UNIVERSIDADE FEDERAL DE OURO PRETO

Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo: Um Estudo de Caso Industrial de Larga Escala

Roberto Dias Aquino
Universidade Federal de Ouro Preto

Orientador: Marcone Jamilson Freitas Souza

Coorientador: Jonatas Batista Costa das Chagas

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto, como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Ouro Preto, Novembro de 2018

Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo: Um Estudo de Caso Industrial de Larga Escala

Roberto Dias Aquino
Universidade Federal de Ouro Preto

Orientador: Marcone Jamilson Freitas Souza

Coorientador: Jonatas Batista Costa das Chagas

A56a Aquino, Roberto Dias .
Abordagem exata e heurísticas para o problema de planejamento de ordens de manutenção de longo prazo [manuscrito]: um estudo de caso industrial de larga escala / Roberto Dias Aquino. - 2018.
95f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Coorientador: Prof. MSc. Jonatas Batista Costa das Chagas.

Dissertação (Mestrado) - Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Programação (Matemática) . 2. Programação heurística. 3. Otimização combinatória. I. Souza, Marcone Jamilson Freitas. II. Chagas, Jonatas Batista Costa das. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 004.421.2

Catalogação: www.sisbin.ufop.br

Ata da Defesa Pública de Dissertação de Mestrado

Aos 30 dias do mês de novembro de 2018, às 14:00 horas na Sala de Seminários do DECOM no Instituto de Ciências Exatas e Biológicas (ICEB), reuniram-se os membros da banca examinadora composta pelos professores **Prof. Dr. Marcone Jamilson Freitas Souza (presidente e orientador), Prof. Dr. Marco Antonio Moreira de Carvalho, Prof. Dr. Sérgio Ricardo de Souza e MSc. Jonatas Batista Costa das Chagas** aprovada pelo Colegiado do Programa de Pós-Graduação em Ciência da Computação, a fim de arguirem o mestrando **Roberto Dias Aquino**, com o título **"Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo: Um Estudo de Caso Industrial de Larga Escala"**. Aberta a sessão pelo presidente, coube ao candidato, na forma regimental, expor o tema de sua dissertação, dentro do tempo regulamentar, sendo em seguida questionado pelos membros da banca examinadora, tendo dado as explicações que foram necessárias.

Recomendações da Banca:

(X) Aprovada sem recomendações

( ) Reprovada

( ) Aprovada com recomendações: _____

Banca Examinadora:

Prof. Dr. Marcone Jamilson Freitas Souza

Prof. Dr. Marco Antonio Moreira de Carvalho

Prof. Dr. Sérgio Ricardo de Souza

MSc. Jonatas Batista Costa das Chagas

Prof. Dr. Joubert de Castro Lima
Coordenador do Programa de Pós-Graduação em Ciência da Computação
DECOM/ICEB/UFOP

Ouro Preto, 30 de novembro de 2018.

Dedico este trabalho à minha esposa Telma Cristina Ruas Aquino

Resumo

Este trabalho propõe uma modelagem de programação linear inteira mista e algoritmos meta-heurísticos para um problema real de planejamento de manutenção de longo prazo para uma planta de beneficiamento de minério de ferro no Brasil. Este é um problema complexo de programação de ordens de manutenção preventiva, para o qual é necessário atribuir ordens de manutenção preventiva para as equipes de trabalho disponíveis em um horizonte de 52 semanas. Foi desenvolvido um modelo de programação inteira mista e os resultados foram utilizados como um *benchmark*. Como o modelo não foi capaz de resolver a instância real, foram propostos algoritmos meta-heurísticos para resolvê-la. Esses algoritmos foram baseados nos métodos *Simulated Annealing*, *Variable Neighborhood Search*, *Multi-Start*, *Biased Random-Key Genetic Algorithm* e algoritmos meméticos. Os algoritmos heurísticos desenvolvidos foram capazes de resolver a instância real, assim como melhorar a maioria dos resultados das instâncias de dimensões menores, levando a novos *benchmarks*.

Palavras-chave: Manutenção preventiva, Programação de ordens de manutenção, Escalonamento, *Simulated Annealing*, *Variable Neighborhood Search*, Otimização combinatória, Algoritmos Genéticos, Meta-heurísticas.

Abstract

In this work we propose a Mixed Integer Linar Programming (MILP) model and metaheuristic approaches for the long-term maintenance programming of an iron ore processing plant of a company in Brazil. The problem is a complex maintenance programming where machine preventive programming orders have to be assigned to the available work teams over a 52-week planning. In order to evaluate our solution we developed a general mixed integer programming model and used the numerical results as the benchmark. As the proposed model was not able for solving the real instance, then we have also proposed metaheuristic approaches based on *Simulated Annealing*, *Variable Neighborhood Search*, *Multi-Start*, *Biased Random-Key Genetic Algorithm* and *Memetic Algorithms*. The proposed metaheuristic approaches were able to solve the real instance and also improved most of the small instances leading to new benchmarks.

