, apresentada na subseção 5.2.1.1, com a diferença de utilizar apenas o método *SubtaskSequenceA* na vizinhança *SubtaskSequence* para reduzir o tempo de execução.

A etapa de busca local também contém apenas pequenas diferenças quando comparada a VNS–Heur. Como a busca local é sucessora da geração da solução inicial, que normalmente já tem uma solução eficiente, a vizinhança *OperationSplit* é utilizada com o método *OperationSplitB* em vez da versão *A*. Em acréscimo, *OperationSplitB* é repetida enquanto encontrar novas soluções melhores, assim como as demais vizinhanças subsequentes. Para respeitar o critério de parada, caso o limite de tempo seja alcançado a próxima vizinhança não é executada e a heurística é finalizada com a melhor solução encontrada até aquele momento. O fluxograma correspondente a LS–Heur Determinística, incluindo as etapas de geração da solução inicial e busca local, pode ser visualizado na Figura 14.

5.3.1.2 LS–Heur Estocástica

Para ampliar a capacidade de exploração do espaço de variáveis de decisão de LS–Heur Determinística mas, ao mesmo tempo, manter o critério de parada de cinco minutos, a versão LS–Heur Estocástica adiciona uma vizinhança de busca local estocástica e mantém a heurística restrita a uma iteração. A hipótese que originou a proposta desta heurística é que, mesmo adicionando uma componente aleatória, que fornece uma imprevisibilidade ao resultado, na maior parte das execuções a heurística deve gerar um resultado igual ou melhor que a versão determinística.

A etapa de geração da solução inicial de LS–Heur Estocástica é exatamente igual à mesma etapa de LS–Heur Determinística. A etapa de busca local é diferente apenas pela adição da vizinhança *RandomSearch*. Essa vizinhança é executada durante quarenta segundos, utilizando a variante *RandomSearchA*, em que uma nova solução é fornecida apenas se a solução incumbente for aprimorada. A vizinhança continua sendo executada enquanto houver aperfeiçoamento da solução e, por isso um tempo de quarenta segundos foi estimado na tentativa de conseguir encontrar uma solução dentro do critério de parada de cinco minutos.

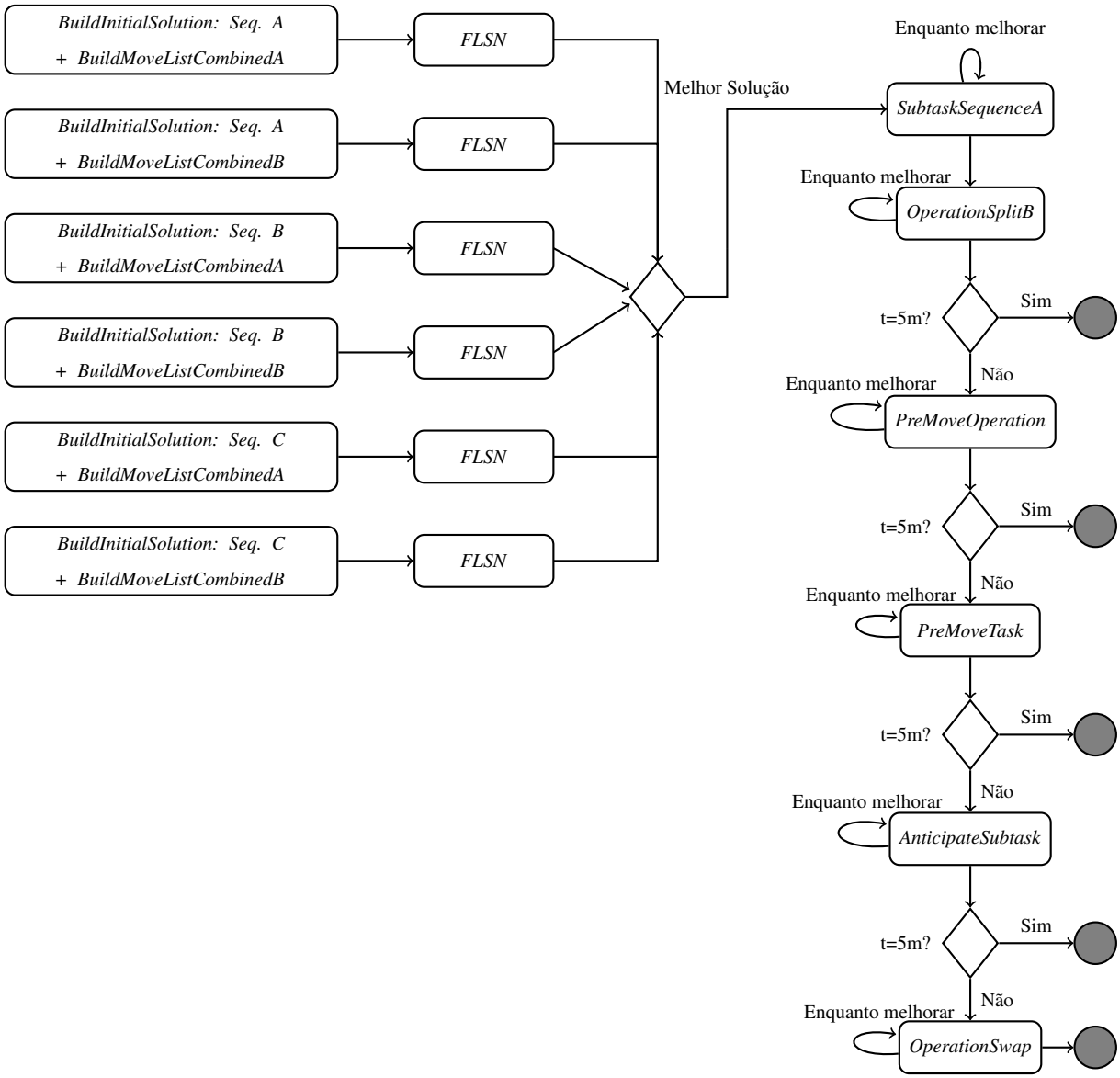


Figura 14 – Fluxograma completo de LS–Heur Determinística

5.4 Síntese das Heurísticas Propostas

As referências bibliográficas apresentadas no Capítulo 2 destacam diversos trabalhos com similaridades ao deslocamento de bobinas em uma fábrica de tubos flexíveis. São abordados casos como o agendamento de ponte rolante em um almoxarifado de bobinas, planejamento do armazenamento de placas de aço e movimentação de *containers* em terminais portuários. Para instâncias pequenas foram empregadas estratégias como redução de variáveis, programação dinâmica e *branch-and-bound*, mas ao tratar instâncias grandes, é notável o uso de métodos heurísticos como algoritmos genéticos e heurísticas de busca local.

Como não é possível aplicar uma abordagem exata para otimizar o deslocamento de bobinas completamente, assim como na grande maioria dos estudos citados na revisão bibliográ-

fica, foram propostas e apresentadas neste capítulo algumas heurísticas. Essas heurísticas estão divididas em três grupos (VNS–Heur, ILP–Heur e LS–Heur) e são aplicadas em instâncias de qualquer tamanho. VNS–Heur e ILP–Heur foram concebidas para o problema de planejamento e LS–Heur para o replanejamento, utilizando diferentes estratégias de otimização. VNS–Heur é baseada na meta-heurística VNS enquanto ILP–Heur aplica métodos exatos a pequenas etapas da resolução do problema acompanhados de vizinhanças de busca local e LS–Heur é completamente definida como uma heurística de busca local. Mesmo ILP–Heur aplicando métodos exatos apenas a pequenas etapas da resolução do problema, ainda são estabelecidas diversas estratégias para reduzir o tamanho do modelo como definição de uma solução inicial, limite da quantidade de movimentos e retirada de bobinas e posições que potencialmente não serão utilizadas na respectiva execução.

É importante citar que as estratégias utilizadas para o planejamento e o replanejamento são diferentes e independentes. Apesar do replanejamento ser uma ação que acontece após o planejamento devido a imprevistos na operação da fábrica, as heurísticas são completamente independentes entre si e o resultado prévio do planejamento não é utilizado durante o replanejamento. Isso acontece porque não é possível prever a magnitude de alteração do plano de produção durante o replanejamento e é mais provável que a geração de uma nova solução inicial factível seja mais rápida e eficiente do que transformar a solução anterior em um resultado factível.

De acordo com o problema foram estabelecidos para as heurísticas diferentes critérios de parada. Em acréscimo, em alguns casos foram propostas múltiplas versões alternativas para uma mesma heurística. ILP–Heur é dividida em duas versões conforme o modelo exato utilizado na etapa de geração da solução inicial (ILP–Heur Temporal e ILP–Heur em Estágios), cada um com duas versões internas referentes aos parâmetros de entrada (lista de submovimentos vazia ou proveniente de *BuildMoveListCombined*). LS–Heur também é dividida em duas versões (LS–Heur Determinística e LS–Heur Estocástica) de acordo com as vizinhanças selecionadas para a etapa de busca local. Todas as heurísticas e respectivas versões são consolidadas na Tabela 8.

Tabela 8 – Síntese das heurísticas propostas.

Grupo	**Problema**	**Estratégia**	**Crit. Par.**	**Heurística**	**Sub-heurística**
VNS–Heur	Planejamento	Meta-heurística VNS	1h	VNS–Heur	-
ILP–Heur	Planejamento	Modelo matemático + buscal local	4h	ILP–Heur Temporal ILP–Heur em Estágios	Lista vazia BuildMoveList-Combined
LS–Heur	Replanejamento	Busca local	5 min	LS–Heur Determinística LS–Heur Estocástica	- -

6 Resultados e Discussão

As heurísticas para planejamento e replanejamento, em todas as suas respectivas versões, foram aplicadas em um conjunto de 13 instâncias reais do problema de planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis da empresa Baker Hughes[1]. As mesmas instâncias foram utilizadas tanto para o planejamento quanto para o replanejamento, já que ambos devem considerar o mesmo horizonte de planejamento, alterando apenas o tempo de execução.

As instâncias foram coletadas em múltiplos dias e apesar delas serem da mesma fábrica, existem pequenas diferenças na configuração das posições, disposição das bobinas e necessidade de movimentação, dependendo do layout da fábrica no dia em que o plano foi recebido e das tarefas a serem executadas. As instâncias mencionadas podem ser consultadas em um repositório público disponível em [28].

Cada instância é composta de seis arquivos:

- Vértices: Define todas as posições da região interna da fábrica, incluindo posições de armazenamento, máquina e temporária. Também são incluídas uma ou mais posições externas para mapear a conexão entre a região externa e o CAR2.
- Arcos: Lista de caminhos permitidos entre duas posições adjacentes para cada unidade de movimentação.
- Posição inicial: Lista de todas as bobinas na região interna da fábrica no instante de tempo zero, assim como as suas respectivas posições. Caso alguma bobina a ser utilizada pelo plano de produção esteja na região externa, ela também é adicionada nesse arquivo nas posições externas mencionadas no primeiro item (não existe um mapeamento exato para posições externas).
- Carros: Define quais vértices representam os carros (CAR1 e CAR2).
- Posições bloqueadas: Define quais vértices representam as máquinas que constroem a primeira camada do tubo flexível e quais são as respectivas posições bloqueadas durante o deslocamento da bobina.
- Planejamento: Lista de tarefas correspondentes ao plano de produção, incluindo os tempos inicial e final, assim como as bobinas e posições de cada subtarefa. Também são incluídas tarefas que já estão em execução para bloquear as respectivas posições de máquina.

[1] https://www.bakerhughes.com/

Vale ressaltar que não existem arquivos específicos para as posições de máquina e temporárias. As posições de máquina são definidas dinamicamente conforme as posições do plano de produção e as temporárias são aquelas que contém apenas arestas para saída de bobina, mas nunca de entrada. Também é importante mencionar que não existem valores de referência para as instâncias coletadas e, portanto, a análise de qualidade dos resultados neste capítulo será executada a partir da comparação dos valores encontrados para as diferentes heurísticas.

Um pequeno descritivo das instâncias é apresentado na Tabela 9, indicando quantas bobinas estão inicialmente na fábrica e a quantidade de subtarefas em cada caso (sem discriminar subtarefas pré-posicionadas). Esse descritivo não representa necessariamente a complexidade de cada instância, mas pode auxiliar essa análise, já que quanto mais carregada a planta menos espaços livres para movimentação existem e que quanto mais subtarefas mais bobinas precisam ser deslocadas até a posição de destino.

Tabela 9 – Descritivo das instâncias

ID instância	Qtd. Bobinas	Qtd. Subtarefas
A	36	27
B	38	27
C	39	27
D	40	30
E	42	27
F	40	28
G	37	28
H	40	36
I	40	27
J	40	36
K	40	41
L	42	32
M	42	30

Para proteger informações sensíveis da empresa, as seguintes modificações, que não alteram as soluções ótimas, foram aplicadas às instâncias:

1. Os nomes das posições, bobinas e tarefas foram substituídos por identificadores numéricos.
2. Todos os tempos aplicados foram transformados de minutos para u.t. Essa transformação é executada multiplicando o intervalo de tempo por uma dada constante e arredondando para o valor inteiro imediatamente superior.

Em todas as heurísticas foram definidos tempos de deslocamento comuns para as unidades de movimentação em todas as instâncias:

1. Tempo de deslocamento CAR1 e CAR2: cinco u.t.
2. Tempo de deslocamento Ponte Rolante A e Ponte Rolante B: três u.t.

As heurísticas propostas foram implementadas em Python 3.7 e, nos casos com modelos matemáticos, foram integradas ao *software* de otimização CPLEX [2]. Os resultados foram obtidos a partir de execuções em um computador com 16 GB de memória RAM e processador *Intel Core i7-8565U*.

Os resultados obtidos para as heurísticas em todas as versões, assim com as respectivas análises críticas, são apresentados neste capítulo. Primeiramente os resultados para VNS–Heur e ILP–Heur são descritos na seção 6.1, seguidos pelos resultados de LS–Heur na seção 6.2.

6.1 Heurísticas para Planejamento

As heurísticas para planejamento foram propostas com o intuito de encontrar as melhores soluções possíveis para o planejamento de movimentação das bobinas. A subseção 6.1.1 apresenta os resultados para VNS–Heur e a subseção 6.1.2 descreve os resultados da ILP–Heur. Por fim, a subseção 6.1.3 realiza um comparativo entre todas as versões apresentadas para o planejamento a fim de identificar aquela com melhores resultados.

6.1.1 VNS–Heur

Devido à sua natureza estocástica, a heurística VNS–Heur foi executada dez vezes, durante uma hora, para cada instância. Os resultados obtidos são apresentados na Tabela 10. Para cada instância são reportadas quatro linhas: 1ª iteração, Melhor Caso, Pior Caso e Mediana. A 1ª iteração é o valor obtido logo antes da primeira execução do operador de perturbação (geração da solução inicial mais a primeira busca local), enquanto os demais valores são os resultados finais obtidos após todas as repetições.

As colunas F_1 até F_5 identificam os valores obtidos para os cinco objetivos do problema. Como explicado anteriormente, os objetivos F_1 e F_2 dizem respeito à factibilidade da solução e, portanto, o resultado só é factível caso os valores sejam iguais a zero. F_3, F_4 e F_5 dizem respeito ao desempenho. Conforme a composição lexicográfica, F_3 é o objetivo de desempenho mais importante e F_5 o menos. Para a otimização da solução, F_3 e F_5 devem ser minimizados e F_4 maximizado.

Os resultados finais de todas as repetições podem ser verificados no Apêndice A, mas algumas análises podem ser realizadas a partir dos resultados da Tabela 10:

- VNS–Heur é consistente nos resultados, uma vez que a heurística conseguiu encontrar resultado factível (F_1 e F_2 iguais a zero) para todas as instâncias, na primeira iteração e ao final, em todas as repetições.

[2] https://www.ibm.com/analytics/cplex-optimizer

Tabela 10 – Resultados encontrados pela VNS–Heur.

ID instância	**Tipo do Resultado**	F_1	F_2	F_3	F_4	F_5
A	1ª iteração	0	0	36	392	56
	Melhor Caso	0	0	36	395	52
	Pior Caso	0	0	36	395	54
	Mediana	0	0	36	395	53
B	1ª iteração	0	0	12	520	66
	Melhor Caso	0	0	6	535	71
	Pior Caso	0	0	9	520	68
	Mediana	0	0	6	520	70
C	1ª iteração	0	0	12	1057	73
	Melhor Caso	0	0	12	2460	61
	Pior Caso	0	0	12	1551	70
	Mediana	0	0	12	2457	65
D	1ª iteração	0	0	83	117	101
	Melhor Caso	0	0	70	123	105
	Pior Caso	0	0	78	438	100
	Mediana	0	0	75	120	99
E	1ª iteração	0	0	21	1643	85
	Melhor Caso	0	0	21	2191	105
	Pior Caso	0	0	21	2060	85
	Mediana	0	0	21	2093	94
F	1ª iteração	0	0	18	3678	92
	Melhor Caso	0	0	18	3705	91
	Pior Caso	0	0	18	3678	91
	Mediana	0	0	18	3681	89
G	1ª iteração	0	0	32	460	84
	Melhor Caso	0	0	23	1722	77
	Pior Caso	0	0	32	460	84
	Mediana	0	0	23	1548	84
H	1ª iteração	0	0	92	1948	74
	Melhor Caso	0	0	81	1954	70
	Pior Caso	0	0	88	1954	71
	Mediana	0	0	81	1954	73
I	1ª iteração	0	0	37	1783	50
	Melhor Caso	0	0	37	1786	49
	Pior Caso	0	0	37	1786	49
	Mediana	0	0	37	1786	49
J	1ª iteração	0	0	143	64	65
	Melhor Caso	0	0	143	64	61
	Pior Caso	0	0	143	64	65
	Mediana	0	0	143	64	65
K	1ª iteração	0	0	138	60	85
	Melhor Caso	0	0	130	60	75
	Pior Caso	0	0	134	60	88
	Mediana	0	0	132	60	77
L	1ª iteração	0	0	58	0	47
	Melhor Caso	0	0	52	0	44
	Pior Caso	0	0	58	0	47
	Mediana	0	0	52	0	44
M	1ª iteração	0	0	25	15	41
	Melhor Caso	0	0	8	15	36
	Pior Caso	0	0	11	15	41
	Mediana	0	0	8	15	37

- Ao comparar os valores finais com a 1ª iteração de cada instância, é possível perceber que em diversos casos existe uma melhora significativa após a execução de uma hora (exemplo: para a instância M, a mediana, melhor e pior caso são bem melhores do que a 1ª iteração). Entretanto, um ponto negativo da heurística é que em quatro instâncias a 1ª iteração foi mantida no pior caso.