Keywords: Long-term maintenance programming, Scheduling, *Simulated Annealing*, *Variable Neighborhood Search*, Combinatorial Optimization,Genetic Algorithms, Metaheuristics.

Declaração

Esta dissertação é resultado de meu próprio trabalho, exceto quando referência explícita é feita ao trabalho de outros, e não foi submetida para outra dissertação nesta nem em outra universidade.

Roberto Dias Aquino

Agradecimentos

À minha família, que sempre me apoiou e acreditou em mim.

Um agradecimento especial ao meu orientador Prof. Dr. Marcone Jamilson Freitas Souza e ao meu coorientador MSc Jonatas Batista Costa das Chagas pela incansável ajuda, inestimável apoio, ensino e confiança.

Aos amigos da Vale. Em especial ao Carlos Soares, Felipe Caldas, Juliene Oliveira e Vicentino Rodrigues pelo apoio e incentivo para a realização desta dissertação.

À UFOP, por prover um ensino de qualidade.

Ao Programa de Pós-graduação em Ciência da Computação (PPGCC/UFOP) pela oportunidade de formação.

À Pró-Reitoria de Pesquisa e Pós-Graduação (PROPP/UFOP), pelo fomento e apoio à pesquisa, pós-graduação e inovação.

Ao professor Dr. Alan Robert Resende de Freitas, pela grande ajuda para a análise estatística.

À Mariana Ferreira Lanna, pelo apoio.

À CAPES, FAPEMIG, CNPq, pela disponibilização de recursos para financiar as pesquisas.

Sumário

1 Introdução **1**

1.1 Objetivos 3

1.2 Motivação 3

1.3 Metodologia 4

1.4 Estrutura do Trabalho 5

2 Conceitos Básicos **7**

2.1 Manutenção Industrial 7

2.2 Planejamento e Controle da Manutenção 8

2.3 Problemas de Sequenciamento 10

2.4 Planejamento de Ordens de Manutenção Preventiva de Longo Prazo . . . 12

2.5 Conclusões Parciais 14

3 Revisão da Literatura **15**

3.1 Estratégias de Manutenção 15

3.2 Problemas Relacionados 16

3.3 Conclusões Parciais 18

4 Uma Formulação de Programação Matemática para o PPOMPLP **19**

4.1 Notação 19

4.2 Modelo 20
4.3 Conclusões Parciais 22

5 Algoritmos Heurísticos para o PPOMPLP 23

5.1 Representação da Solução 23
5.2 Algoritmo de Alocação de Ordens de Manutenção 24
5.3 Solução Inicial e Estrutura de Vizinhança 30
5.4 *Variable Neighborhood Search* 32
5.5 *Multi-Start Variable Neighborhood Search* 34
5.6 *Simulated Annealing* 35
5.7 *Biased Random-Key Genetic Algorithm* 36
5.8 *Biased Random-Key Memetic Algorithm* 40
5.9 *Conclusões Parciais* 42

6 Experimentos e Resultados Computacionais 45

6.1 Descrição das Instâncias 46
6.2 Comparação entre MILP, VNS e SA 47
6.3 Comparação entre MSVNS, BRKGA e BRKMA 53

7 Conclusões e Trabalhos Futuros 63

7.1 Publicações 65

A Comparação entre os algoritmos SA, BRKGA e BRKMA 71

Capítulo 1

Introdução

O objetivo de qualquer indústria é a produção com qualidade e quantidade, de acordo com a demanda do mercado; no entanto, para que o empreendimento tenha viabilidade econômica, é necessário minimizar o custo de produção. Moreira (2012) aborda os conceitos da administração da produção na empresa moderna, trabalho este que aponta diversos fatores que podem contribuir para que esse objetivo seja alcançado. Dentre os objetivos mencionados por Moreira (2012), a folha de pagamento tem, evidentemente, um papel importante. Além disso, é fato que há indústrias que disponibilizam menos recursos financeiros para contratação de mão de obra do que a sua legítima demanda. Este cenário faz com que a disputa pela utilização da mão de obra disponível fique ainda mais acirrada, fazendo-se necessária uma correta utilização dos recursos humanos já disponíveis nas indústrias a fim de aumentar a produtividade e a qualidade.

Uma prática muito frequente em grandes indústrias é a realização de manutenções preventivas em seus equipamentos, manutenções estas que buscam manter o sistema de produção em boas condições de funcionamento. O objetivo é corrigir eventuais falhas antes que elas ocorram, de forma a evitar a interrupção da produção. Essa prática é fortemente adotada, visto que o custo das manutenções corretivas, aquelas realizadas após falhas do equipamentos, é significativamente maior.