- Considerando o resultado mediana para o objetivo de maior importância para o desempenho da solução conforme a composição lexicográfica (objetivo F_3 - atraso), VNS–Heur consegue aperfeiçoar a 1ª iteração, em média, 14 %.

- Com exceção da instância J, nos casos em que não houve diferença no objetivo de atraso entre a 1ª iteração e a mediana, sempre houve melhora do objetivo de adiantamento (F_4). Essa assertiva indica que a importância da etapa de perturbação somada à iteração de uma hora de VNS–Heur é maior do que os 14 % mencionados no item anterior.

- Em diversas instâncias a mediana se aproximou mais do melhor caso do que do pior caso, indicando que apesar de possíveis execuções ruins, em média os valores se aproximam mais do melhor caso.

- Existe uma diferença significativa entre alguns resultados de pior caso e mediana. Para evitar o pior caso de VNS–Heur como solução final, essa heurística poderia ser executada até quatro vezes para obter a melhor solução, respeitando o limite de tempo de quatro horas estipulado para o planejamento. Como a mediana, em geral, se aproxima mais do melhor caso do que do pior caso, mesmo executando quatro iterações em vez de dez, a heurística provavelmente retornaria uma solução similar ou igual ao melhor caso. Essa hipótese foi testada em uma nova execução e os resultados são apresentados na Tabela 11.

Os resultados da Tabela 11 comprovam a hipótese de que ao repetir VNS–Heur quatro vezes, os valores finais, em geral, se aproximam do melhor caso obtido anteriormente. Nove instâncias obtiveram resultado muito similar ou igual ao melhor caso, duas encontraram valores ainda melhores e apenas duas obtiveram resultados intermediários entre o melhor e o pior caso.

6.1.2 ILP–Heur

A heurística ILP–Heur foi executada uma vez, com critério de parada de quatro horas, para cada versão da heurística e cada instância. A subseção 6.1.2.1 apresenta os resultados de ILP–Heur Temporal e a subseção 6.1.2.2 descreve os resultados de ILP–Heur em Estágios, ambas para as duas versões de solução inicial (lista de submovimentos vazia e utilizando o operador *BuildMoveListCombined*).

Tabela 11 – Comparativo resultados VNS–Heur

ID instância	Tipo do Resultado	F_1	F_2	F_3	F_4	F_5
A	Melhor Caso (10 iter.)	0	0	36	395	52
	Pior Caso (10 iter.)	0	0	36	395	54
	Melhor Caso (4 iter.)	0	0	36	395	54
B	Melhor Caso (10 iter.)	0	0	6	535	71
	Pior Caso (10 iter.)	0	0	9	520	68
	Melhor Caso (4 iter.)	0	0	6	535	69
C	Melhor Caso (10 iter.)	0	0	12	2460	61
	Pior Caso (10 iter.)	0	0	12	1551	70
	Melhor Caso (4 iter.)	0	0	12	2789	67
D	Melhor Caso (10 iter.)	0	0	70	123	105
	Pior Caso (10 iter.)	0	0	78	438	100
	Melhor Caso (4 iter.)	0	0	74	120	102
E	Melhor Caso (10 iter.)	0	0	21	2191	105
	Pior Caso (10 iter.)	0	0	21	2060	85
	Melhor Caso (4 iter.)	0	0	21	2239	96
F	Melhor Caso (10 iter.)	0	0	18	3705	91
	Pior Caso (10 iter.)	0	0	18	3678	91
	Melhor Caso (4 iter.)	0	0	18	3681	92
G	Melhor Caso (10 iter.)	0	0	23	1722	77
	Pior Caso (10 iter.)	0	0	32	460	84
	Melhor Caso (4 iter.)	0	0	23	849	77
H	Melhor Caso (10 iter.)	0	0	81	1954	70
	Pior Caso (10 iter.)	0	0	88	1954	71
	Melhor Caso (4 iter.)	0	0	81	1954	72
I	Melhor Caso (10 iter.)	0	0	37	1786	49
	Pior Caso (10 iter.)	0	0	37	1786	49
	Melhor Caso (4 iter.)	0	0	37	1789	51
J	Melhor Caso (10 iter.)	0	0	143	64	61
	Pior Caso (10 iter.)	0	0	143	64	65
	Melhor Caso (4 iter.)	0	0	143	64	65
K	Melhor Caso (10 iter.)	0	0	130	60	75
	Pior Caso (10 iter.)	0	0	134	60	88
	Melhor Caso (4 iter.)	0	0	130	60	79
L	Melhor Caso (10 iter.)	0	0	52	0	44
	Pior Caso (10 iter.)	0	0	58	0	47
	Melhor Caso (4 iter.)	0	0	52	0	44
M	Melhor Caso (10 iter.)	0	0	8	15	36
	Pior Caso (10 iter.)	0	0	11	15	41
	Melhor Caso (4 iter.)	0	0	8	15	37

6.1.2.1 ILP–Heur Temporal

A Tabela 12 descreve os resultados da heurística ILP–Heur Temporal utilizando como parâmetro de entrada uma lista de submovimentos vazia. Para cada instância são reportados três valores. A primeira coluna apresenta o tempo de execução apenas do modelo temporal e a segunda coluna, dividida de F_1 até F_5, indica os valores obtidos para os cinco objetivos do problema logo após o modelo temporal. A terceira coluna indica os valores para os objetivos obtidos ao final da heurística, após a etapa de busca local.

Tabela 12 – Resultados ILP–Heur Temporal/Lista Vazia

Instância	T(s) Modelo Temporal	Resultado Modelo Temporal					Resultado Final				
		F_1	F_2	F_3	F_4	F_5	F_1	F_2	F_3	F_4	F_5
A	2686 - Falha	-	-	-	-	-	-	-	-	-	-
B	2662 - Falha	-	-	-	-	-	-	-	-	-	-
C	4949 - Falha	-	-	-	-	-	-	-	-	-	-
D	2427 - Falha	-	-	-	-	-	-	-	-	-	-
E	2414 - Falha	-	-	-	-	-	-	-	-	-	-
F	2556 - Falha	-	-	-	-	-	-	-	-	-	-
G	3230 - Falha	-	-	-	-	-	-	-	-	-	-
H	1379 - Falha	-	-	-	-	-	-	-	-	-	-
I	634	0	10	222	253	98	0	0	40	1766	50
J	6330 - Falha	-	-	-	-	-	-	-	-	-	-
K	5192 - Falha	-	-	-	-	-	-	-	-	-	-
L	338	0	9	322	0	104	0	0	60	2	43
M	2701	0	11	622	0	111	0	0	9	15	36

Para complementar os resultados, é possível verificar no Apêndice B, para cada instância, o tempo de execução do modelo temporal dividido por subtarefa e, nos casos de falha, qual subtarefa foi a responsável pela falha.

A Tabela 12 evidencia que apenas três de 13 instâncias foram solucionadas pelo modelo temporal. Isso indica que ILP–Heur Temporal com lista vazia de submovimentos como parâmetro de entrada, limite de 26 instantes de movimentação e tempo máximo de quarenta minutos por subtarefa não é uma boa heurística para o problema de movimentação de bobinas.

Uma justificativa para o desempenho da heurística é a qualidade da solução inicial utilizada. Como a bobina alvo não é deslocada até a posição destino da subtarefa, a solução inicial de lista vazia apesar de factível para o modelo temporal não é válida para a heurística. Assim, quando o modelo exato não consegue encontrar uma outra solução dentro de quarenta minutos, a heurística falha. Em acréscimo, o fato de otimizar o plano de produção por subtarefas e não globalmente, pode gerar soluções em uma determinada subtarefa que prejudicam as subtarefas seguintes. Essa situação aumenta a quantidade de instantes de movimentação necessários para o caminho mínimo, podendo inclusive, ultrapassar o limite de 26 instantes.

Mesmo com os resultados ruins, é possível realizar algumas análises adicionais para validar a importância das estratégias de redução do modelo adotadas. O primeiro teste é sobre como a quantidade de instantes de movimentação disponíveis afetam o tempo de execução da heurística. Já o segundo teste demonstra a relação entre o filtro inicial de posições e bobinas e o tempo de execução. A instância L foi selecionada para os testes por obter sucesso com menor tempo de execução e solucionar todas as subtarefas do plano de produção com apenas 15 instantes de movimentação.

Originalmente, ILP–Heur Temporal disponibiliza 15 instantes de movimentação e caso essa quantidade não seja suficiente, são disponibilizados 26 em uma segunda tentativa. Para analisar o tempo de execução, o primeiro teste propõe fixar 26 instantes de movimentação e, em

uma segunda iteração, 40. A heurística original executa o modelo temporal em 338 segundos para a instância L. Ao fixar 26 instantes de movimentação, são necessários 770 segundos e 4012 segundos para 40 instantes de movimentação. Ao executar o segundo teste com 15 instantes de movimentação e sem filtro de posições e bobinas, a heurística demanda 2310 segundos para solucionar o modelo temporal.

Ao fixar a quantidade de variáveis do modelo exato ou filtrar parâmetros de entrada a heurística restringe o espaço de busca disponível. Essa é uma característica negativa, já que em alguns casos é possível que uma solução ou a solução ótima não seja encontrada em virtude das modificações aplicadas. Todavia, ao diminuir a quantidade de variáveis fixadas e apresentar tempos de execução muito maiores que o original, os testes propostos comprovam e validam a importância de medidas que reduzam tamanho do modelo, mesmo retringindo o espaço de busca, a fim de alcançar um tempo de execução factível.

Utilizando o mesmo formato da tabela anterior, a Tabela 13 descreve os resultados da heurística ILP–Heur Temporal utilizando como parâmetro de entrada o operador *BuildMoveList-Combined*. Os tempos de execução de cada instância por subtarefa também são apresentados no Apêndice B.

Tabela 13 – Resultados ILP–Heur Temporal/BuildMoveListCombined

Instância	T(s) Modelo Temporal	Resultado Modelo Temporal					Resultado Final				
		F_1	F_2	F_3	F_4	F_5	F_1	F_2	F_3	F_4	F_5
A	5172	0	3	279	0	73	0	0	50	409	57
B	5714	0	10	210	405	104	0	0	9	529	61
C	8159	0	12	195	663	120	0	0	12	1251	83
D	5327 - Falha	-	-	-	-	-	-	-	-	-	-
E	12525	0	10	239	672	136	0	0	21	1652	84
F	14400 - Falha	-	-	-	-	-	-	-	-	-	-
G	6893	0	12	312	55	116	0	0	26	460	87
H	7569	0	8	315	44	110	0	0	81	1954	82
I	746	0	9	113	355	72	0	0	40	1766	50
J	3343	0	15	345	28	125	0	0	143	64	65
K	11402	0	18	586	11	147	0	0	153	60	84
L	575	0	11	206	0	93	0	0	60	2	43
M	9572	0	8	593	0	92	0	0	9	15	36

Enquanto a versão que utiliza uma lista de submovimentos vazia como parâmetro de entrada encontra resultados para apenas três de 13 instâncias, a versão com solução inicial proveniente de *BuildMoveListCombined* consegue finalizar a execução dentro do critério de parada de quatro horas em onze instâncias.

Dois motivos principais são responsáveis pela discrepante melhoria da segunda versão frente a primeira. Iniciar o modelo exato com uma solução factível e eficiente permite cortar diversas regiões do espaço de busca com valor objetivo inferior e assim reduzir o tempo de execução. Em acréscimo, utilizar uma quantidade de instantes de movimentação personalizada

por subtarefa, modifica a heurística para efetuar sempre uma única execução do modelo por subtarefa, reduz a quantidade de instantes de movimentação e o tempo de execução, principalmente quando a quantidade necessária está entre 15 e 26 instantes e, por fim, permite executar o modelo com mais de 26 instantes de movimentação em casos específicos.

Mesmo nesta segunda versão, com resultados visivelmente melhores, não é possível encontrar resultados para duas instâncias e em algumas o tempo de execução se aproxima do critério de parada. Nesse cenário e após a discussão sobre a importância do tamanho do modelo exato para o tempo de execução do algoritmo, é evidente a necessidade de dividir a otimização da construção do plano de movimentação das bobinas por subtarefas em vez de aplicar um único modelo para o plano de produção completo.

Como o modelo temporal é restrito por subtarefa e utiliza uma função objetivo própria, o resultado do modelo temporal é muito inferior ao resultado final da heurística. Como apresentado na coluna do objetivo F_2, a falta do conceito de operação no modelo temporal gera soluções com bobina posicionada nos carros ao final das operações em todas as instâncias. Em acréscimo, os objetivos F_3, F_4 e F_5 têm desempenho muito pior em todas as instâncias no modelo temporal. Essa comparação de valores destaca a importância da vizinhança *SubtaskSequence* e da etapa de busca local para otimizar a solução de forma completa e com funções objetivo que melhor descrevem o problema.

Por fim, vale ressaltar que o resultado do modelo temporal para cada instância é diferente entre as duas versões avaliadas nesta subseção. Isso pode acontecer caso existam duas ou mais soluções eficientes com o mesmo valor objetivo em algumas subtarefas ou caso o filtro de posições e bobinas resulte em diferentes parâmetros de entrada. Apesar dessa divergência no resultado do modelo temporal, não houve impacto no resultado final, que obteve exatamente o mesmo valor para as três instâncias onde as duas versões encontraram resultados.

6.1.2.2 ILP–Temporal em Estágios

A Tabela 14 descreve os resultados da heurística ILP–Heur em Estágios utilizando como parâmetro de entrada uma lista de submovimentos vazia. Para cada instância são reportados quatro valores. A primeira coluna apresenta o tempo de execução apenas do submodelo atemporal e a segunda coluna destaca o tempo de execução apenas do submodelo de encaixe temporal. Em seguida a terceira coluna, dividida de F_1 até F_5 , indica os valores obtidos para os cinco objetivos do problema logo após o modelo em estágios e a quarta coluna indica os valores para os objetivos obtidos ao final da heurística, após a etapa de busca local.

Para complementar os resultados, é possível verificar no Apêndice C, para cada instância, o tempo de execução do submodelo atemporal dividido por subtarefa e, nos casos de falha, qual subtarefa foi a responsável pela falha.

Tabela 14 – Resultados ILP–Heur em Estágios/Lista Vazia

Instância	Atemporal T(s)	Encaixe Temporal T(s)	Resultado Modelo Estágios					Resultado Final				
			F_1	F_2	F_3	F_4	F_5	F_1	F_2	F_3	F_4	F_5
A	2655	105	0	8	77	18	82	0	0	50	409	57
B	2588 - FALHA	-	-	-	-	-	-	-	-	-	-	-
C	4886 - FALHA	-	-	-	-	-	-	-	-	-	-	-
D	2423 - FALHA	-	-	-	-	-	-	-	-	-	-	-
E	5144 - FALHA	-	-	-	-	-	-	-	-	-	-	-
F	2527 - FALHA	-	-	-	-	-	-	-	-	-	-	-
G	2806 - FALHA	-	-	-	-	-	-	-	-	-	-	-
H	591 - FALHA	-	-	-	-	-	-	-	-	-	-	-
I	324	215	0	12	47	1588	86	0	0	40	1766	50
J	4024 - FALHA	-	-	-	-	-	-	-	-	-	-	-
K	2003 - FALHA	-	-	-	-	-	-	-	-	-	-	-
L	187	288	0	8	104	0	80	0	0	60	2	43
M	5293	247	0	10	1324	0	89	0	0	9	15	36

O submodelo atemporal é capaz de encontrar solução eficiente em quatro instâncias, contra três do modelo temporal. Apesar de dividir as responsabilidades do modelo temporal em dois submodelos potencialmente menores, assim como na ILP–Heur Temporal, a versão da heurística ILP–Heur em Estágios que utiliza como parâmetro de entrada uma lista de submovimentos vazia falha em diversas instâncias. Essas evidências indicam que as propostas de redução e modificação do modelo não são suficientes para tornar a versão com lista de submovimentos vazia em uma boa heurística para o problema de movimentação de bobinas.

Utilizando o mesmo formato da tabela anterior, a Tabela 15 descreve os resultados da heurística ILP–Heur em Estágios utilizando como parâmetro de entrada o operador *BuildMoveListCombined*. Os tempos de execução do submodelo temporal de cada instância por subtarefa também são apresentados no Apêndice C.

Tabela 15 – Resultados ILP–Heur em Estágios/BuildMoveListCombined

Instância	Atemporal T(s)	Encaixe Temporal T(s)	Resultado Modelo Estágios					Resultado Final				
			F_1	F_2	F_3	F_4	F_5	F_1	F_2	F_3	F_4	F_5
A	305	71	0	5	94	0	87	0	0	50	409	57
B	4999	537	0	5	15	514	105	0	0	9	529	61
C	10242	1495	0	12	41	-799	118	0	0	35	1255	84
D	5093 - FALHA	-	-	-	-	-	-	-	-	-	-	-
E	7453	1009	0	10	42	969	123	0	0	21	1631	89
F	9261	FALHA	-	-	-	-	-	-	-	-	-	-
G	4032	837	0	10	114	428	121	0	0	32	460	90
H	3532	796	0	14	218	820	125	0	0	81	1954	82
I	193	157	0	11	47	489	80	0	0	40	1766	50
J	235	709	0	16	216	33	125	0	0	143	64	65
K	5069	610	0	20	261	51	135	0	0	140	60	77
L	144	434	0	11	93	0	85	0	0	60	2	43
M	438	90	0	7	184	15	65	0	0	9	15	36

Assim como na ILP–Heur Temporal, ILP–Heur em Estágios utilizando *BuildMoveList-Combined* consegue finalizar com sucesso, dentro do critério de parada de quatro horas, 11 de 13 instâncias e as justificativas de aumento de desempenho entre as duas versões de solução inicial são as mesmas. Entretanto, é válido realizar um comparativo de valores finais entre as duas heurísticas.

Ao analisar valor objetivo final, as duas heurísticas são muito parecidas. Em nove instâncias elas têm o mesmo ou quase o mesmo resultado final e nas outras quatro, ILP–Heur Temporal é superior em duas e ILP–Heur em Estágios é superior em duas. Ao analisar tempo de execução final, apenas em quatro instâncias o valor é similar e em oito casos ILP–Heur em Estágios é superior.

Esses valores validam a hipótese de que ao dividir em dois submodelos menores, em geral, a heurística pode ser executada mais rapidamente. Em contrapartida, os valores não confirmam uma possível melhoria na qualidade da solução final ao otimizar de forma global durante o submodelo de encaixe temporal. Essa falta de aumento de desempenho provavelmente está vinculada ao fato da qualidade do resultado do modelo exato ser muito inferior ao resultado final e também, devido ao limite imposto ao encaixe temporal de apenas reorganizar os submovimentos previamente estabelecidos pelo submodelo atemporal.

6.1.3 Comparação de resultados

Nesta seção, os resultados das heurísticas de planejamento foram apresentados divididos conforme os grupos de heurísticas. Primeiramente foram descritos os resultados de VNS–Heur, definida por apenas uma versão, seguidos pelos resultados de ILP–Heur, divididos por modelo exato e versão da solução inicial.

Ao discorrer sobre os resultados de ILP–Heur, as quatro opções propostas foram comparadas. As versões que utilizam lista de submovimentos vazia como parâmetro de entrada são capazes de solucionar poucas instâncias. Entre as versões que utilizam a solução inicial do operador *BuildMoveListCombined* como parâmetro de entrada, apenas duas instâncias são não finalizadas com sucesso. Ao comparar esses últimos resultados, o valor final para os objetivos F_1 a F_5 são muito semelhantes para ILP–Heur Temporal e ILP–Heur em Estágios, mas o tempo de execução da heurística com modelo em estágios é melhor.

Por essa razão, esta subseção de comparação de resultados entre VNS–Heur e ILP–Heur para o problema de planejamento utiliza como base os valores de mediana para VNS–Heur e os valores encontrados para ILP–Heur em Estágios com solução inicial proveniente de *BuildMoveListCombined*. Para facilitar a análise, os resultados previamente apresentados são repetidos na Tabela 16. Para cada instância e heurística são descritos os valores encontrados para os objetivos F_1 a F_5 e o tempo de execução final.

Tabela 16 – Comparação de resultados VNS–Heur e ILP–Heur em Estágios

Instância	VNS–Heur						ILP–Heur em Estágios					
	F_1	F_2	F_3	F_4	F_5	T(s)	F_1	F_2	F_3	F_4	F_5	T(s)
A	0	0	36	395	53	3600	0	0	50	409	57	376
B	0	0	6	520	70	3600	0	0	9	529	61	5536
C	0	0	12	2457	65	3600	0	0	35	1255	84	11737
D	0	0	75	120	99	3600	-	-	-	-	-	-
E	0	0	21	2093	94	3600	0	0	21	1631	89	8462
F	0	0	18	3681	89	3600	-	-	-	-	-	-
G	0	0	23	1548	84	3600	0	0	32	460	90	4869
H	0	0	81	1954	73	3600	0	0	81	1954	82	4328
I	0	0	37	1786	49	3600	0	0	40	1766	50	350
J	0	0	143	64	65	3600	0	0	143	64	65	944
K	0	0	132	60	77	3600	0	0	140	60	77	5679
L	0	0	52	0	44	3600	0	0	60	2	43	578
M	0	0	8	15	37	3600	0	0	9	15	36	528

Além de falhar em duas instâncias, ILP–Heur em Estágios não obteve valor objetivo melhor que VNS–Heur em nenhum caso de teste. Ao comparar o tempo de execução, a heurística com componentes exatos conseguiu ser mais rápida que VNS–Heur em cinco instâncias.

Essa análise permite concluir que VNS–Heur é a heurística mais indicada para o planejamento, mas que existem possíveis pontos de melhorias em relação ao critério de parada. Em vez de fixar um único critério de parada de uma hora de execução, VNS–Heur também poderia encerrar após uma análise de estabilidade da solução (finalizar após uma determinada quantidade de iterações sem modificação no melhor desempenho).

Como reportado na Tabela 10, existe uma diferença significativa entre alguns resultados de pior caso e mediana. Os valores obtidos na Tabela 11 indicam que o pior caso pode ser evitado ao executar VNS–Heur quatro vezes para obter a melhor solução. A sugestão de melhoria seria ainda mais eficiente caso as quatro repetições pudessem ser executadas em paralelo.

Unindo múltiplas repetições em paralelo com a adição de um novo critério de parada, provavelmente VNS–Heur conseguiria obter um valor mais consistente em um tempo de execução inferior a uma hora.

6.2 Heurística para Replanejamento

As heurísticas para replanejamento foram propostas com o intuito de encontrar, dentro de um limite de cinco minutos, boas soluções para o replanejamento de movimentação das bobinas. A subseção 6.2.1 apresenta os resultados para LS–Heur Determinística e a subseção 6.2.2 descreve os resultados da LS–Heur Estocástica. Por fim, a subseção 6.2.3 realiza um comparativo entre as versões apresentadas para o replanejamento a fim de identificar aquela com melhores resultados.

6.2.1 LS–Heur Determinística

Os resultados de duas execuções diferentes da LS–Heur Determinística são sempre os mesmos dado que o algoritmo não tem nenhum componente estocástico. Entretanto, como critério de parada de cinco minutos é um fator muito importante no replanejamento, os testes contemplaram três execuções da heurística em cada instância para estimar o tempo de execução, que pode variar de execução para execução devido ao processamento multitarefa do sistema operacional. Os resultados obtidos são apresentados na Tabela 17.

Tabela 17 – Resultados encontrados pela LS–Heur Determinística.

ID instância	Tipo do Resultado	F_1	F_2	F_3	F_4	F_5	T(s)
A	Solução Inicial	0	0	36	392	62	
	Completo	0	0	36	392	56	36
B	Solução Inicial	0	0	21	520	65	
	Completo	0	0	12	520	66	72
C	Solução Inicial	0	0	12	1057	76	
	Completo	0	0	12	1057	73	63
D	Solução Inicial	0	0	96	120	110	
	Completo	0	0	83	117	101	199
E	Solução Inicial	0	0	27	1495	87	
	Completo	0	0	21	1513	81	168
F	Solução Inicial	0	0	18	3516	87	
	Completo	0	0	18	3605	102	307
G	Solução Inicial	0	0	44	74	97	
	Completo	0	0	32	460	80	213
H	Solução Inicial	0	0	112	1941	78	
	Completo	0	0	92	1945	79	147
I	Solução Inicial	0	0	40	1444	55	
	Completo	0	0	37	1780	50	50
J	Solução Inicial	0	0	215	33	78	
	Completo	0	0	175	61	70	177
K	Solução Inicial	0	0	191	54	94	
	Completo	0	0	145	60	82	187
L	Solução Inicial	0	0	72	0	59	
	Completo	0	0	58	0	47	24
M	Solução Inicial	0	0	34	15	38	
	Completo	0	0	19	15	39	47

Para cada instância são reportados dois tipos de resultados: Solução Inicial (resultado após a etapa de geração da solução inicial) e Completo (resultado ao final da heurística). As colunas F_1 até F_5 identificam os valores obtidos para os cinco objetivos do problema e a coluna T(s) indica o tempo necessário para o algoritmo convergir no pior caso das três execuções (em segundos). Caso o critério de parada de cinco minutos seja atingido, são reportados o valor e o tempo obtidos até este momento.

O teste de todas as instâncias foi finalizado em menos de cinco minutos, com exceção da instância F. Nesse caso, a heurística foi executada até a vizinhança *AnticipateSubtask*, quando atigiu o critério de parada e deixou sem executar a vizinhança *OperationsSwap*.

Algumas análises podem ser realizadas a partir dos resultados da Tabela 17:

- LS–Heur Determinística é consistente nos resultados, uma vez que a heurística conseguiu encontrar resultado factível (F_1 e F_2 iguais a zero) para todas as instâncias, nas soluções inicial e completa.
- Ao comparar os valores finais com a solução inicial de cada instância, é possível verificar que a etapa de busca local sempre conseguiu aperfeiçoar a solução inicial em pelo menos um dos objetivos relacionados ao desempenho. Vale ressaltar que em dez instâncias, a etapa de busca local conseguiu melhorar o objetivo de desempenho de maior relevância (F_3), conforme a composição lexicográfica.
- O tempo de execução é um fator muito importante no problema de replanejamento, já que a fábrica precisa paralisar novas movimentações enquanto o algoritmo é executado. Ao analisar a coluna T(s) é possível identificar que seis instâncias são finalizadas com menos de 100 segundos, cinco com menos de 200 segundos e apenas duas entre 200 e 300 segundos. Esses dados validam a eficácia de LS–Heur Determinística em encontrar soluções dentro ou até mais rápido que o limite de tempo estipulado.

Para aprofundar a análise de como as instâncias e seus respectivos tamanhos influenciam no desempenho da heurística, foram construídas instâncias alternativas variando a quantidade inicial de bobinas na fábrica. Para cada instância original foram geradas duas variantes com quantidade fixa de bobinas, a primeira com 26 e a segunda com 33. Para garantir a factibilidade da instância, foram retiradas apenas bobinas que não eram utilizadas pelo plano de produção. Estas instâncias adicionais também foram adicionadas no repositório público mencionado no início do capítulo.

O resultado comparativo a partir da variação da quantidade de bobinas pode ser visualizado na Tabela 18.

Para cada instância são reportados três resultados, dois alternativos e um original. Para cada resultado são informados a quantidade inicial de bobinas utilizada e a quantidade de operações da lista de movimentos resultante, assim como o respectivo tempo de execução. Para o resultado original os valores do tipo completo da Tabela 17 foram apenas repetidos com o objetivo de facilitar a comparação.

Ao considerar uma mesma instância, para quase todas as instâncias, a quantidade de operações e o tempo de execução aumentam conforme o aumento da quantidade de bobinas. Esses dados indicam que, em geral, quanto mais as posições da fábrica estão ocupadas, mais díficil é construir a rota de movimentação das bobinas, necessitando de mais operações e demandando um tempo de execução maior da heurística. Para os casos de exceção, como a instância G, que demanda mais operações para a alternativa com 26 bobinas do que para a alternativa com 33

bobinas, a discrepância provavelmente acontece por uma dificuldade da heurística em percorrer o espaço de busca com as vizinhanças propostas e encontrar uma solução melhor nas condições estabelecidas.

Entretanto, não é possível realizar uma associação direta entre a quantidade inicial de bobinas na fábrica e a quantidade de operações resultantes ou o tempo de execução. Enquanto a instância B gera 52 operações utilizando 26 bobinas, a instância M é capaz de encontrar uma solução com apenas 21 operações para a mesma quantidade de bobinas. Essa análise pode indicar que a complexidade do problema não está necessariamente associada ao carregamento da planta, mas na complexidade de se realizar cada movimento.

Tabela 18 – Resultados comparativos conforme quantidade de bobinas.

ID instância	Tipo da Instância	Qtd. Bobinas	Qtd. Operações	T(s)
A	Alternativa	26	41	22
	Alternativa	33	47	20
	Original	36	56	36
B	Alternativa	26	52	24
	Alternativa	33	60	48
	Original	38	66	72
C	Alternativa	26	40	46
	Alternativa	33	58	55
	Original	39	73	63
D	Alternativa	26	65	75
	Alternativa	33	79	106
	Original	40	101	199
E	Alternativa	26	59	73
	Alternativa	33	66	79
	Original	42	81	168
F	Alternativa	26	58	137
	Alternativa	33	78	190
	Original	40	102	207
G	Alternativa	26	72	119
	Alternativa	33	68	159
	Original	37	80	213
H	Alternativa	26	46	84
	Alternativa	33	57	99
	Original	40	79	147
I	Alternativa	26	34	33
	Alternativa	33	35	34
	Original	40	50	50
J	Alternativa	26	46	91
	Alternativa	33	46	102
	Original	40	70	177
K	Alternativa	26	60	195
	Alternativa	33	67	100
	Original	40	82	187
L	Alternativa	26	33	14
	Alternativa	33	38	16
	Original	42	47	24
M	Alternativa	26	21	7
	Alternativa	33	24	11
	Original	42	39	47

Em geral, quanto mais bobinas mais complexas são as operações. Todavia, mesmo que a planta esteja muito carregada, se todas as subtarefas forem simples (e.g. mover a bobina para uma posição de máquina ao lado), a complexidade e quantidade de operações, assim como o tempo de execução serão baixos. Para demonstrar que a quantidade inicial de bobinas tem impacto no desempenho das heurísticas mas não é o fator primariamente determinante para a complexidade das instâncias, foi realizado um teste adicional com a instância B na versão com 26 bobinas.

O plano de produção foi alterado três vezes, trocando as bobinas das subtarefas, a fim de controlar a quantidade média de operações necessária para concluir o planejamento. Em média, na primeira versão são necessárias duas operações para completar cada subtarefa e esse valor é ligeiramente aumentado nas outras duas versões. Os resultados deste teste estão disponíveis na Tabela 19.

Tabela 19 – Resultados de complexidade versus média de operações.

ID instância	Média operações	Qtd. bobinas	F_1	F_2	F_3	F_4	F_5	T(s)
B	2,1	26	0	0	6	460	44	20
	2,9	26	0	0	9	523	64	57
	3,3	26	0	0	26	514	72	112

Conforme esperado, a quantidade de operações (F_5) e o tempo de execução aumentaram de acordo com o aumento da média das operações. O resultado obtido pela LS–Heur Determinística para o atraso (F_3) também piorou de acordo com o aumento da complexidade, indicando não uma falha da heurística, mas que potencialmente configurações iniciais mais complexas geram listas de movimentos com mais atraso.

6.2.2 LS–Heur Estocástica

Ao adicionar a vizinhança aleatória *RandomSearch* a heurística LS–Heur Determinística é transformada na heurística LS–Heur Estocástica. Para encontrar os resultados dessa nova versão, LS–Heur Estocástica foi executada dez vezes para cada instância e os resultados obtidos são apresentados na Tabela 20. Os resultados finais de todas as repetições podem ser verificados no Apêndice D.

Nas diversas repetições de uma mesma instância, é possível encontrar diferentes valores objetivo e tempos de execução. Para cada instância são reportadas na tabela indicada acima três linhas: Melhor Caso, Pior Caso e Mediana, com os respectivos resultados obtidos ao final da execução. O tempo de execução apresentado é relativo à uma repetição. Esses três resultados foram classificados a partir do valor objetivo de todas as repetições e o tempo de execução foi utilizado apenas como critério de desempate na classificação.

Tabela 20 – Resultados encontrados pela LS–Heur Estocástica.