As manutenções preventivas são realizadas periodicamente baseando-se na experiência técnica ou nos manuais dos fabricantes. A realização delas consome uma parcela considerável do recurso humano disponível nas indústrias e, por consequência, é uma parte do orçamento anual que deve ser otimizada. Visto tal cenário, é fácil concluir que um bom planejamento das ordens de manutenções preventivas a serem realizadas pelas

equipes de trabalho resulta em um menor custo para as indústrias.

Em grandes empresas é comumente realizado um planejamento de todas as manutenções preventivas a serem executadas durante todo o ano. Todas essas manutenções são compiladas em um documento chamado de mapa de 52 semanas, no qual cada manutenção, ou, mais especificamente, cada ordem de manutenção, é relativa a uma atividade específica de manutenção. A manutenção deve ser executada em uma determinada janela de tempo em um equipamento por uma equipe especializada naquela atividade e com uma duração pré-determinada.

Este trabalho trata um estudo de caso relativo às manutenções preventivas de uma unidade de beneficiamento de minério de ferro, localizada no Estado de Minas Gerais. Nessa unidade, o mapa de 52 semanas é realizado por uma equipe de engenharia de manutenção especializada com o auxílio de um software de programação de manutenção. O produto final entregue pela equipe é o mapa de 52 semanas.

Neste cenário, quando todas as ordens de manutenção são mapeadas em um único documento, é possível notar a sobreposição de algumas delas em um único equipamento na mesma janela de tempo. Isso ocorre porque cada área de manutenção (ex.: mecânica, elétrica, lubrificação) faz um planejamento separado para o que seria ideal para cada equipamento (o quê, quando, onde, como e por qual especialidade). Além disso, como o planejamento é focado no equipamento, não é verificada a disponibilidade de mão de obra necessária para a realização das manutenções, apenas a área. Dessa forma, é de responsabilidade da a equipe de manutenção de curto prazo (planejamento do mês), a alocação real das manutenções. O problema é que, atualmente, na unidade estudada, menos de 50% das ordens de manutenção são realizáveis sem alocação de mão de obra extra ou contratação de mão de obra externa.

Visto que um melhor planejamento das ordens de manutenções preventivas é necessário para maximizar o número de ordens de manutenção executadas e minimizar a mão de obra necessária, este trabalho propõe uma modelagem matemática para o problema, assim como abordagens heurísticas, com o objetivo de encontrar o melhor ou bons planejamentos das ordens de manutenções no cenário abordado.

1.1 Objetivos

Este trabalho tem como objetivo geral desenvolver algoritmos capazes de gerar soluções viáveis para o problema de alocação de ordens de manutenção de longo prazo utilizando técnicas de programação matemática inteira mista e meta-heurísticas.

Especificamente, tem-se os seguintes objetivos:

- Estudar um problema real de manutenção preventiva com grande relevância na indústria, relativo a uma unidade de beneficiamento de minério de ferro situada no Estado de Minas Gerais;
- Propor um modelo exato para servir como base para estudos de técnicas computacionais de otimização;
- Propor instâncias teste para disponibilizá-las para estudos posteriores;
- Desenvolver e aplicar algoritmos heurísticos de otimização capazes de resolver instâncias reais;
- Realizar experimentos computacionais e análises da eficiência dos algoritmos propostos.

1.2 Motivação

As manutenções preventivas têm grande impacto nos resultados financeiros das grandes empresas, tanto em termos de produtividade como em termos de custo. Uma manutenção preventiva corretamente realizada gera uma maior segurança que os equipamentos irão funcionar de acordo com o planejado, sem paradas para manutenções corretivas.

A alocação das ordens de manutenções programadas do ano para os equipamentos e para a mão de obra disponível é realizada na empresa estudada por uma equipe de engenharia de manutenção e, geralmente, envolve muitas ordens de manutenção, não sendo possível alocar todas as ordens necessárias. Uma alocação eficiente dessas ordens de manutenção pode aumentar o número de ordens realizadas ou diminuir a mão de obra necessária. Dessa forma, pode-se ter um maior índice de realização das manutenções. Quanto maior o índice, menor a probabilidade de ocorrência de paradas para manuten-

ções corretivas e, em consequência, menor custo para as empresas. Assim, destaca-se o grande interesse prático na solução deste problema.

Por outro lado, há interesse de pesquisa em métodos de solução para o problema, visto que ele é da classe NP-difícil. De fato, esse problema se reduz ao de Sequenciamento em Máquinas Paralelas Não-relacionadas, ao considerarmos as ordens de serviço como tarefas e as equipes como máquinas. Assim, se justifica o desenvolvimento de algoritmos heurísticos que possam resolvê-lo.