ID instância	Tipo do Resultado	F_1	F_2	F_3	F_4	F_5	T(s)
A	Melhor Caso	0	0	36	395	54	108
	Pior Caso	0	0	36	392	56	77
	Mediana	0	0	36	395	55	114
B	Melhor Caso	0	0	12	520	65	143
	Pior Caso	0	0	12	520	67	142
	Mediana	0	0	12	520	66	140
C	Melhor Caso	0	0	12	1057	73	103
	Pior Caso	0	0	12	1057	73	103
	Mediana	0	0	12	1057	73	103
D	Melhor Caso	0	0	74	117	99	297
	Pior Caso	0	0	83	117	101	237
	Mediana	0	0	83	117	100	301
E	Melhor Caso	0	0	21	1848	99	292
	Pior Caso	0	0	21	1513	82	234
	Mediana	0	0	21	1771	83	246
F	Melhor Caso	0	0	18	3606	97	322
	Pior Caso	0	0	18	3592	92	326
	Mediana	0	0	18	3601	91	305
G	Melhor Caso	0	0	29	864	82	259
	Pior Caso	0	0	38	527	91	199
	Mediana	0	0	32	460	80	250
H	Melhor Caso	0	0	81	1954	78	291
	Pior Caso	0	0	92	1944	77	202
	Mediana	0	0	91	1951	76	230
I	Melhor Caso	0	0	37	1789	53	126
	Pior Caso	0	0	37	1579	52	118
	Mediana	0	0	37	1624	59	124
J	Melhor Caso	0	0	137	34	63	253
	Pior Caso	0	0	169	61	65	290
	Mediana	0	0	141	34	63	240
K	Melhor Caso	0	0	135	44	79	220
	Pior Caso	0	0	160	54	88	259
	Mediana	0	0	142	57	78	280
L	Melhor Caso	0	0	60	0	51	104
	Pior Caso	0	0	61	0	51	107
	Mediana	0	0	60	0	54	103
M	Melhor Caso	0	0	11	15	44	122
	Pior Caso	0	0	20	15	44	120
	Mediana	0	0	18	15	44	126

Algumas análises podem ser realizadas a partir dos resultados:

- LS–Heur Estocástica é consistente nos resultados, uma vez que a heurística conseguiu encontrar resultado factível (F_1 e F_2 iguais a zero) em todas as repetições.
- Em algumas instâncias o critério de parada de cinco minutos foi atingido ou quase atingido. Essa marca indica que a heurística está no limite para o tempo de execução e em algumas situações o resultado pode ser comprometido pela falta de tempo hábil para executar todas as vizinhanças.

- Ao contrário de VNS–Heur em que a mediana estava mais próxima do melhor caso em diversas instâncias, na LS–Heur Estocástica, a mediana ficou mais dividida entre os melhores e piores valores. Esse resultado pode indicar uma menor estabilidade da heurística, já que ela é executada em apenas uma iteração e a vizinhança aleatória é repetida menos vezes.

Uma sugestão de melhoria para LS–Heur Estocástica seria realizar múltiplas execuções em paralelo para obter o melhor resultado ao fim de todas as execuções ou ao fim de cinco minutos. Essa sugestão poderia reduzir o impacto da análise apresentada nos últimos dois itens.

6.2.3 Comparação de resultados

Nesta seção, os resultados das heurísticas de replanejamento foram apresentados divididos pelas duas versões da heurística LS–Heur: LS–Heur Determinística e LS–Heur Estocástica. Ao comparar os resultados destas duas versões é possível perceber que:

- Os valores de mediana da versão estocástica, são superiores ou iguais aos da versão determinística em dez instâncias. Entretanto, vale ressaltar que ao analisar o pior caso, LS–Heur Estocástica só é melhor que LS–Heur Determinística em uma instância.
- Os tempos de execução da versão estocástica são maiores que o da versão determinística em todas as instâncias.

A hipótese que originou a proposta da heurística LS–Heur Estocástica considerava que, mesmo adicionando uma componente aleatória, na maior parte das execuções a heurística deve gerar um resultado igual ou melhor que a versão determinística. Essa hipótese é verdadeira ao analisar os resultados obtidos para o melhor caso e a mediana. Entretanto, como uma perturbação aleatória pode, às vezes, gerar uma degradação de desempenho, o pior caso de algumas instâncias gerou resultados inferiores ao da versão determinística.

Apesar de resultados razoáveis para a mediana, ao considerar que a heurística também pode produzir valores próximos do pior caso e que, para o tempo de execução, que é um ponto crítico do problema de replanejamento, LS–Heur Estocástica é sempre pior que LS–Heur Determinística, é possível indicar LS–Heur Determinística como a heurística mais adequada para o replanejamento. Todavia, como a versão estocástica é capaz de alcançar valores superiores em algumas instâncias, fica claro a possibilidade de melhorias na versão determinística para que ela também seja capaz de encontrar tais valores sem a adição de componentes aleatórios. Vale ressaltar também a sugestão de melhoria para LS–Heur Estocástica realizar múltiplas execuções em paralelo, que pode melhorar o desempenho desta heurística e diminuir a ocorrência do pior caso.

Como uma última análise, é válido comparar os resultados obtidos para os problemas de planejamento e replanejamento. A Tabela 21 apresenta os valores alcançados neste trabalho para as melhores heurística de cada problema: VNS–Heur (valores de mediana) e LS–Heur Determinística. Para cada instância e heurística são descritos os valores encontrados para os objetivos F_1 a F_5 e o tempo de execução final.

Tabela 21 – Comparação de resultados VNS–Heur e LS–Heur Determinística

Instância	VNS–Heur						LS–Heur Determinística					
	F_1	F_2	F_3	F_4	F_5	T(s)	F_1	F_2	F_3	F_4	F_5	T(s)
A	0	0	36	395	53	3600	0	0	36	392	56	36
B	0	0	6	520	70	3600	0	0	12	520	66	72
C	0	0	12	2457	65	3600	0	0	12	1057	73	63
D	0	0	75	120	99	3600	0	0	83	117	101	199
E	0	0	21	2093	94	3600	0	0	21	1513	81	168
F	0	0	18	3681	89	3600	0	0	18	3605	102	307
G	0	0	23	1548	84	3600	0	0	32	460	80	213
H	0	0	81	1954	73	3600	0	0	92	1945	79	147
I	0	0	37	1786	49	3600	0	0	37	1780	50	50
J	0	0	143	64	65	3600	0	0	175	61	70	177
K	0	0	132	60	77	3600	0	0	145	60	82	187
L	0	0	52	0	44	3600	0	0	58	0	47	24
M	0	0	8	15	37	3600	0	0	19	15	39	47

Com exceção da instância A, VNS–Heur alcançou melhores resultados que LS–Heur Determinística em todas as instâncias e conseguiu valores 15% melhores que os de LS–Heur para o objetivo F_3. Em acréscimo, nos casos em que F_3 foi igual para as duas heurísticas, VNS–Heur conseguiu ser mais eficiente no objetivo F_4.

Esse resultado demonstra a importância da etapa de perturbação e re-execução da busca local para explorar novas regiões do espaço de soluções e obter melhor desempenho. Todavia, ao comparar o tempo de execução das heurísticas e verificar que LS–Heur Determinística demanda apenas 8% do tempo utilizado por VNS–Heur, fica claro que ambas as heurísticas são adequadas ao seu respectivo propósito: resultados mais eficientes com folga no tempo de execução para o planejamento e tempo de execução restrito em detrimento do melhor desempenho para o replanejamento.

Em uma situação onde o replanejamento não é muito recorrente, a fábrica de tubos flexíveis poderia optar por utilizar o algoritmo de replanejamento para reiniciar a fábrica quando surgisse alguma alteração no plano de produção ou imprevisto na operação. Em seguida, com a fábrica reiniciada, o algoritmo de planejamento poderia ser executado durante uma hora com condição inicial projetada para daqui uma hora. Dessa maneira, também seria possível garantir resultados mais eficientes uma hora após o replanejamento, em vez de apenas uma vez por dia, durante o planejamento diário.

7 Conclusão

Este trabalho descreve o problema de elaborar planos de movimentação de bobinas entre máquinas em uma fábrica de tubos flexíveis. Apesar de ser possível aplicar o estudo em diferentes fábricas com condições de operação semelhantes, o problema é definido de acordo com as necessidades de uma fábrica específica da empresa Baker Hughes.

Nessa fábrica são gerados planos de produção que indicam quais tubos flexíveis, em quais máquinas e quando eles devem ser produzidos, além de quais bobinas são responsáveis por armazená-los e transportá-los ao longo do horizonte de tempo considerado. Em sequência, é necessário construir o plano de movimentação das bobinas que armazenam os tubos para concretização do plano de produção. Com o plano de movimentação, os operadores da fábrica sabem quando devem deslocar cada bobina e por qual caminho. Devido às limitações físicas da fábrica, existem diversas restrições para o deslocamento das bobinas e, por isso, é importante que o plano de movimentação seja elaborado de forma otimizada para garantir que todas as tarefas sejam atendidas com atraso mínimo e adiantamento máximo, na menor quantidade de operações. Vale ressaltar também a importância de se evitar *deadlocks*, que bloqueiam a movimentação de bobinas em uma determinada região da fábrica.

O plano de movimentação é gerado diariamente de acordo com a atualização do plano de produção, mas devido a imprevistos, também é necessário que ele sofra alterações ao longo do dia. Nesse contexto, este trabalho propõe duas abordagens, que consideram o mesmo horizonte de planejamento de cinco a dez dias, para tratar o problema. A primeira abordagem, voltada ao planejamento, inclui heurísticas com tempo de execução máximo de quatro horas, enquanto a segunda, destinada ao replanejamento, é baseada em heurísticas com critério de parada de cinco minutos. Todas as heurísticas propostas utilizam componentes compartilhados para construção de caminhos mínimos e exploração do espaço de busca, além de serem divididas em duas etapas: geração da solução inicial e busca local.

Entre as heurísticas para planejamento, VNS–Heur foi proposta a partir de vizinhanças de busca local específicas para o problema. Ao final da etapa de busca local, aplica-se um operador de perturbação, responsável por gerar uma nova solução incumbente e iniciar a busca local novamente. Esse fluxo é repetido durante uma hora. Uma segunda opção para otimizar o planejamento são as heurísticas ILP–Heur, que incorporam modelos matemáticos durante a geração da solução inicial. ILP–Heur é dividida em duas heurísticas conforme o modelo matemático: ILP–Heur Temporal e ILP–Heur em Estágios, que também são subdivididas em duas versões conforme os parâmetros de entrada utilizados: lista de submovimentos vazia ou proveniente do operador *BuildMoveListCombined*.

Como a dimensão do problema é muito grande, os modelos matemáticos além de serem aplicados apenas durante a geração da solução inicial, também são utilizados por subtarefa e a solução inicial completa é gerada a partir da concatenação dos resultados individuais. Apesar de otimizar no contexto da subtarefa, ainda são necessárias outras estratégias para redução da dimensão do modelo, como quantidade limitada de variáveis, filtro de parâmetros de entrada e solução inicial para restringir o espaço de busca. Essa tentativa de redução do tempo de execução de ILP–Heur é a origem para a divisão em duas heurísticas com duas versões cada.

LS–Heur Determinística e LS–Heur Estocástica são as duas heurísticas propostas para o replanejamento. Elas são muito semelhantes, ao utilizar as mesmas vizinhanças de busca local, mas a versão estocástica adiciona uma vizinhança com comportamento aleatório, na intenção de explorar novas regiões do espaço de busca. Para ambas heurísticas o critério de parada é um fator muito importante, já que a fábrica precisa pausar a movimentação enquanto o algoritmo é executado.

As heurísticas para planejamento e replanejamento, em todas as suas respectivas versões, foram aplicadas em um conjunto de 13 instâncias. Entre as versões de ILP–Heur, aquela que mais se destaca é ILP–Heur em Estágios utilizando o operador *BuildMoveListCombined*. As versões com lista de submovimentos vazia conseguem solucionar poucas instâncias e entre as duas versões com o operador *BuildMoveListCombined*, ambas têm valores objetivo similares, mas a heurística em estágios tem tempo de execução médio inferior. Ao comparar VNS–Heur e ILP–Heur, VNS–Heur é selecionada como a heurística recomendada para o planejamento, por conseguir encontrar soluções viáveis para todas as instâncias, obter valores objetivo melhores e demandar um tempo de execução inferior.

Ao comparar o desempenho das heurísticas de replanejamento, apesar de ser possível encontrar resultados superiores no melhor caso de algumas instâncias usando LS–Heur Estocástica, sugere-se o uso de LS–Heur Determinística, pela capacidade de encontrar valores razoáveis e constantes em um menor tempo de execução.

Mesmo entre as heurísticas selecionadas para o planejamento e replanejamento, é possível propor melhorias para trabalhos futuros. Para aperfeiçoar a heurística VNS–Heur, próximos trabalhos podem:

- Experimentar adicionar um critério de parada de acordo com a estabilidade da solução e talvez reduzir o tempo de execução de uma hora sem piorar os valores já encontrados.
- Realizar múltiplas repetições em paralelo de VNS–Heur para diminuir a incidência do pior caso, sem aumentar o tempo de execução.
- Testar outras formas de perturbar a solução candidata. Uma alternativa seria aumentar a perturbação à medida em que não são geradas soluções melhores e voltar ao seu nível mínimo quando uma solução melhor for encontrada.

- Como a melhor ordem de vizinhanças para uma instância não é necessariamente a melhor ordem para outra instância, definir a ordenação dos procedimentos de busca local de forma adaptativa utilizando aprendizado por reforço ou alguma outra técnica de aprendizado.

- Testar outras variantes do método de busca local VND, como BVND (Basic VND), que retorna à primeira vizinhança sempre que há melhora na solução corrente, garantindo que a solução final retornada pelo método seja um ótimo local com relação a todas as vizinhanças.

Para garantir resultados melhores e mais estáveis em LS–Heur, trabalhos futuros podem tentar adicionar novas vizinhanças que alcancem os resultados do melhor caso da versão estocástica sem a adição de componentes aleatórios ou podem paralelizar a execução de múltiplas repetições de LS–Heur estocástica para tentar reduzir a incidência do pior caso. Uma análise adicional que pode ser realizada e potencialmente transformada em uma nova vizinhança de busca local é construção de um método para detecção de *deadlocks* nos planos de movimentação que não levam à interrupção do algoritmo. Esse método deve identificar quando o atraso não é gerado pela complexidade da movimentação mas porque o caminho até a posição de máquina está bloqueado enquanto outras tarefas não forem concluídas.

Como as referências bibliográficas enumeram diversos problemas semelhantes ao planejamento de movimentação de bobinas que foram tratados com outros métodos heurísticos, outros estudos também podem tentar tratar este mesmo problema com outros métodos, como algoritmos genéticos.

Outros dois estudos que podem aperfeiçoar o desempenho da produção são sugeridos para a fábrica de tubos flexíveis. No primeiro, sugere-se experimentar outras posições para os carros ou até para a configuração da fábrica como um todo (posições, máquinas e unidades de movimentação). A partir da elaboração de um algoritmo capaz de otimizar o layout da fábrica, no caso de obras na infraestrutura atual ou possível construção de novas fábricas, a nova configuração poderia melhorar o desempenho da produção. A segunda recomendação é unificar as atividades de sequenciamento/agendamento das tarefas, atribuição das bobinas e construção do plano de movimentação. Um estudo capaz de construir um algoritmo que otimize todos esses problemas em conjunto tem grande potencial para aperfeiçoar o plano de movimentação das bobinas, gerando instâncias menos complexas, com uma média de operações por caminho mínimo menor.

Por fim, vale ressaltar que a heurística determinística desenvolvida para replanejamento foi implantada na fábrica de tubos flexíveis. Como ponto positivo para o algoritmo, é possível mencionar a capacidade de gerar movimentos que facilitem deslocamentos posteriores que normalmente, devido a complexidade de planejar manualmente diversas tarefas em simultâneo, não são realizados quando o planejamento é feito manualmente. Em contrapartida, um ponto negativo é a restrição do algoritmo em movimentar bobinas para a área externa da fábrica. Esses

movimentos podem ser executados pelos operadores e em algumas situações podem facilitar deslocamentos mais complexos. Entretanto, quando os movimentos para a área externa são executados, a lista de movimentos prévia proveniente da heurística é invalidada e o algoritmo necessita de uma nova execução para sincronização do posicionamento das bobinas.

Referências

[1] Tiziano Bacci, Sara Mattia, and Paolo Ventura. A branch-and-cut algorithm for the restricted block relocation problem. *European Journal of Operational Research*, 287:452–459, 2020.

[2] Eduardo G. Carrano, Gisele P. da Silva, Edgard P. Cardoso, and Ricardo H. C. Takahashi. Subpermutation-based evolutionary multiobjective algorithm for load restoration in power distribution networks. *IEEE Transactions on Evolutionary Computation*, 20(4):546–562, 2016.

[3] Eduardo G. Carrano, Ricardo H. C. Takahashi, Carlos M. Fonseca, and Oriane M. Neto. Nonlinear network optimization – an embedding vector space approach. *IEEE Transactions on Evolutionary Computation*, 14(2):206–226, 2010.

[4] Marco Caserta, Silvia Schwarze, and Stefan Vob. A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219:96–104, 2012.

[5] Marco Caserta, Stefan Vob, and Moshe Sniedovich. Applying the corridor method to a blocks relocation problem. *OR Spectrum*, 33:915–929, 2011.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[7] Carlos F. Daganzo. The crane scheduling problem. *Transportation Research Part B: Methodological*, 23(3):159–175, 1989.

[8] Yongpei Guan, Kang-Hung Yang, and Zhili Zhou. The crane scheduling problem: models and solution approaches. *Annals of Operations Research*, 203:119–139, 2013.

[9] Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. Variable Neighbourhood Search: methods and applications. *Annals of Operations Research*, 175(1):367–407, oct 2009.

[10] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[11] Damla Kizilay, Pascal Van Hentenryck, and Deniz T. Eliiyi. Constraint programming models for integrated container terminal operations. *European Journal of Operational Research*, 286:945–962, 2020.

[12] F. G. Konig, M. E. Lubbecke, R. H. Mohring, G. Schafer, and I. Spenke. Solutions to real-world instances of pspace-complete stacking. In *Proc. 15th Annual European Symposium on Algorithms – ESA*, volume 4698 of *Lecture Notes in Computer Science*. Springer, 2007.

[13] Pasquale Legato, Roberto Trunfio, and Frank Meisel. Modeling and solving rich quay crane scheduling problems. *Computers and Operations Research*, 39:2063–2078, 2012.

[14] Jana Lehnfeld and Sigrid Knust. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239:297–312, 2014.

[15] F. V. C. Martins, E. G. Carrano, E. F. Wanner, R. H. C. Takahashi, G. R. Mateus, and F. G. Nakamura. On a vector space representation in genetic algorithms for sensor scheduling in wireless sensor networks. *Evolutionary Computation*, 22(3):361–403, 2014.

[16] Gabriela N Maschietto, Yassine Ouazene, Martin G Ravetti, Maurício C de Souza, and Farouk Yalaoui. Crane scheduling problem with non-interference constraints in a steel coil distribution centre. *International Journal of Production Research*, 55(6):1607–1622, 2017.

[17] W. C. Ng. Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164:64–78, 2005.

[18] Matthew E.H. Petering and Mazen I. Hussein. A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *European Journal of Operational Research*, 231:120–130, 2013.

[19] Shunji Tanaka and Kenta Takii. A faster branch-and-bound algorithm for the block relocation problem. *IEEE Transactions on Automation Science and Engineering*, 13(1):181–190, 2016.

[20] Fabien Tricoire, Judith Scagnetti, and Andreas Beham. New insights on the block relocation problem. *Computers and Operations Research*, 89:127–139, 2018.

[21] Xie Xie, Yongyue Zheng, and Yanping Li. Genetic algorithm and its performance analysis for scheduling a single crane. *Discrete Dynamics in Nature and Society*, 2015:1–12, 2015.

[22] Yuan Yuan and Lixin Tang. Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse. *European Journal of Operational Research*, 262(2):424–437, 2017.

[23] Chenhao Zhou, Byung Kwon Lee, and Haobin Li. Integrated optimization on yard crane scheduling and vehicle positioning at container yards. *Transportation Research Part E*, 138, 2020.

[24] Bernard G. Zweers, Sandjai Bhulai, and Rob D. van der Mei. Optimizing pre-processing and relocation moves in the stochastic container relocation problem. *European Journal of Operational Research*, 283:954–971, 2020.

[25] Bernard G. Zweers, Sandjai Bhulai, and Rob D. van der Mei. Pre-processing a container yard under limited available time. *Computers and Operations Research*, 123, 2020.

[26] E. G. Carrano, L. D. Cruz, D. Baptista, D. Camargo, R. H. C. Takahashi. An efficient and fast local search based heuristic for reel management in a production line of oil extraction pipes. *Computers and Operations Research*, 137:105547, 2022.

[27] L. D. Cruz, E. G. Carrano, R. H. C. Takahashi. Modelos de programação linear para o planejamento do deslocamento de bobinas em uma fábrica de tubos flexíveis. *XV Simpósio Brasileiro de Automação Inteligente - SBAI*, 215-222, 2021.

[28] L. D. Cruz, E. G. Carrano, R. H. C. Takahashi. Repositório público com instâncias para o problemas de movimentação de bobinas. Disponível em https://github.com/ldiniz/rms_public. Acesso em 30 mar. 2024.

Apêndices

A Resultados Adicionais VNS–Heur

Devido ao comportamento estocástico de VNS–Heur, foram realizadas 10 repetições para o teste de cada instância. No texto principal foram apresentados os valores de mediana, melhor e pior caso. Neste apêndice são disponibilizados os resultados de todas as repetições para todas as instâncias.

Tabela 22 – Resultados encontrados pela VNS–Heur na Instância A.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	36	395	54
2	0	0	36	395	53
3	0	0	36	395	54
4	0	0	36	395	52
5	0	0	36	395	54
6	0	0	36	395	54
7	0	0	36	395	54
8	0	0	36	395	54
9	0	0	36	395	54
10	0	0	36	395	54

Tabela 23 – Resultados encontrados pela VNS–Heur na Instância B.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	6	535	71
2	0	0	6	520	69
3	0	0	6	520	70
4	0	0	6	520	69
5	0	0	9	520	66
6	0	0	6	520	70
7	0	0	9	520	68
8	0	0	9	520	67
9	0	0	6	520	71
10	0	0	6	520	69

Tabela 24 – Resultados encontrados pela VNS–Heur na Instância C.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	12	2457	65
2	0	0	12	2460	61
3	0	0	12	2457	65
4	0	0	12	2460	61
5	0	0	12	2460	61
6	0	0	12	1551	70
7	0	0	12	1551	64
8	0	0	12	2460	62
9	0	0	12	1551	64
10	0	0	12	1551	70

Tabela 25 – Resultados encontrados pela VNS–Heur na Instância D.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	74	126	108
2	0	0	76	438	96
3	0	0	76	438	98
4	0	0	78	438	100
5	0	0	76	117	106
6	0	0	74	126	106
7	0	0	77	117	100
8	0	0	74	126	111
9	0	0	70	123	105
10	0	0	75	120	99

Tabela 26 – Resultados encontrados pela VNS–Heur na Instância E.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	21	2093	94
2	0	0	21	2096	94
3	0	0	21	2093	94
4	0	0	21	2093	94
5	0	0	21	2093	94
6	0	0	21	2093	94
7	0	0	21	2191	105
8	0	0	21	2093	94
9	0	0	21	2060	85
10	0	0	21	2060	82

Tabela 27 – Resultados encontrados pela VNS–Heur na Instância F.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	18	3681	88
2	0	0	18	3705	93
3	0	0	18	3678	91
4	0	0	18	3681	95
5	0	0	18	3705	92
6	0	0	18	3681	97
7	0	0	18	3705	91
8	0	0	18	3678	91
9	0	0	18	3681	89
10	0	0	18	3681	89

Tabela 28 – Resultados encontrados pela VNS–Heur na Instância G.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	26	883	80
2	0	0	26	1518	84
3	0	0	23	1548	73
4	0	0	32	460	84
5	0	0	23	1548	84
6	0	0	23	849	77
7	0	0	23	1548	81
8	0	0	23	1722	77
9	0	0	23	1548	78
10	0	0	23	1548	85

Tabela 29 – Resultados encontrados pela VNS–Heur na Instância H.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	81	1954	72
2	0	0	88	1954	71
3	0	0	81	1954	76
4	0	0	81	1954	73
5	0	0	81	1954	70
6	0	0	81	1954	78
7	0	0	81	1954	73
8	0	0	81	1954	73
9	0	0	81	1954	73
10	0	0	81	1954	74

Tabela 30 – Resultados encontrados pela VNS–Heur na Instância I.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	37	1786	49
2	0	0	37	1786	49
3	0	0	37	1786	49
4	0	0	37	1786	49
5	0	0	37	1786	49
6	0	0	37	1786	49
7	0	0	37	1786	49
8	0	0	37	1786	49
9	0	0	37	1786	49
10	0	0	37	1786	49

Tabela 31 – Resultados encontrados pela VNS–Heur na Instância J.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	143	64	65
2	0	0	143	64	65
3	0	0	143	64	65
4	0	0	143	64	65
5	0	0	143	64	61
6	0	0	143	64	65
7	0	0	143	64	65
8	0	0	143	64	65
9	0	0	143	64	65
10	0	0	143	64	65

Tabela 32 – Resultados encontrados pela VNS–Heur na Instância K.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	130	60	79
2	0	0	130	60	76
3	0	0	132	60	80
4	0	0	133	60	78
5	0	0	130	60	75
6	0	0	130	60	79
7	0	0	132	60	77
8	0	0	132	60	79
9	0	0	130	60	78
10	0	0	134	60	88

Tabela 33 – Resultados encontrados pela VNS–Heur na Instância L.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	52	0	44
2	0	0	52	0	44
3	0	0	52	0	44
4	0	0	52	0	44
5	0	0	52	0	44
6	0	0	52	0	44
7	0	0	52	0	47
8	0	0	58	0	47
9	0	0	58	0	47
10	0	0	52	0	44

Tabela 34 – Resultados encontrados pela VNS–Heur na Instância M.

Repetição	F_1	F_2	F_3	F_4	F_5
1	0	0	8	15	36
2	0	0	11	15	41
3	0	0	8	15	37
4	0	0	11	15	40
5	0	0	8	15	37
6	0	0	11	15	40
7	0	0	8	15	36
8	0	0	11	15	41
9	0	0	11	15	40
10	0	0	8	15	37

B Res. Adicionais Modelo Temporal

No texto principal é apresentado o tempo completo de execução do modelo temporal na heurística ILP–Heur para cada instância e versão. Este apêndice detalha o tempo de execução por subtarefa, além de indicar a quantidade de instantes de movimentação utilizados.

Tabela 35 – Tempo de execução - Modelo Temporal - Instância A.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2686 segundos - FALHA	5172 segundos
Subtarefa A3-1 - 1 segundos - 15 instantes	Subtarefa A3-1 - 2 segundos - 4 instantes
Subtarefa A5-2 - 1 segundos - 15 instantes	Subtarefa A5-2 - 1 segundos - 3 instantes
Subtarefa A6-1 - 14 segundos - 15 instantes	Subtarefa A6-1 - 18 segundos - 12 instantes
Subtarefa A6-2 - 86 segundos - 15 instantes	Subtarefa A6-2 - 2457 segundos - 19 instantes
Subtarefa A7-1 - 2 segundos - 15 instantes	Subtarefa A7-1 - 2 segundos - 6 instantes
Subtarefa A7-2 - 42 segundos - 15 instantes	Subtarefa A7-2 - 14 segundos - 4 instantes
Subtarefa A8-2 - 2 segundos - 15 instantes	Subtarefa A8-2 - 2 segundos - 7 instantes
Subtarefa A9-1 - 30 segundos - 15 instantes	Subtarefa A9-1 - 15 segundos - 5 instantes
Subtarefa A9-2 - 2 segundos - 15 instantes	Subtarefa A9-2 - 2 segundos - 8 instantes
Subtarefa A10-1 - 2 segundos - 15 instantes	Subtarefa A10-1 - 2 segundos - 5 instantes
Subtarefa A10-2 - 23 segundos - 15 instantes	Subtarefa A10-2 - 7 segundos - 3 instantes
Subtarefa A11-1 - 1 segundos - 15 instantes	Subtarefa A11-1 - 1 segundos - 5 instantes
Subtarefa A12-1 - 2 segundos - 15 instantes	Subtarefa A12-1 - 2 segundos - 7 instantes
Subtarefa A12-2 - 32 segundos - 15 instantes	Subtarefa A12-2 - 10 segundos - 3 instantes
Subtarefa A13-1 - 31 segundos - 15 instantes	Subtarefa A13-1 - 16 segundos - 5 instantes
Subtarefa A13-2 - Falha - 2408 segundos - 26 instantes	Subtarefa A13-2 - 7 segundos - 14 instantes
-	Subtarefa A14-1 - 2410 segundos - 19 instantes
-	Subtarefa A14-2 - 34 segundos - 8 instantes
-	Subtarefa A15-1 - 91 segundos - 10 instantes
-	Subtarefa A15-2 - 91 segundos - 12 instantes

Tabela 36 – Tempo de execução - Modelo Temporal - Instância B.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2662 segundos - FALHA	5714 segundos
Subtarefa B4-1 - 56 segundos - 15 instantes	Subtarefa B4-1 - 29 segundos - 7 instantes
Subtarefa B5-1 - 2 segundos - 15 instantes	Subtarefa B5-1 - 2 segundos - 9 instantes
Subtarefa B6-1 - 42 segundos - 15 instantes	Subtarefa B6-1 - 25 segundos - 8 instantes
Subtarefa B6-2 - 2 segundos - 15 instantes	Subtarefa B6-2 - 1 segundos - 5 instantes
Subtarefa B7-1 - 2 segundos - 15 instantes	Subtarefa B7-1 - 2 segundos - 9 instantes
Subtarefa B7-2 - 36 segundos - 15 instantes	Subtarefa B7-2 - 17 segundos - 6 instantes
Subtarefa B8-1 - 35 segundos - 15 instantes	Subtarefa B8-1 - 30 segundos - 9 instantes
Subtarefa B8-2 - 1 segundos - 15 instantes	Subtarefa B8-2 - 2 segundos - 6 instantes
Subtarefa B9-1 - 1 segundos - 15 instantes	Subtarefa B9-1 - 3 segundos - 13 instantes
Subtarefa B10-1 - 11 segundos - 15 instantes	Subtarefa B10-1 - 83 segundos - 26 instantes
Subtarefa B10-2 - 25 segundos - 15 instantes	Subtarefa B10-2 - 6 segundos - 3 instantes
Subtarefa B11-1 - 33 segundos - 15 instantes	Subtarefa B11-1 - 36 segundos - 9 instantes
Subtarefa B11-2 - 1 segundos - 15 instantes	Subtarefa B11-2 - 1 segundos - 5 instantes
Subtarefa B12-1 - Falha - 2407 segundos - 26 instantes	Subtarefa B12-1 - 2406 segundos - 22 instantes
-	Subtarefa B12-2 - 8 segundos - 3 instantes
-	Subtarefa B15-1 - 2468 segundos - 47 instantes
-	Subtarefa B15-2 - 7 segundos - 13 instantes
-	Subtarefa B16-1 - 595 segundos - 14 instantes
-	Subtarefa B16-2 - 2 segundos - 4 instantes

Tabela 37 – Tempo de execução - Modelo Temporal - Instância C.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
4949 segundos - FALHA	8159 segundos
Subtarefa C7-1 - 39 segundos - 15 instantes	Subtarefa C7-1 - 13 segundos - 3 instantes
Subtarefa C8-1 - 9 segundos - 15 instantes	Subtarefa C8-1 - 45 segundos - 13 instantes
Subtarefa C9-1 - 2 segundos - 15 instantes	Subtarefa C9-1 - 30 segundos - 13 instantes
Subtarefa C9-2 - 40 segundos - 15 instantes	Subtarefa C9-2 - 23 segundos - 7 instantes
Subtarefa C10-1 - 2439 segundos - 26 instantes	Subtarefa C10-1 - 8 segundos - 11 instantes
Subtarefa C11-1 - Falha - 2416 segundos - 26 instantes	Subtarefa C11-1 - 2415 segundos - 23 instantes
-	Subtarefa C11-2 - 8 segundos - 2 instantes
-	Subtarefa C12-1 - 2712 segundos - 58 instantes
-	Subtarefa C12-2 - 2677 segundos - 52 instantes
-	Subtarefa C14-1 - 41 segundos - 9 instantes
-	Subtarefa C14-2 - 165 segundos - 19 instantes
-	Subtarefa C15-1 - 27 segundos - 8 instantes
-	Subtarefa C15-2 - 1 segundos - 5 instantes

Tabela 38 – Tempo de execução - Modelo Temporal - Instância D.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2427 segundos - FALHA	5327 segundos - FALHA
Subtarefa D5-2 - Falha - 2427 segundos - 26 instantes	Subtarefa D5-2 - 2432 segundos - 36 instantes
-	Subtarefa D6-1 - 1 segundos - 5 instantes
-	Subtarefa D6-2 - 2467 segundos - 27 instantes
-	Subtarefa D7-2 - 1 segundos - 4 instantes
-	Subtarefa D8-1 - 368 segundos - 19 instantes
-	Subtarefa D9-1 - 47 segundos - 21 instantes
-	Subtarefa D9-2 - Falha - 13 segundos - 92 instantes

Tabela 39 – Tempo de execução - Modelo Temporal - Instância E.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2414 segundos - FALHA	12525 segundos
Subtarefa E4-1 - 2 segundos - 15 instantes	Subtarefa E4-1 - 2 segundos - 5 instantes
Subtarefa E5-1 - 2 segundos - 15 instantes	Subtarefa E5-1 - 2 segundos - 6 instantes
Subtarefa E6-1 - 3 segundos - 15 instantes	Subtarefa E6-1 - 6 segundos - 14 instantes
Subtarefa E6-2 - 1 segundos - 15 instantes	Subtarefa E6-2 - 1 segundos - 4 instantes
Subtarefa E7-1 - Falha - 2404 segundos - 26 instantes	Subtarefa E7-1 - 2406 segundos - 29 instantes
-	Subtarefa E7-2 - 20 segundos - 3 instantes
-	Subtarefa E8-1 - 19 segundos - 4 instantes
-	Subtarefa E8-2 - 1 segundos - 4 instantes
-	Subtarefa E9-1 - 2405 segundos - 24 instantes
-	Subtarefa E10-1 - 3 segundos - 9 instantes
-	Subtarefa E10-2 - 202 segundos - 11 instantes
-	Subtarefa E11-1 - 2457 segundos - 40 instantes
-	Subtarefa E11-2 - 4 segundos - 18 instantes
-	Subtarefa E12-1 - 22 segundos - 3 instantes
-	Subtarefa E12-2 - 2 segundos - 4 instantes
-	Subtarefa E13-1 - 3 segundos - 7 instantes
-	Subtarefa E13-2 - 2410 segundos - 18 instantes
-	Subtarefa E14-1 - 6 segundos - 11 instantes
-	Subtarefa E15-1 - 154 segundos - 13 instantes
-	Subtarefa E15-2 - 2409 segundos - 38 instantes

Tabela 40 – Tempo de execução - Modelo Temporal - Instância F.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2556 segundos - FALHA	14400 segundos - FALHA
Subtarefa F3-1 - 3 segundos - 15 instantes	Subtarefa F3-1 - 1 segundos - 4 instantes
Subtarefa F3-2 - 45 segundos - 15 instantes	Subtarefa F3-2 - 32 segundos - 9 instantes
Subtarefa F6-1 - 2 segundos - 15 instantes	Subtarefa F6-1 - 4 segundos - 10 instantes
Subtarefa F7-1 - 3 segundos - 15 instantes	Subtarefa F7-1 - 9 segundos - 11 instantes
Subtarefa F7-2 - 37 segundos - 15 instantes	Subtarefa F7-2 - 11 segundos - 3 instantes
Subtarefa F8-1 - 49 segundos - 15 instantes	Subtarefa F8-1 - 76 segundos - 12 instantes
Subtarefa F8-2 - Falha - 2414 segundos - 26 instantes	Subtarefa F8-2 - 272 segundos - 19 instantes
-	Subtarefa F9-1 - 2 segundos - 4 instantes
-	Subtarefa F9-2 - 65 segundos - 10 instantes
-	Subtarefa F10-1 - 246 segundos - 14 instantes
-	Subtarefa F10-2 - 2 segundos - 5 instantes
-	Subtarefa F11-1 - 2406 segundos - 29 instantes
-	Subtarefa F12-1 - 2508 segundos - 33 instantes
-	Subtarefa F12-2 - 2435 segundos - 52 instantes
-	Subtarefa F13-1 - 2403 segundos - 27 instantes
-	Subtarefa F13-2 - 27 segundos - 10 instantes
-	Subtarefa F14-1 - 2405 segundos - 17 instantes
-	Subtarefa F15-1 - 1330 segundos - 14 instantes
-	Subtarefa F15-2 - 166 segundos - 41 instantes - Limite de tempo.