1.3 Metodologia

Este trabalho está dividido em quatro partes, as quais refletem as etapas de seu desenvolvimento.

A primeira fase foi o desenvolvimento de um modelo matemático linear inteiro misto, a fim de resolver o problema tratado de forma exata. O objetivo desta fase foi obter um conjunto de soluções para serem usadas como *benchmarking*.

Como o modelo foi capaz de resolver apenas instâncias de pequenas dimensões, uma segunda fase teve como objetivo a resolução de problemas de dimensões maiores. Esta fase foi dividida em duas etapas. A primeira foi a proposição de uma representação indireta da solução, com o objetivo de facilitar a exploração do espaço de soluções viáveis do problema. Consequentemente, foi proposto um algoritmo simples de alocação das manutenções às equipes a partir da representação indireta definida. A segunda etapa consistiu na implementação da meta-heurística *Simulated Annealing* (SA). Nestas duas primeiras fases já foram obtidas soluções viáveis e melhores do que as realizadas na indústria em estudo. Os resultados do desenvolvimento até essa fase foram apresentados na *International Conference on Intelligent Systems Design and Applications - ISDA 2017* (Aquino et al., 2017).

A terceira fase teve como objetivo a implementação de uma outra abordagem meta-heurística e a comparação com os resultados já obtidos. Nesta fase foi implementada a meta-heurística *Variable Neighborhood Search* (VNS). Os resultados obtidos com o VNS foram apresentados na *20th International Conference on Enterprise Information Systems – ICEIS 2018* (Aquino et al., 2018).

Apesar dos ganhos observados com a implementação do VNS e do SA, as instâncias de maior dimensão consumiam tempo computacional excessivo. Portanto, uma quarta

fase foi então proposta, com o objetivo de utilizar processamento paralelo para chegar a soluções para as instâncias de dimensões maiores em um tempo computacional menor. Foram, então, implementadas as meta-heurísticas *Multi-Start Variable Neighborhood Search* (MSVNS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Biased Random-Key Memetic Algorithm* (BRKMA).

1.4 Estrutura do Trabalho

O restante deste trabalho está estruturado como segue. No Capítulo 2 são apresentados conceitos básicos relacionados aos processos de manutenção, bem como as características específicas do problema abordado. A revisão da literatura é realizada no Capítulo 3, onde são discutidas as principais características dos problemas clássicos de sequenciamento, bem como é apresentado um comparativo com o problema abordado neste trabalho. O Capítulo 4 descreve formalmente o problema por meio de uma formulação de programação matemática. O Capítulo 5 apresenta os algoritmos meta-heurísticos desenvolvidos para resolver o problema. Já no Capítulo 6 são apresentados os experimentos computacionais. Finalmente, o Capítulo 7 conclui esta dissertação e apresenta sugestões para trabalhos futuros.

Capítulo 2

Conceitos Básicos

Este capítulo inicia-se com uma breve descrição dos processos de manutenção propriamente ditos, baseados em conceitos amplamente discutidos em Pinto and Nascif (2009), Lafraia (2001) e em Jonge (2017). Em seguida, são apresentados conceitos gerais de problemas de sequenciamento e descrito o problema tratado neste trabalho.

2.1 Manutenção Industrial

A manutenção industrial consiste em um conjunto de intervenções necessárias para que as máquinas e equipamentos tenham uma disponibilidade previamente estabelecida. A manutenção é normalmente dividida em corretiva, preventiva e preditiva, conforme breve explicação a seguir:

- Manutenção corretiva: aquela que é realizada quando é detectado algum problema que impeça o desempenho, o funcionamento parcial ou o funcionamento total de uma máquina ou equipamento. Apesar de a manutenção corretiva ser realizada em qualquer indústria, raramente ela é considerada uma estratégia de manutenção e só é realizada quando algo inesperado ocorre. Apenas quando o custo total da manutenção corretiva somado com os custos de perda de produção é muito inferior à de uma estratégia de manutenção preventiva ou preditiva é que vale a pena adotar a manutenção corretiva como estratégia de manutenção.
- Manutenção preventiva: aquela que é programada de acordo com recomendações do fabricante ou da experiência da equipe de engenharia e manutenção. A manu-

tenção preventiva tem o objetivo de minimizar a ocorrência de eventos inesperados, reduzindo o número de manutenções corretivas. A manutenção preventiva é realizada de acordo com uma frequência pré-determinada ou de acordo com o número de horas de operação. Esta é uma estratégia de manutenção baseada no tempo e estudada há muito tempo, como discutido em Barlow and Hunter (1960).