Tabela 41 – Tempo de execução - Modelo Temporal - Instância G.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
3230 segundos - FALHA	6893 segundos
Subtarefa G4-1 - 5 segundos - 15 instantes	Subtarefa G4-1 - 30 segundos - 15 instantes
Subtarefa G4-2 - 34 segundos - 15 instantes	Subtarefa G4-2 - 22 segundos - 9 instantes
Subtarefa G5-1 - 3 segundos - 15 instantes	Subtarefa G5-1 - 1 segundos - 3 instantes
Subtarefa G6-1 - 37 segundos - 15 instantes	Subtarefa G6-1 - 17 segundos - 6 instantes
Subtarefa G6-2 - 19 segundos - 15 instantes	Subtarefa G6-2 - 34 segundos - 20 instantes
Subtarefa G7-2 - 31 segundos - 15 instantes	Subtarefa G7-2 - 4 segundos - 13 instantes
Subtarefa G8-1 - 586 segundos - 26 instantes	Subtarefa G8-1 - 2407 segundos - 19 instantes
Subtarefa G8-2 - 53 segundos - 15 instantes	Subtarefa G8-2 - 26 segundos - 8 instantes
Subtarefa G9-1 - 48 segundos - 15 instantes	Subtarefa G9-1 - 26 segundos - 8 instantes
Subtarefa G9-2 - Falha - 2410 segundos - 26 instantes	Subtarefa G9-2 - 2406 segundos - 18 instantes
-	Subtarefa G10-1 - 222 segundos - 14 instantes
-	Subtarefa G10-2 - 163 segundos - 14 instantes
-	Subtarefa G11-1 - 3 segundos - 8 instantes
-	Subtarefa G12-1 - 5 segundos - 12 instantes
-	Subtarefa G12-2 - 2 segundos - 12 instantes
-	Subtarefa G13-2 - 1493 segundos - 25 instantes
-	Subtarefa G14-1 - 12 segundos - 5 instantes
-	Subtarefa G14-2 - 1 segundos - 5 instantes
-	Subtarefa G15-1 - 27 segundos - 8 instantes
-	Subtarefa G15-2 - 3 segundos - 7 instantes

Tabela 42 – Tempo de execução - Modelo Temporal - Instância H.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
1379 segundos - FALHA	7569 segundos
Subtarefa H9-2 - 39 segundos - 15 instantes	Subtarefa H9-2 - 12 segundos - 4 instantes
Subtarefa H10-1 - 4 segundos - 15 instantes	Subtarefa H10-1 - 3 segundos - 8 instantes
Subtarefa H10-2 - 62 segundos - 15 instantes	Subtarefa H10-2 - 186 segundos - 13 instantes
Subtarefa H11-1 - 2 segundos - 15 instantes	Subtarefa H11-1 - 6 segundos - 15 instantes
Subtarefa H11-2 - 4 segundos - 15 instantes	Subtarefa H11-2 - 4 segundos - 14 instantes
Subtarefa H12-1 - 30 segundos - 15 instantes	Subtarefa H12-1 - 16 segundos - 7 instantes
Subtarefa H13-1 - 5 segundos - 15 instantes	Subtarefa H13-1 - 6 segundos - 10 instantes
Subtarefa H13-2 - 35 segundos - 15 instantes	Subtarefa H13-2 - 14 segundos - 5 instantes
Subtarefa H14-1 - 3 segundos - 15 instantes	Subtarefa H14-1 - 1 segundos - 3 instantes
Subtarefa H15-1 - 3 segundos - 15 instantes	Subtarefa H15-1 - 3 segundos - 7 instantes
Subtarefa H15-2 - 39 segundos - 15 instantes	Subtarefa H15-2 - 48 segundos - 10 instantes
Subtarefa H16-1 - 4 segundos - 15 instantes	Subtarefa H16-1 - 171 segundos - 15 instantes
Subtarefa H17-1 - 38 segundos - 15 instantes	Subtarefa H17-1 - 16 segundos - 5 instantes
Subtarefa H17-2 - 1 segundos - 15 instantes	Subtarefa H17-2 - 1 segundos - 8 instantes
Subtarefa H18-1 - 28 segundos - 15 instantes	Subtarefa H18-1 - 15 segundos - 7 instantes
Subtarefa H19-1 - 3 segundos - 15 instantes	Subtarefa H19-1 - 4 segundos - 9 instantes
Subtarefa H19-2 - 38 segundos - 15 instantes	Subtarefa H19-2 - 1979 segundos - 17 instantes
Subtarefa H20-1 - 30 segundos - 15 instantes	Subtarefa H20-1 - 16 segundos - 8 instantes
Subtarefa H20-2 - Falha - 1002 segundos - 26 instantes	Subtarefa H20-2 - 2459 segundos - 36 instantes
-	Subtarefa H21-1 - 178 segundos - 16 instantes
-	Subtarefa H21-2 - 2423 segundos - 30 instantes
-	Subtarefa H22-1 - 17 segundos - 7 instantes
-	Subtarefa H22-2 - 1 segundos - 7 instantes

Tabela 43 – Tempo de execução - Modelo Temporal - Instância I.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
634 segundos	746 segundos
Subtarefa I6-2 - 1 segundos - 15 instantes	Subtarefa I6-2 - 8 segundos - 16 instantes
Subtarefa I7-1 - 35 segundos - 15 instantes	Subtarefa I7-1 - 34 segundos - 13 instantes
Subtarefa I8-1 - 5 segundos - 15 instantes	Subtarefa I8-1 - 1 segundos - 3 instantes
Subtarefa I8-2 - 29 segundos - 15 instantes	Subtarefa I8-2 - 11 segundos - 5 instantes
Subtarefa I10-1 - 4 segundos - 15 instantes	Subtarefa I10-1 - 2 segundos - 5 instantes
Subtarefa I10-2 - 32 segundos - 15 instantes	Subtarefa I10-2 - 16 segundos - 8 instantes
Subtarefa I11-1 - 395 segundos - 26 instantes	Subtarefa I11-1 - 586 segundos - 16 instantes
Subtarefa I12-1 - 1 segundos - 15 instantes	Subtarefa I12-1 - 1 segundos - 6 instantes
Subtarefa I12-2 - 1 segundos - 15 instantes	Subtarefa I12-2 - 3 segundos - 13 instantes
Subtarefa I14-1 - 8 segundos - 15 instantes	Subtarefa I14-1 - 22 segundos - 12 instantes
Subtarefa I14-2 - 29 segundos - 15 instantes	Subtarefa I14-2 - 18 segundos - 8 instantes
Subtarefa I15-1 - 24 segundos - 15 instantes	Subtarefa I15-1 - 8 segundos - 4 instantes
Subtarefa I15-2 - 2 segundos - 15 instantes	Subtarefa I15-2 - 2 segundos - 7 instantes
Subtarefa I16-1 - 2 segundos - 15 instantes	Subtarefa I16-1 - 4 segundos - 11 instantes
Subtarefa I16-2 - 19 segundos - 26 instantes	Subtarefa I16-2 - 3 segundos - 11 instantes
Subtarefa I17-1 - 35 segundos - 15 instantes	Subtarefa I17-1 - 25 segundos - 10 instantes
Subtarefa I17-2 - 1 segundos - 15 instantes	Subtarefa I17-2 - 11 segundos - 13 instantes

Tabela 44 – Tempo de execução - Modelo Temporal - Instância J.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
6330 segundos - FALHA	3343 segundos
Subtarefa J10-1 - 32 segundos - 15 instantes	Subtarefa J10-1 - 13 segundos - 5 instantes
Subtarefa J10-2 - 19 segundos - 15 instantes	Subtarefa J10-2 - 21 segundos - 14 instantes
Subtarefa J11-1 - 2 segundos - 15 instantes	Subtarefa J11-1 - 1 segundos - 4 instantes
Subtarefa J11-2 - 8 segundos - 15 instantes	Subtarefa J11-2 - 15 segundos - 20 instantes
Subtarefa J12-1 - 2 segundos - 15 instantes	Subtarefa J12-1 - 1 segundos - 5 instantes
Subtarefa J12-2 - 25 segundos - 15 instantes	Subtarefa J12-2 - 15 segundos - 8 instantes
Subtarefa J14-1 - 1 segundos - 15 instantes	Subtarefa J14-1 - 2 segundos - 10 instantes
Subtarefa J15-1 - 21 segundos - 15 instantes	Subtarefa J15-1 - 10 segundos - 6 instantes
Subtarefa J16-1 - 746 segundos - 26 instantes	Subtarefa J16-1 - 64 segundos - 18 instantes
Subtarefa J17-1 - 24 segundos - 15 instantes	Subtarefa J17-1 - 186 segundos - 13 instantes
Subtarefa J17-2 - 2415 segundos - 26 instantes	Subtarefa J17-2 - 9 segundos - 15 instantes
Subtarefa J18-1 - 4 segundos - 15 instantes	Subtarefa J18-1 - 4 segundos - 11 instantes
Subtarefa J18-2 - 21 segundos - 15 instantes	Subtarefa J18-2 - 2420 segundos - 22 instantes
Subtarefa J19-1 - 11 segundos - 15 instantes	Subtarefa J19-1 - 2 segundos - 7 instantes
Subtarefa J19-2 - 46 segundos - 15 instantes	Subtarefa J19-2 - 16 segundos - 8 instantes
Subtarefa J20-1 - 29 segundos - 15 instantes	Subtarefa J20-1 - 24 segundos - 10 instantes
Subtarefa J20-2 - 9 segundos - 15 instantes	Subtarefa J20-2 - 6 segundos - 11 instantes
Subtarefa J21-1 - 11 segundos - 15 instantes	Subtarefa J21-1 - 479 segundos - 17 instantes
Subtarefa J21-2 - 103 segundos - 26 instantes	Subtarefa J21-2 - 50 segundos - 19 instantes
Subtarefa J22-1 - 358 segundos - 26 instantes	Subtarefa J22-1 - 2 segundos - 8 instantes
Subtarefa J23-1 - Falha - 2431 segundos - 26 instantes	Subtarefa J23-1 - 1 segundos - 6 instantes
-	Subtarefa J24-2 - 10 segundos - 11 instantes

Tabela 45 – Tempo de execução - Modelo Temporal - Instância K.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
5192 segundos - FALHA	11402 segundos
Subtarefa K12-1 - 3 segundos - 15 instantes	Subtarefa K12-1 - 4 segundos - 9 instantes
Subtarefa K13-1 - 12 segundos - 15 instantes	Subtarefa K13-1 - 43 segundos - 17 instantes
Subtarefa K14-1 - 5 segundos - 15 instantes	Subtarefa K14-1 - 8 segundos - 4 instantes
Subtarefa K14-2 - 1 segundos - 15 instantes	Subtarefa K14-2 - 1 segundos - 6 instantes
Subtarefa K15-1 - 23 segundos - 15 instantes	Subtarefa K15-1 - 8 segundos - 4 instantes
Subtarefa K16-1 - 23 segundos - 15 instantes	Subtarefa K16-1 - 10 segundos - 6 instantes
Subtarefa K16-2 - 3 segundos - 15 instantes	Subtarefa K16-2 - 1 segundos - 3 instantes
Subtarefa K17-2 - 66 segundos - 15 instantes	Subtarefa K17-2 - 61 segundos - 12 instantes
Subtarefa K18-1 - 3 segundos - 15 instantes	Subtarefa K18-1 - 2 segundos - 8 instantes
Subtarefa K18-2 - 20 segundos - 15 instantes	Subtarefa K18-2 - 6 segundos - 4 instantes
Subtarefa K19-1 - 7 segundos - 15 instantes	Subtarefa K19-1 - 22 segundos - 14 instantes
Subtarefa K19-2 - 26 segundos - 15 instantes	Subtarefa K19-2 - 119 segundos - 12 instantes
Subtarefa K20-1 - 30 segundos - 15 instantes	Subtarefa K20-1 - 26 segundos - 18 instantes
Subtarefa K20-2 - 29 segundos - 15 instantes	Subtarefa K20-2 - 25 segundos - 10 instantes
Subtarefa K21-1 - 3 segundos - 15 instantes	Subtarefa K21-1 - 55 segundos - 11 instantes
Subtarefa K21-2 - 2457 segundos - 26 instantes	Subtarefa K21-2 - 2441 segundos - 31 instantes
Subtarefa K22-1 - 2 segundos - 15 instantes	Subtarefa K22-1 - 21 segundos - 16 instantes
Subtarefa K23-2 - 1 segundos - 15 instantes	Subtarefa K23-2 - 6 segundos - 12 instantes
Subtarefa K24-1 - Falha - 2468 segundos - 26 instantes	Subtarefa K24-1 - 2438 segundos - 27 instantes
-	Subtarefa K24-2 - 2270 segundos - 19 instantes
-	Subtarefa K25-1 - 145 segundos - 15 instantes
-	Subtarefa K25-2 - 5 segundos - 3 instantes
-	Subtarefa K26-1 - 1257 segundos - 22 instantes
-	Subtarefa K26-2 - 2442 segundos - 28 instantes

Tabela 46 – Tempo de execução - Modelo Temporal - Instância L.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
338 segundos	575 segundos
Subtarefa L3-2 - 12 segundos - 15 instantes	Subtarefa L3-2 - 16 segundos - 16 instantes
Subtarefa L6-2 - 3 segundos - 15 instantes	Subtarefa L6-2 - 1 segundos - 3 instantes
Subtarefa L12-1 - 11 segundos - 15 instantes	Subtarefa L12-1 - 4 segundos - 13 instantes
Subtarefa L13-2 - 5 segundos - 15 instantes	Subtarefa L13-2 - 36 segundos - 19 instantes
Subtarefa L14-1 - 2 segundos - 15 instantes	Subtarefa L14-1 - 2 segundos - 7 instantes
Subtarefa L14-2 - 23 segundos - 15 instantes	Subtarefa L14-2 - 11 segundos - 6 instantes
Subtarefa L15-1 - 19 segundos - 15 instantes	Subtarefa L15-1 - 6 segundos - 4 instantes
Subtarefa L17-1 - 2 segundos - 15 instantes	Subtarefa L17-1 - 1 segundos - 5 instantes
Subtarefa L17-2 - 111 segundos - 15 instantes	Subtarefa L17-2 - 255 segundos - 15 instantes
Subtarefa L18-1 - 13 segundos - 15 instantes	Subtarefa L18-1 - 21 segundos - 18 instantes
Subtarefa L18-2 - 29 segundos - 15 instantes	Subtarefa L18-2 - 77 segundos - 22 instantes
Subtarefa L19-1 - 32 segundos - 15 instantes	Subtarefa L19-1 - 15 segundos - 6 instantes
Subtarefa L19-2 - 4 segundos - 15 instantes	Subtarefa L19-2 - 22 segundos - 13 instantes
Subtarefa L20-1 - 8 segundos - 15 instantes	Subtarefa L20-1 - 64 segundos - 19 instantes
Subtarefa L21-1 - 50 segundos - 15 instantes	Subtarefa L21-1 - 33 segundos - 9 instantes
Subtarefa L21-2 - 6 segundos - 15 instantes	Subtarefa L21-2 - 17 segundos - 10 instantes

Tabela 47 – Tempo de execução - Modelo Temporal - Instância M.

Modelo Temporal - Lista Vazia	**Modelo Temporal - BuildMoveListCombined**
2701 segundos	9572 segundos
Subtarefa M13-1 - 2 segundos - 15 instantes	Subtarefa M13-1 - 1 segundos - 3 instantes
Subtarefa M13-2 - 27 segundos - 15 instantes	Subtarefa M13-2 - 12 segundos - 6 instantes
Subtarefa M14-1 - 29 segundos - 15 instantes	Subtarefa M14-1 - 16 segundos - 8 instantes
Subtarefa M15-1 - 2 segundos - 15 instantes	Subtarefa M15-1 - 3 segundos - 9 instantes
Subtarefa M15-2 - 899 segundos - 26 instantes	Subtarefa M15-2 - 2411 segundos - 22 instantes
Subtarefa M16-1 - 2 segundos - 15 instantes	Subtarefa M16-1 - 2 segundos - 5 instantes
Subtarefa M17-1 - 85 segundos - 15 instantes	Subtarefa M17-1 - 291 segundos - 13 instantes
Subtarefa M17-2 - 24 segundos - 15 instantes	Subtarefa M17-2 - 2467 segundos - 40 instantes
Subtarefa M18-1 - 731 segundos - 26 instantes	Subtarefa M18-1 - 2443 segundos - 33 instantes
Subtarefa M18-2 - 853 segundos - 26 instantes	Subtarefa M18-2 - 50 segundos - 14 instantes
Subtarefa M19-1 - 38 segundos - 15 instantes	Subtarefa M19-1 - 30 segundos - 10 instantes
Subtarefa M19-2 - 2 segundos - 15 instantes	Subtarefa M19-2 - 1853 segundos - 25 instantes

C Res. Adicionais Modelo em Estágios

No texto principal é apresentado o tempo completo de execução do modelo em estágios/submodelo atemporal na heurística ILP–Heur para cada instância e versão. Este apêndice detalha o tempo de execução por subtarefa, além de indicar a quantidade de instantes de movimentação utilizados.

Tabela 48 – Tempo de execução - Modelo em Estágios - Instância A.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2655 segundos	305 segundos
Subtarefa A3-1 - 1 segundos - 15 instantes	Subtarefa A3-1 - 1 segundos - 4 instantes
Subtarefa A5-2 - 0 segundos - 15 instantes	Subtarefa A5-2 - 1 segundos - 3 instantes
Subtarefa A6-1 - 10 segundos - 15 instantes	Subtarefa A6-1 - 9 segundos - 12 instantes
Subtarefa A6-2 - 35 segundos - 15 instantes	Subtarefa A6-2 - 56 segundos - 19 instantes
Subtarefa A7-1 - 1 segundos - 15 instantes	Subtarefa A7-1 - 1 segundos - 3 instantes
Subtarefa A7-2 - 25 segundos - 15 instantes	Subtarefa A7-2 - 9 segundos - 4 instantes
Subtarefa A8-2 - 1 segundos - 15 instantes	Subtarefa A8-2 - 1 segundos - 6 instantes
Subtarefa A9-1 - 23 segundos - 15 instantes	Subtarefa A9-1 - 10 segundos - 5 instantes
Subtarefa A9-2 - 1 segundos - 15 instantes	Subtarefa A9-2 - 1 segundos - 6 instantes
Subtarefa A10-1 - 1 segundos - 15 instantes	Subtarefa A10-1 - 1 segundos - 6 instantes
Subtarefa A10-2 - 19 segundos - 15 instantes	Subtarefa A10-2 - 5 segundos - 3 instantes
Subtarefa A11-1 - 1 segundos - 15 instantes	Subtarefa A11-1 - 1 segundos - 5 instantes
Subtarefa A12-1 - 1 segundos - 15 instantes	Subtarefa A12-1 - 1 segundos - 7 instantes
Subtarefa A12-2 - 25 segundos - 15 instantes	Subtarefa A12-2 - 7 segundos - 3 instantes
Subtarefa A13-1 - 25 segundos - 15 instantes	Subtarefa A13-1 - 10 segundos - 5 instantes
Subtarefa A13-2 - 2408 segundos - 26 instantes	Subtarefa A13-2 - 3 segundos - 14 instantes
Subtarefa A14-1 - 4 segundos - 15 instantes	Subtarefa A14-1 - 121 segundos - 19 instantes
Subtarefa A14-2 - 28 segundos - 15 instantes	Subtarefa A14-2 - 17 segundos - 8 instantes
Subtarefa A15-1 - 31 segundos - 15 instantes	Subtarefa A15-1 - 24 segundos - 10 instantes
Subtarefa A15-2 - 6 segundos - 15 instantes	Subtarefa A15-2 - 37 segundos - 15 instantes

Tabela 49 – Tempo de execução - Modelo em Estágios - Instância B.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2588 segundos - FALHA	4999 segundos
Subtarefa B4-1 - 40 segundos - 15 instantes	Subtarefa B4-1 - 22 segundos - 7 instantes
Subtarefa B5-1 - 2 segundos - 15 instantes	Subtarefa B5-1 - 2 segundos - 9 instantes
Subtarefa B6-1 - 31 segundos - 15 instantes	Subtarefa B6-1 - 19 segundos - 8 instantes
Subtarefa B6-2 - 2 segundos - 15 instantes	Subtarefa B6-2 - 1 segundos - 5 instantes
Subtarefa B7-1 - 1 segundos - 15 instantes	Subtarefa B7-1 - 2 segundos - 9 instantes
Subtarefa B7-2 - 27 segundos - 15 instantes	Subtarefa B7-2 - 14 segundos - 6 instantes
Subtarefa B8-1 - 24 segundos - 15 instantes	Subtarefa B8-1 - 17 segundos - 9 instantes
Subtarefa B8-2 - 0 segundos - 15 instantes	Subtarefa B8-2 - 1 segundos - 6 instantes
Subtarefa B9-1 - 1 segundos - 15 instantes	Subtarefa B9-1 - 2 segundos - 13 instantes
Subtarefa B10-1 - 3 segundos - 15 instantes	Subtarefa B10-1 - 7 segundos - 24 instantes
Subtarefa B10-2 - 20 segundos - 15 instantes	Subtarefa B10-2 - 5 segundos - 3 instantes
Subtarefa B11-1 - 23 segundos - 15 instantes	Subtarefa B11-1 - 16 segundos - 9 instantes
Subtarefa B11-2 - 1 segundos - 15 instantes	Subtarefa B11-2 - 1 segundos - 5 instantes
Subtarefa B12-1 - 2409 segundos - 26 instantes	Subtarefa B12-1 - 2405 segundos - 23 instantes
-	Subtarefa B12-2 - 7 segundos - 3 instantes
-	Subtarefa B15-1 - 2446 segundos - 47 instantes
-	Subtarefa B15-2 - 3 segundos - 13 instantes
-	Subtarefa B16-1 - 38 segundos - 14 instantes
-	Subtarefa B16-2 - 1 segundos - 4 instantes

Tabela 50 – Tempo de execução - Modelo em Estágios - Instância C.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
4886 segundos - FALHA	10242 segundos
Subtarefa C7-1 - 29 segundos - 15 instantes	Subtarefa C7-1 - 12 segundos - 3 instantes
Subtarefa C8-1 - 6 segundos - 15 instantes	Subtarefa C8-1 - 5 segundos - 13 instantes
Subtarefa C9-1 - 2 segundos - 15 instantes	Subtarefa C9-1 - 4 segundos - 13 instantes
Subtarefa C9-2 - 25 segundos - 15 instantes	Subtarefa C9-2 - 16 segundos - 7 instantes
Subtarefa C10-1 - 2408 segundos - 26 instantes	Subtarefa C10-1 - 2236 segundos - 22 instantes
Subtarefa C11-1 - 2412 segundos - 26 instantes	Subtarefa C11-1 - 429 segundos - 27 instantes
-	Subtarefa C11-2 - 3 segundos - 10 instantes
-	Subtarefa C12-1 - 2581 segundos - 65 instantes
-	Subtarefa C12-2 - 2481 segundos - 30 instantes
-	Subtarefa C14-1 - 2449 segundos - 23 instantes
-	Subtarefa C14-2 - 7 segundos - 10 instantes
-	Subtarefa C15-1 - 22 segundos - 8 instantes
-	Subtarefa C15-2 - 3 segundos - 11 instantes

Tabela 51 – Tempo de execução - Modelo em Estágios - Instância D.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2423 segundos - FALHA	5093 segundos - FALHA
Subtarefa D5-2 - 2422 segundos - 26 instantes	Subtarefa D5-2 - 2423 segundos - 36 instantes
-	Subtarefa D6-1 - 2 segundos - 10 instantes
-	Subtarefa D6-2 - 2446 segundos - 27 instantes
-	Subtarefa D7-2 - 2 segundos - 6 instantes
-	Subtarefa D8-1 - 205 segundos - 22 instantes
-	Subtarefa D9-1 - 7 segundos - 21 instantes
-	Subtarefa D9-2 - 13 segundos - 92 instantes

Tabela 52 – Tempo de execução - Modelo em Estágios - Instância E.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
5144 segundos - FALHA	7453 segundos
Subtarefa E4-1 - 2 segundos - 15 instantes	Subtarefa E4-1 - 1 segundos - 5 instantes
Subtarefa E5-1 - 1 segundos - 15 instantes	Subtarefa E5-1 - 1 segundos - 6 instantes
Subtarefa E6-1 - 3 segundos - 15 instantes	Subtarefa E6-1 - 4 segundos - 14 instantes
Subtarefa E6-2 - 1 segundos - 15 instantes	Subtarefa E6-2 - 1 segundos - 4 instantes
Subtarefa E7-1 - 2404 segundos - 26 instantes	Subtarefa E7-1 - 2406 segundos - 29 instantes
Subtarefa E7-2 - 41 segundos - 15 instantes	Subtarefa E7-2 - 12 segundos - 3 instantes
Subtarefa E8-1 - 38 segundos - 15 instantes	Subtarefa E8-1 - 13 segundos - 4 instantes
Subtarefa E8-2 - 2 segundos - 15 instantes	Subtarefa E8-2 - 1 segundos - 4 instantes
Subtarefa E9-1 - 1672 segundos - 26 instantes	Subtarefa E9-1 - 2 segundos - 10 instantes
Subtarefa E10-1 - 2 segundos - 15 instantes	Subtarefa E10-1 - 2 segundos - 10 instantes
Subtarefa E10-2 - 49 segundos - 15 instantes	Subtarefa E10-2 - 36 segundos - 11 instantes
Subtarefa E11-1 - 923 segundos - 26 instantes	Subtarefa E11-1 - 2432 segundos - 33 instantes
-	Subtarefa E11-2 - 2 segundos - 18 instantes
-	Subtarefa E12-1 - 15 segundos - 3 instantes
-	Subtarefa E12-2 - 1 segundos - 3 instantes
-	Subtarefa E13-1 - 2 segundos - 7 instantes
-	Subtarefa E13-2 - 76 segundos - 18 instantes
-	Subtarefa E14-1 - 2 segundos - 8 instantes
-	Subtarefa E15-1 - 39 segundos - 13 instantes
-	Subtarefa E15-2 - 2412 segundos - 26 instantes

Tabela 53 – Tempo de execução - Modelo em Estágios - Instância F.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2527 segundos - FALHA	9261 segundos
Subtarefa F3-1 - 2 segundos - 15 instantes	Subtarefa F3-1 - 1 segundos - 4 instantes
Subtarefa F3-2 - 34 segundos - 15 instantes	Subtarefa F3-2 - 22 segundos - 9 instantes
Subtarefa F6-1 - 2 segundos - 15 instantes	Subtarefa F6-1 - 2 segundos - 10 instantes
Subtarefa F7-1 - 2 segundos - 15 instantes	Subtarefa F7-1 - 3 segundos - 11 instantes
Subtarefa F7-2 - 29 segundos - 15 instantes	Subtarefa F7-2 - 9 segundos - 3 instantes
Subtarefa F8-1 - 32 segundos - 15 instantes	Subtarefa F8-1 - 27 segundos - 12 instantes
Subtarefa F8-2 - 2421 segundos - 26 instantes	Subtarefa F8-2 - 8 segundos - 16 instantes
-	Subtarefa F9-1 - 1 segundos - 4 instantes
-	Subtarefa F9-2 - 21 segundos - 10 instantes
-	Subtarefa F10-1 - 30 segundos - 14 instantes
-	Subtarefa F10-2 - 1 segundos - 5 instantes
-	Subtarefa F11-1 - 2405 segundos - 29 instantes
-	Subtarefa F12-1 - 2475 segundos - 33 instantes
-	Subtarefa F12-2 - 2421 segundos - 39 instantes
-	Subtarefa F13-1 - 1645 segundos - 27 instantes
-	Subtarefa F13-2 - 17 segundos - 10 instantes
-	Subtarefa F14-1 - 105 segundos - 17 instantes
-	Subtarefa F15-1 - 41 segundos - 14 instantes
-	Subtarefa F15-2 - 16 segundos - 15 instantes
-	Subtarefa F16-1 - 17 segundos - 5 instantes
-	Subtarefa F16-2 - 3 segundos - 9 instantes

Tabela 54 – Tempo de execução - Modelo em Estágios - Instância G.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2806 segundos - FALHA	4032 segundos
Subtarefa G4-1 - 3 segundos - 15 instantes	Subtarefa G4-1 - 6 segundos - 15 instantes
Subtarefa G4-2 - 27 segundos - 15 instantes	Subtarefa G4-2 - 17 segundos - 9 instantes
Subtarefa G5-1 - 2 segundos - 15 instantes	Subtarefa G5-1 - 1 segundos - 3 instantes
Subtarefa G6-1 - 30 segundos - 15 instantes	Subtarefa G6-1 - 14 segundos - 6 instantes
Subtarefa G6-2 - 12 segundos - 15 instantes	Subtarefa G6-2 - 9 segundos - 20 instantes
Subtarefa G7-2 - 3 segundos - 15 instantes	Subtarefa G7-2 - 6 segundos - 17 instantes
Subtarefa G8-1 - 246 segundos - 26 instantes	Subtarefa G8-1 - 1331 segundos - 19 instantes
Subtarefa G8-2 - 36 segundos - 15 instantes	Subtarefa G8-2 - 21 segundos - 8 instantes
Subtarefa G9-1 - 36 segundos - 15 instantes	Subtarefa G9-1 - 20 segundos - 8 instantes
Subtarefa G9-2 - 2405 segundos - 26 instantes	Subtarefa G9-2 - 19 segundos - 18 instantes
-	Subtarefa G10-1 - 9 segundos - 14 instantes
-	Subtarefa G10-2 - 34 segundos - 14 instantes
-	Subtarefa G11-1 - 3 segundos - 8 instantes
-	Subtarefa G12-1 - 3 segundos - 12 instantes
-	Subtarefa G12-2 - 1 segundos - 12 instantes
-	Subtarefa G13-2 - 105 segundos - 25 instantes
-	Subtarefa G14-1 - 10 segundos - 5 instantes
-	Subtarefa G14-2 - 2409 segundos - 24 instantes
-	Subtarefa G15-1 - 22 segundos - 8 instantes
-	Subtarefa G15-2 - 2 segundos - 7 instantes

Tabela 55 – Tempo de execução - Modelo em Estágios - Instância H.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
591 segundos - FALHA	3532 segundos
Subtarefa H9-2 - 31 segundos - 15 instantes	Subtarefa H9-2 - 11 segundos - 4 instantes
Subtarefa H10-1 - 3 segundos - 15 instantes	Subtarefa H10-1 - 2 segundos - 8 instantes
Subtarefa H10-2 - 37 segundos - 15 instantes	Subtarefa H10-2 - 27 segundos - 13 instantes
Subtarefa H11-1 - 2 segundos - 15 instantes	Subtarefa H11-1 - 3 segundos - 15 instantes
Subtarefa H11-2 - 1 segundos - 15 instantes	Subtarefa H11-2 - 3 segundos - 14 instantes
Subtarefa H12-1 - 25 segundos - 15 instantes	Subtarefa H12-1 - 13 segundos - 7 instantes
Subtarefa H13-1 - 4 segundos - 15 instantes	Subtarefa H13-1 - 4 segundos - 10 instantes
Subtarefa H13-2 - 29 segundos - 15 instantes	Subtarefa H13-2 - 12 segundos - 5 instantes
Subtarefa H14-1 - 2 segundos - 15 instantes	Subtarefa H14-1 - 1 segundos - 3 instantes
Subtarefa H15-1 - 4 segundos - 15 instantes	Subtarefa H15-1 - 2 segundos - 7 instantes
Subtarefa H15-2 - 30 segundos - 15 instantes	Subtarefa H15-2 - 20 segundos - 10 instantes
Subtarefa H16-1 - 43 segundos - 15 instantes	Subtarefa H16-1 - 33 segundos - 15 instantes
Subtarefa H17-1 - 33 segundos - 15 instantes	Subtarefa H17-1 - 14 segundos - 5 instantes
Subtarefa H17-2 - 1 segundos - 15 instantes	Subtarefa H17-2 - 1 segundos - 8 instantes
Subtarefa H18-1 - 23 segundos - 15 instantes	Subtarefa H18-1 - 12 segundos - 7 instantes
Subtarefa H19-1 - 2 segundos - 15 instantes	Subtarefa H19-1 - 2 segundos - 9 instantes
Subtarefa H19-2 - 33 segundos - 15 instantes	Subtarefa H19-2 - 34 segundos - 17 instantes
Subtarefa H20-1 - 21 segundos - 15 instantes	Subtarefa H20-1 - 13 segundos - 8 instantes
Subtarefa H20-2 - 258 segundos - 26 instantes	Subtarefa H20-2 - 2445 segundos - 36 instantes
-	Subtarefa H21-1 - 7 segundos - 16 instantes
-	Subtarefa H21-2 - 862 segundos - 30 instantes
-	Subtarefa H22-1 - 18 segundos - 8 instantes
-	Subtarefa H22-2 - 2 segundos - 13 instantes