- Manutenção preditiva: esta manutenção pode ser considerada como um caso específico de manutenção preventiva, mas se diferencia na utilização de técnicas, equipamentos e softwares que são capazes de predizerem o momento ideal de realização da manutenção. Ao contrário da manutenção preventiva, a manutenção preditiva propriamente dita não é realizada de forma pré-determinada; no entanto, é necessária a programação das inspeções e análises a serem realizadas. Apesar de a manutenção preditiva ser mais atraente, é importante destacar que o monitoramento da condição do equipamento pode ser tecnicamente inviável ou, se viável, o custo do monitoramento pode não compensar a adoção dessa estratégia.

2.2 Planejamento e Controle da Manutenção

O Planejamento e Controle da Manutenção (PCM) é a área responsável por garantir que todas as máquinas e equipamentos tragam o máximo retorno para a indústria, garantindo que todos os indicadores de desempenho dos equipamentos estejam dentro do programado. Os principais indicadores de desempenho estão listados a seguir:

- Disponibilidade Física (DF): é a relação entre o número de horas que o equipamento está disponível para operação e o número total de horas;
- Índice de execução: é o número de Ordens de Serviço (OS) executadas em relação ao número de OS planejadas em um determinado período;
- Tempo médio entre falhas (*Mean Time Between Failures* - MTBF): de acordo com Wendy and Victor (2018), é o tempo médio decorrido entre todas as falhas, medidos entre o final de uma falha e o início da próxima falha;
- Tempo médio de reparo (*Mean Time To Repair* - MTTR): de acordo com Wendy and Victor (2018), é o tempo médio da execução de um reparo após a ocorrência da falha, ou seja, é o tempo total das intervenções dividido pelo número de intervenções.

Para garantir o seu funcionamento, um PCM é subdividido em quatro funções: planejamento, programação, supervisão e controle. Cada uma dessas funções é brevemente descrita a seguir:

- Planejamento: é responsável pelo desenvolvimento e atendimento de longo prazo dos planos de manutenção e inspeção preventivas, atendimento a demandas específicas da área de operação e projetos, assim como atendimento a manutenções corretivas. O plano de manutenção e inspeção é um produto da área de Engenharia de Manutenção e é principal insumo da manutenção preventiva. Este plano consiste em um conjunto de ordens de serviço com todas as atividades que devem ser realizadas para cada equipamento. Além da descrição da atividade, cada OS define qual o perfil do profissional que tem a habilidade para executá-la, qual a duração e qual a prioridade. Para a definição deste plano, a área de Engenharia de Manutenção leva em consideração os planos recomendados pelos fabricantes, a experiência prévia da própria área de engenharia e de manutenção, bem como parâmetros de prioridade de acordo com o impacto nas áreas de saúde, meio ambiente, produção, qualidade e custos. O produto final desta área é um mapa de manutenção preventiva de 52 semanas, objeto de estudo deste trabalho.

- Programação: é responsável pelo planejamento de curto prazo, com o objetivo de garantir a execução diária de todas as atividades planejadas, assim como a negociação de serviços não programados e emergenciais. O programador é responsável por negociar a prioridade das atividades, definir os recursos humanos e materiais e definir o horário e ordem de execução de cada Ordem de Serviço.

- Supervisão: a principal atividade da supervisão é a coordenação das atividades junto ao profissional de manutenção no campo, de forma a garantir o cumprimento da programação. Para isso, o supervisor é responsável por garantir que todos os recursos necessários para a atividade sejam providenciados, assim como atuar na resolução de qualquer problema ocorrido na execução dos serviços.

- Controle: é responsável pelo gerenciamento da execução e melhoria contínua de todo o processo de manutenção. O profissional responsável pelo controle deve gerenciar a apropriação dos serviços executados; o orçamento; gerar os principais indicadores relacionados à eficiência da manutenção; e gerar planos de ação, com o objetivo de melhoria contínua de todo o processo.

2.3 Problemas de Sequenciamento

Problemas de sequenciamento são aqueles que envolvem a alocação de n tarefas em m máquinas, sendo que cada tarefa demanda um conjunto de serviços e/ou recursos, enquanto as máquinas ofertam uma quantidade limitada de tais serviços e recursos. O objetivo é alocar todas as tarefas, de forma a otimizar uma função objetivo, respeitando-se as restrições do problema.