Tabela 56 – Tempo de execução - Modelo em Estágios - Instância I.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
324 segundos	193 segundos
Subtarefa I6-2 - 1 segundos - 15 instantes	Subtarefa I6-2 - 3 segundos - 16 instantes
Subtarefa I7-1 - 28 segundos - 15 instantes	Subtarefa I7-1 - 25 segundos - 13 instantes
Subtarefa I8-1 - 4 segundos - 15 instantes	Subtarefa I8-1 - 1 segundos - 3 instantes
Subtarefa I8-2 - 24 segundos - 15 instantes	Subtarefa I8-2 - 10 segundos - 5 instantes
Subtarefa I10-1 - 3 segundos - 15 instantes	Subtarefa I10-1 - 1 segundos - 5 instantes
Subtarefa I10-2 - 21 segundos - 15 instantes	Subtarefa I10-2 - 13 segundos - 8 instantes
Subtarefa I11-1 - 152 segundos - 26 instantes	Subtarefa I11-1 - 89 segundos - 16 instantes
Subtarefa I12-1 - 0 segundos - 15 instantes	Subtarefa I12-1 - 1 segundos - 6 instantes
Subtarefa I12-2 - 0 segundos - 15 instantes	Subtarefa I12-2 - 2 segundos - 13 instantes
Subtarefa I14-1 - 5 segundos - 15 instantes	Subtarefa I14-1 - 4 segundos - 12 instantes
Subtarefa I14-2 - 22 segundos - 15 instantes	Subtarefa I14-2 - 15 segundos - 8 instantes
Subtarefa I15-1 - 19 segundos - 15 instantes	Subtarefa I15-1 - 7 segundos - 4 instantes
Subtarefa I15-2 - 2 segundos - 15 instantes	Subtarefa I15-2 - 1 segundos - 7 instantes
Subtarefa I16-1 - 2 segundos - 15 instantes	Subtarefa I16-1 - 3 segundos - 11 instantes
Subtarefa I16-2 - 4 segundos - 15 instantes	Subtarefa I16-2 - 5 segundos - 18 instantes
Subtarefa I17-1 - 27 segundos - 15 instantes	Subtarefa I17-1 - 19 segundos - 10 instantes
Subtarefa I17-2 - 1 segundos - 15 instantes	Subtarefa I17-2 - 1 segundos - 8 instantes

Tabela 57 – Tempo de execução - Modelo em Estágios - Instância J.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
4024 segundos - FALHA	235 segundos
Subtarefa J10-1 - 27 segundos - 15 instantes	Subtarefa J10-1 - 12 segundos - 5 instantes
Subtarefa J10-2 - 16 segundos - 15 instantes	Subtarefa J10-2 - 16 segundos - 14 instantes
Subtarefa J10-2 - 16 segundos - 15 instantes	Subtarefa J11-1 - 1 segundos - 4 instantes
Subtarefa J11-2 - 7 segundos - 15 instantes	Subtarefa J11-2 - 12 segundos - 17 instantes
Subtarefa J12-1 - 2 segundos - 15 instantes	Subtarefa J12-1 - 3 segundos - 9 instantes
Subtarefa J12-2 - 21 segundos - 15 instantes	Subtarefa J12-2 - 11 segundos - 7 instantes
Subtarefa J14-1 - 1 segundos - 15 instantes	Subtarefa J14-1 - 2 segundos - 10 instantes
Subtarefa J15-1 - 18 segundos - 15 instantes	Subtarefa J15-1 - 9 segundos - 6 instantes
Subtarefa J16-1 - 491 segundos - 26 instantes	Subtarefa J16-1 - 10 segundos - 18 instantes
Subtarefa J17-1 - 17 segundos - 15 instantes	Subtarefa J17-1 - 15 segundos - 10 instantes
Subtarefa J17-2 - 856 segundos - 26 instantes	Subtarefa J17-2 - 1 segundos - 11 instantes
Subtarefa J18-1 - 3 segundos - 15 instantes	Subtarefa J18-1 - 4 segundos - 12 instantes
Subtarefa J18-2 - 12 segundos - 15 instantes	Subtarefa J18-2 - 21 segundos - 21 instantes
Subtarefa J19-1 - 2 segundos - 15 instantes	Subtarefa J19-1 - 3 segundos - 11 instantes
Subtarefa J19-2 - 23 segundos - 15 instantes	Subtarefa J19-2 - 13 segundos - 8 instantes
Subtarefa J20-1 - 23 segundos - 15 instantes	Subtarefa J20-1 - 17 segundos - 10 instantes
Subtarefa J20-2 - 9 segundos - 15 instantes	Subtarefa J20-2 - 10 segundos - 17 instantes
Subtarefa J21-1 - 7 segundos - 15 instantes	Subtarefa J21-1 - 44 segundos - 17 instantes
Subtarefa J21-2 - 28 segundos - 26 instantes	Subtarefa J21-2 - 7 segundos - 19 instantes
Subtarefa J22-1 - 34 segundos - 26 instantes	Subtarefa J22-1 - 1 segundos - 8 instantes
Subtarefa J23-1 - 2415 segundos - 26 instantes	Subtarefa J23-1 - 31 segundos - 15 instantes
-	Subtarefa J24-2 - 1 segundos - 7 instantes

Tabela 58 – Tempo de execução - Modelo em Estágios - Instância K.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
2003 segundos - FALHA	5069 segundos
Subtarefa K12-1 - 2 segundos - 15 instantes	Subtarefa K12-1 - 2 segundos - 9 instantes
Subtarefa K13-1 - 11 segundos - 15 instantes	Subtarefa K13-1 - 7 segundos - 17 instantes
Subtarefa K14-1 - 4 segundos - 15 instantes	Subtarefa K14-1 - 7 segundos - 4 instantes
Subtarefa K14-2 - 1 segundos - 15 instantes	Subtarefa K14-2 - 1 segundos - 6 instantes
Subtarefa K15-1 - 17 segundos - 15 instantes	Subtarefa K15-1 - 6 segundos - 4 instantes
Subtarefa K16-1 - 16 segundos - 15 instantes	Subtarefa K16-1 - 8 segundos - 6 instantes
Subtarefa K16-2 - 2 segundos - 15 instantes	Subtarefa K16-2 - 1 segundos - 3 instantes
Subtarefa K17-2 - 22 segundos - 15 instantes	Subtarefa K17-2 - 13 segundos - 12 instantes
Subtarefa K18-1 - 1 segundos - 15 instantes	Subtarefa K18-1 - 1 segundos - 8 instantes
Subtarefa K18-2 - 14 segundos - 15 instantes	Subtarefa K18-2 - 5 segundos - 4 instantes
Subtarefa K19-1 - 2 segundos - 15 instantes	Subtarefa K19-1 - 6 segundos - 14 instantes
Subtarefa K19-2 - 20 segundos - 15 instantes	Subtarefa K19-2 - 15 segundos - 12 instantes
Subtarefa K20-1 - 5 segundos - 15 instantes	Subtarefa K20-1 - 9 segundos - 18 instantes
Subtarefa K20-2 - 21 segundos - 15 instantes	Subtarefa K20-2 - 15 segundos - 10 instantes
Subtarefa K21-1 - 3 segundos - 15 instantes	Subtarefa K21-1 - 3 segundos - 11 instantes
Subtarefa K21-2 - 1402 segundos - 26 instantes	Subtarefa K21-2 - 2444 segundos - 34 instantes
Subtarefa K22-1 - 1 segundos - 15 instantes	Subtarefa K22-1 - 3 segundos - 16 instantes
Subtarefa K23-2 - 4 segundos - 15 instantes	Subtarefa K23-2 - 5 segundos - 12 instantes
Subtarefa K24-1 - 20 segundos - 15 instantes	Subtarefa K24-1 - 2430 segundos - 27 instantes
Subtarefa K24-2 - 1 segundos - 15 instantes	Subtarefa K24-2 - 4 segundos - 15 instantes
Subtarefa K25-1 - 5 segundos - 15 instantes	Subtarefa K25-1 - 9 segundos - 15 instantes
Subtarefa K25-2 - 15 segundos - 15 instantes	Subtarefa K25-2 - 4 segundos - 3 instantes
Subtarefa K26-1 - 7 segundos - 15 instantes	Subtarefa K26-1 - 31 segundos - 22 instantes
Subtarefa K26-2 - 395 segundos - 26 instantes	Subtarefa K26-2 - 51 segundos - 28 instantes

Tabela 59 – Tempo de execução - Modelo em Estágios - Instância L.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
187 segundos	144 segundos
Subtarefa L3-2 - 9 segundos - 15 instantes	Subtarefa L3-2 - 12 segundos - 16 instantes
Subtarefa L6-2 - 2 segundos - 15 instantes	Subtarefa L6-2 - 1 segundos - 3 instantes
Subtarefa L12-1 - 7 segundos - 15 instantes	Subtarefa L12-1 - 3 segundos - 13 instantes
Subtarefa L13-2 - 4 segundos - 15 instantes	Subtarefa L13-2 - 7 segundos - 19 instantes
Subtarefa L14-1 - 2 segundos - 15 instantes	Subtarefa L14-1 - 1 segundos - 7 instantes
Subtarefa L14-2 - 16 segundos - 15 instantes	Subtarefa L14-2 - 8 segundos - 6 instantes
Subtarefa L15-1 - 16 segundos - 15 instantes	Subtarefa L15-1 - 6 segundos - 4 instantes
Subtarefa L17-1 - 1 segundos - 15 instantes	Subtarefa L17-1 - 1 segundos - 5 instantes
Subtarefa L17-2 - 21 segundos - 15 instantes	Subtarefa L17-2 - 23 segundos - 15 instantes
Subtarefa L18-1 - 11 segundos - 15 instantes	Subtarefa L18-1 - 14 segundos - 18 instantes
Subtarefa L18-2 - 15 segundos - 15 instantes	Subtarefa L18-2 - 24 segundos - 22 instantes
Subtarefa L19-1 - 26 segundos - 15 instantes	Subtarefa L19-1 - 12 segundos - 6 instantes
Subtarefa L19-2 - 3 segundos - 15 instantes	Subtarefa L19-2 - 4 segundos - 13 instantes
Subtarefa L20-1 - 2 segundos - 15 instantes	Subtarefa L20-1 - 6 segundos - 19 instantes
Subtarefa L21-1 - 38 segundos - 15 instantes	Subtarefa L21-1 - 26 segundos - 9 instantes
Subtarefa L21-2 - 5 segundos - 15 instantes	Subtarefa L21-2 - 4 segundos - 10 instantes

Tabela 60 – Tempo de execução - Modelo em Estágios - Instância M.

Submodelo Atemporal - Lista Vazia	**Submodelo Atemporal - BuildMoveListCombined**
5293 segundos	438 segundos
Subtarefa M13-1 - 1 segundos - 15 instantes	Subtarefa M13-1 - 1 segundos - 3 instantes
Subtarefa M13-2 - 20 segundos - 15 instantes	Subtarefa M13-2 - 10 segundos - 6 instantes
Subtarefa M14-1 - 19 segundos - 15 instantes	Subtarefa M14-1 - 12 segundos - 8 instantes
Subtarefa M15-1 - 1 segundos - 15 instantes	Subtarefa M15-1 - 1 segundos - 9 instantes
Subtarefa M15-2 - 2440 segundos - 26 instantes	Subtarefa M15-2 - 304 segundos - 22 instantes
Subtarefa M16-1 - 2 segundos - 15 instantes	Subtarefa M16-1 - 2 segundos - 8 instantes
Subtarefa M17-1 - 51 segundos - 15 instantes	Subtarefa M17-1 - 23 segundos - 8 instantes
Subtarefa M17-2 - 2441 segundos - 26 instantes	Subtarefa M17-2 - 1 segundos - 6 instantes
Subtarefa M18-1 - 206 segundos - 26 instantes	Subtarefa M18-1 - 36 segundos - 24 instantes
Subtarefa M18-2 - 102 segundos - 26 instantes	Subtarefa M18-2 - 48 segundos - 19 instantes
Subtarefa M19-2 - 3 segundos - 15 instantes	Subtarefa M19-2 - 5 segundos - 17 instantes

D Resultados Adicionais LS–Heur

Devido ao comportamento estocástico de LS–Heur Estocástica, foram realizadas 10 repetições para o teste de cada instância. No texto principal foram apresentados os valores de mediana, melhor e pior caso. Neste apêndice são disponibilizados os resultados de todas as repetições para todas as instâncias.

Tabela 61 – Resultados encontrados pela LS–Heur Estocástica na Instância A.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	36	395	55	113
2	0	0	36	392	56	77
3	0	0	36	395	54	108
4	0	0	36	395	55	114
5	0	0	36	392	56	77
6	0	0	36	392	56	77
7	0	0	36	395	55	113
8	0	0	36	395	54	108
9	0	0	36	392	56	77
10	0	0	36	395	55	114

Tabela 62 – Resultados encontrados pela LS–Heur Estocástica na Instância B.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	12	520	66	140
2	0	0	12	520	66	140
3	0	0	12	520	67	142
4	0	0	12	520	66	140
5	0	0	12	520	66	140
6	0	0	12	520	66	141
7	0	0	12	520	65	143
8	0	0	12	520	66	140
9	0	0	12	520	66	139
10	0	0	12	520	67	141

Tabela 63 – Resultados encontrados pela LS–Heur Estocástica na Instância C.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	12	1057	73	103
2	0	0	12	1057	73	103
3	0	0	12	1057	73	103
4	0	0	12	1057	73	103
5	0	0	12	1057	73	103
6	0	0	12	1057	73	103
7	0	0	12	1057	73	103
8	0	0	12	1057	73	103
9	0	0	12	1057	73	103
10	0	0	12	1057	73	103

Tabela 64 – Resultados encontrados pela LS–Heur Estocástica na Instância D.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	83	117	101	235
2	0	0	74	117	108	302
3	0	0	83	117	100	301
4	0	0	83	117	101	234
5	0	0	74	117	99	301
6	0	0	83	117	101	237
7	0	0	74	117	99	297
8	0	0	79	438	94	304
9	0	0	83	117	101	236
10	0	0	83	117	101	237

Tabela 65 – Resultados encontrados pela LS–Heur Estocástica na Instância E.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	21	1771	83	246
2	0	0	21	1848	99	292
3	0	0	21	1513	82	234
4	0	0	21	1838	104	305
5	0	0	21	1807	90	271
6	0	0	21	1691	85	232
7	0	0	21	1804	102	256
8	0	0	21	1685	83	262
9	0	0	21	1687	89	302
10	0	0	21	1768	91	268

Tabela 66 – Resultados encontrados pela LS–Heur Estocástica na Instância F.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	18	3592	92	326
2	0	0	18	3592	91	323
3	0	0	18	3601	93	321
4	0	0	18	3605	107	307
5	0	0	18	3601	91	305
6	0	0	18	3604	94	315
7	0	0	18	3601	90	302
8	0	0	18	3606	97	322
9	0	0	18	3601	92	321
10	0	0	18	3592	91	324

Tabela 67 – Resultados encontrados pela LS–Heur Estocástica na Instância G.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	38	527	90	240
2	0	0	31	424	96	254
3	0	0	31	424	96	255
4	0	0	31	424	96	255
5	0	0	32	460	80	252
6	0	0	29	864	82	259
7	0	0	32	460	80	250
8	0	0	35	460	84	208
9	0	0	32	460	80	250
10	0	0	38	527	91	199

Tabela 68 – Resultados encontrados pela LS–Heur Estocástica na Instância H.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	81	1951	76	208
2	0	0	91	1951	76	206
3	0	0	81	1954	78	291
4	0	0	91	1951	76	247
5	0	0	91	1951	76	230
6	0	0	91	1951	76	206
7	0	0	92	1948	74	197
8	0	0	92	1944	77	202
9	0	0	92	1948	73	251
10	0	0	92	1948	74	204

Tabela 69 – Resultados encontrados pela LS–Heur Estocástica na Instância I.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	37	1624	60	124
2	0	0	37	1579	52	118
3	0	0	37	1627	59	127
4	0	0	37	1579	50	118
5	0	0	37	1624	59	124
6	0	0	37	1624	60	127
7	0	0	37	1789	53	126
8	0	0	37	1789	55	133
9	0	0	37	1789	55	133
10	0	0	37	1627	59	126

Tabela 70 – Resultados encontrados pela LS–Heur Estocástica na Instância J.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	141	34	63	240
2	0	0	138	34	62	253
3	0	0	137	34	63	253
4	0	0	143	34	73	309
5	0	0	138	34	65	295
6	0	0	169	61	65	290
7	0	0	169	61	65	251
8	0	0	141	34	62	232
9	0	0	155	34	59	251
10	0	0	141	34	66	285

Tabela 71 – Resultados encontrados pela LS–Heur Estocástica na Instância K

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	142	57	78	280
2	0	0	142	57	77	248
3	0	0	135	44	79	220
4	0	0	137	54	77	248
5	0	0	154	60	82	242
6	0	0	142	57	79	243
7	0	0	157	60	84	250
8	0	0	135	44	79	224
9	0	0	157	60	86	256
10	0	0	160	54	88	259

Tabela 72 – Resultados encontrados pela LS–Heur Estocástica na Instância L.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	60	0	51	104
2	0	0	60	0	54	103
3	0	0	60	0	54	103
4	0	0	60	0	54	102
5	0	0	61	0	51	107
6	0	0	60	0	54	103
7	0	0	61	0	51	107
8	0	0	60	0	51	106
9	0	0	61	0	51	107
10	0	0	60	0	51	105

Tabela 73 – Resultados encontrados pela LS–Heur Estocástica na Instância M.

Repetição	F_1	F_2	F_3	F_4	F_5	T(s)
1	0	0	11	15	44	122
2	0	0	20	15	44	120
3	0	0	18	15	44	126
4	0	0	11	15	45	163
5	0	0	11	15	44	123
6	0	0	20	15	44	120
7	0	0	20	15	44	120
8	0	0	18	15	44	164
9	0	0	18	15	44	126
10	0	0	18	15	44	125