Segundo Pinedo (2008), tais problemas podem ser classificados pela notação $\alpha \,|\, \beta \,|\, \gamma$. No campo α estão as características referentes às máquinas, as quais podem assumir apenas um dentre os seguintes valores:

- 1 (*Single machine*): existe apenas uma máquina para executar todas as tarefas;
- P_m (*Identical machines in parallel*): há m máquinas idênticas em paralelo disponíveis para executar as tarefas. As tarefas podem ser executadas por qualquer máquina ou por qualquer máquina de um subconjunto.
- Q_m (*Machines in parallel with different speeds*): há m máquinas disponíveis, sendo que cada máquina i pode ter uma velocidade de processamento v_i diferente das demais;
- R_m (*Unrelated machines in parallel*): há m máquinas disponíveis, sendo que a velocidade de execução da tarefa j na máquina i é determinada por v_{ij};
- F_m (*Flow shop*): há m máquinas em série e cada tarefa tem que ser executada em cada uma das m máquinas. Todas as tarefas seguem a mesma sequência;
- FF_c (*Flexible flow shop*): há c estágios em série e em cada estágio há um número de máquinas em paralelo. As tarefas têm quer seguir a mesma ordem de estágios;
- J_m (*Job shop*): há m máquinas disponíveis, e cada tarefa segue sua própria rota de máquinas;
- FJ_c (*Flexible job shop*): há c estágios em série e em cada estágio há um número de máquinas em paralelo. Cada tarefa segue sua própria rota de estágios;
- O_m (*Open shop*): não há uma sequência preestabelecida para o processamento das tarefas.

No campo β estão as características das tarefas, as quais podem assumir apenas valores entre os seguintes:

- r_j (*Release dates*): a data de liberação de uma tarefa é o instante a partir da qual a tarefa j pode ser iniciada. Se não for especificada, a tarefa pode ser iniciada a qualquer instante;
- *prmp* (*Preemptions*): a preemptividade significa que uma tarefa pode ser parada e depois retomada na mesma máquina ou em outra máquina a partir do ponto que parou;
- *prec* (*Precedence constraints*): define que uma tarefa só pode ser iniciada após a finalização de um conjunto de tarefas;
- s_{jk} (*Sequence dependent setup times*): define que se considera o tempo de preparação; (*setup*) entre uma tarefa j e uma tarefa k.
- $fmls$ (*Job families*): há F famílias de tarefas. Para as tarefas que estão na mesma família, não há necessidade de um tempo de *setup*. O tempo de *setup* entre duas famílias g e h é definido por s_{gh};
- *batch*(b) (*Batch processing*): uma máquina pode executar b bateladas de tarefas. O tempo de processamento de uma batelada é o tempo de finalização da última tarefa da batelada;
- *brkdwn* (*Breakdowns*): define que há períodos de indisponibilidade das máquinas, ou seja, os períodos nos quais nenhuma tarefa poderá ser executada.
- M_j (*Machine eligibility restrictions*): define que há um conjunto de máquinas que poderá executar as tarefas;
- *prmu* (*Permutation*): define que a mesma ordem que as tarefas são executadas na primeira máquina deve ser seguida para todas as outras máquinas;
- *block* (*Blocking*): se há um *buffer* entre duas máquinas, o *blocking* impede que uma tarefa seja liberada se o *buffer* estiver cheio.
- *nwt* (*No-wait*): define que uma tarefa não pode esperar entre duas máquinas;
- *rcrc* (*Recirculation*): permite que uma tarefa seja executada em uma máquina mais de uma vez.

Segundo Pinedo (2008), a função objetivo é calculada em função do tempo de finalização das tarefas. A finalização (*completion time*) de uma tarefa j em uma máquina i é definida por C_{ij}. O tempo de finalização de uma tarefa j é definida por C_j. O campo γ identifica a função objetivo, conforme a seguir:

- $C_{\max}$ (*Makespan*): é o instante de finalização da última tarefa executada;
- $L_{\max}$ (*Maximum Lateness*): é a maior violação do horário de término de uma tarefa;
- $\sum_j w_j C_j$ (*Total weighted completion time*): é a soma ponderada de todos os tempos de finalização das tarefas;
- $\sum_j w_j(1 - e^{-rC_j})$ (*Discounted total weighted completion time*): igual ao anterior, mas obedece a uma taxa para desconto por atraso na finalização das tarefas;
- $\sum_j w_j T_j$ (*Total weighted tardiness*): soma ponderada dos tempos de atraso;
- $\sum_j w_j U_j$ (*Weighted number of tardy jobs*): soma ponderada do número de tarefas atrasadas.

Em Brucker (2007) é encontrado o conceito de multiprocessamento de tarefas *Multiprocessor Tasks* (MPT). Neste conceito, cada máquina é considerada um processador e uma tarefa é executada por um conjunto de processadores simultaneamente. As tarefas que precisam ser executadas em um mesmo processador são **incompatíveis** e as que não têm nenhum processador em comum são **compatíveis**. O MPT é estendido e discutido para os problemas de *general shop* (GMPT), *job shop* (JMPT), *flow shop* (FMPT), e *open shop* (OMPT).

2.4 Planejamento de Ordens de Manutenção Preventiva de Longo Prazo

O Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP) consiste na atribuição das ordens de manutenção para as equipes de manutenção (grupo de trabalhadores que devem trabalhar juntos) no período de 52 semanas. O produto final deste problema é um mapa de 52 semanas, no qual é possível ver o planejamento anual de todas as ordens de manutenção preventivas.

Na construção desse mapa de 52 semanas, diversas restrições devem ser consideradas, sendo que a maioria delas está diretamente relacionada a problemas de escalonamento e sequenciamento, que são muito estudados na literatura. Uma ordem de manutenção tem que ser alocada simultaneamente a uma equipe de trabalho e a um equipamento, sendo que a equipe de trabalho não pode executar mais de uma ordem de manutenção simultaneamente e cada equipamento não pode ter mais de uma ordem de manutenção sendo executada simultaneamente. Estas duas alocações simultâneas são as principais restrições que caracterizam o problema; há um limite de horas que cada time de trabalho pode trabalhar; há uma janela de tempo que cada ordem de manutenção deve ser executada; cada ordem de manutenção deve ser executada por um tipo específico de mão de obra; pode haver mais de uma equipe de trabalho capaz de executar a mesma ordem de manutenção.

Se considerarmos uma simplificação do PPOMPLP que consista apenas na alocação das ordens de manutenção às equipes de trabalho, ou seja, sem a alocação simultânea das ordens de manutenção às equipes e aos equipamentos, ele pode ser comparado ao problema de sequenciamento em máquinas paralelas. Nesta comparação, as ordens de manutenção seriam as tarefas e as equipes de trabalho seriam as máquinas. Cada grupo de equipes com a mesma especialidade seria um conjunto de máquinas paralelas. Nessa analogia, o problema abordado neste trabalho seria definido como $P_m \,|\, r_j M_j \,|\, \gamma$, no qual o campo γ seria definido de forma a minimizar a mão de obra necessária para executar o maior número de tarefas. A principal diferença seria na função objetivo, pois as funções objetivo dos problemas de sequenciamento são todas relacionadas ao tempo final de alguma tarefa. Como o tempo final é determinado em função do sequenciamento das tarefas, todas as tarefas são alocadas. Já no problema foco deste trabalho, o tempo é fixo e o objetivo é alocar o máximo de tarefas com o mínimo de "máquinas". Dessa forma, não é garantida a alocação de todas as tarefas.

O conceito de multiprocessamento de tarefas pode ser aplicado ao PPOMPLP se considerarmos dois processamentos; um seria pela equipe de trabalho, e o outro pelo equipamento no qual a manutenção deve ser executada. Uma adaptação necessária seria a consideração de que um dos processadores seria um conjunto de máquinas paralelas, pois várias equipes de trabalho são compatíveis para a execução de uma tarefa. Além disso, é necessário mudar a função objetivo (conforme já discutido anteriormente). Nessa analogia, o problema é definido como $MPT2_m \,|\, r_j M_j \,|\, \gamma$.

Na empresa em estudo, o mapa de 52 semanas é feito por uma gerência de engenharia de manutenção utilizando o módulo de manutenção preventiva do software SAP da

empresa alemã SAP SE [1]. Este módulo faz o gerenciamento de alto nível das manutenções preventivas e gerencia seus principais aspectos, tais como ordem de trabalho, inventário, gerenciamento de mão de obra, bem como o gerenciamento de materiais e ferramentas. Além disso, o módulo armazena um histórico de todas as manutenções já realizadas e todos os planos de manutenção. O plano de manutenção de cada equipamento inclui todas as tarefas de manutenção preventiva que devem ser realizadas, assim como a frequência de cada tarefa, a equipe de trabalho que tem as habilidades necessárias, a duração esperada, e os materiais e as ferramentas necessários para execução. A partir dessa base de dados, é gerado o mapa de 52 semanas.

O problema é que esse módulo de manutenção não leva em consideração as restrições do problema e, assim, pode gerar soluções inviáveis. Para tentar executar o máximo de manutenções possíveis, o equipe de programação de manutenção de curto prazo, ou seja, considerando o horizonte de 30 dias, refaz a programação, incluindo horas extras e a utilização de mão de obra de empresas terceirizadas, o que eleva o custo da manutenção. Apesar dessa flexibilidade, nem todas as manutenções preventivas são realizadas. Portanto, o desafio deste trabalho é justamente a proposição de uma forma mais eficiente de alocação dessas manutenções às equipes disponíveis.

2.5 Conclusões Parciais

Para a contextualização do trabalho, neste capítulo foram introduzidos os conceitos de manutenção industrial (corretiva, preventiva e preditiva); foi explicado o funcionamento da área responsável pelo planejamento das manutenções (PCM); introduzido os principais conceitos dos problemas de sequenciamento e, finalmente, introduzido o problema Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo e suas principais características.

[1] `https://www.sap.com/corporate/en.html`

Capítulo 3

Revisão da Literatura

Neste Capítulo são apresentados os trabalhos que contribuíram para que o problema objeto de estudo e correlatos fossem formalmente descritos, assim como os que estão relacionados às técnicas de otimização utilizadas neste trabalho.

3.1 Estratégias de Manutenção

Os problemas de otimização de manutenção não são novos. Apesar de não ter sido encontrado na literatura um problema igual ao problema foco deste trabalho, diferentes problemas relacionados à melhoria ou otimização da manutenção foram objeto de estudos em diversos trabalhos. Simões et al. (2011) fizeram uma revisão bibliográfica de artigos relacionados à medição de desempenho das manutenções em 67 periódicos entre 1969 e 2009. Os autores concluíram que a área de gerenciamento e monitoramento ainda precisa de um esforço sistêmico, com o objetivo de solidificar o entendimento teórico e implementar soluções práticas. Os autores também mostraram que o custo das manutenções é o foco da maioria dos trabalhos pesquisados.

Sharma et al. (2011) também fizeram uma grande revisão nesse tema. Foram analisados 104 artigos a partir do início dos anos 60. De acordo com os autores, esses problemas podem ter diferentes critérios de otimização, como taxa de custo de manutenção, lucratividade, utilização física da planta, eficiência da produtividade e segurança do trabalho. Apesar de terem sido revisados alguns artigos relacionados à otimização do custo de manutenção, a maioria dos artigos é focada na área de conhecimento de engenharia de produção, mais especificamente, estratégias de manutenção como *Reliabi-*

lity Centered Maintenance (RCM), *Total Productive Maintenance* (TPM) e *Plant Asset Management* (PAM). O RCM, como pode ser visto em Moubray (1997), é uma estratégia de manutenção que busca a recuperação da confiabilidade do equipamento, ou seja, é um processo de manutenção que define o que deve ser feito para que o equipamento (ou qualquer parte dele) continue a operar da forma na qual foi projetado. De acordo com Agustiady and Cudney (2015), o TPM foi proposto pelo *Japan Institute of Plant Maintenance* e consiste em uma estratégia de manutenção que envolve, além das equipes de manutenção, os operadores dos equipamentos, com o objetivo de integrar as duas áreas e enfatizar a pró atividade de todos em busca de uma melhoria de todo o processo produtivo. Já a estratégia de manutenção PAM, segundo Márquez et al. (2018), consiste de um conjunto de requisitos mínimos de boas práticas que devem ser implementados, mantidos e melhorados em qualquer ativo de uma organização, com o foco de agregar valor em todo o ciclo de vida do ativo.

As estratégias de manutenção mencionadas acima são amplamente difundidas e discutidas na literatura e tem diversos *trade-offs* que necessitam ser discutidos e balanceados para se chegar a uma boa solução. Nesta dissertação considera-se um programa de manutenção na qual todas as ordens de manutenção, a frequência e a duração já estão definidas e o foco é, então, estabelecer a programação dessas ordens.

3.2 Problemas Relacionados

Yamayee et al. (1983) propuseram uma formulação matemática para a solução exata do problema de escalonamento de manutenções preventivas e desenvolveram um *framework* de programação dinâmica para sua resolução. A programação dinâmica foi considerada a técnica mais apropriada para a resolução do problema. O problema estudado consiste no escalonamento de 21 ordens de manutenção com diferentes tamanhos e diferentes custos. Assim como no presente trabalho, a ordem de manutenção, a duração e a janela de tempo já são conhecidos e fazem parte da instância. O objetivo considerado pelos autores, no entanto, é diferente, pois consiste em minimizar o custo esperado de produção e o custo da falta de confiabilidade. Foi observado por eles um grande esforço computacional para resolver o problema e identificado que o motivo é que a função objetivo tem dois componentes. Para estudar o efeito de cada componente da função objetivo, foram realizados dois estudos de caso, um com o custo esperado de produção e o outro com o custo da falta de confiabilidade como função objetivo. Os

escalonamentos foram totalmente diferentes. A análise do estudo de caso mostrou que o custo de produção varia muito pouco nas proximidades do ponto ótimo. Foi observado que a minimização do custo de falta de confiabilidade altera muito pouco o custo de produção. Além disso, o tempo computacional para a minimização do custo da falta de confiabilidade é muito menor e, portanto, é a mais recomendada.

Yao et al. (2004) estudaram um problema de manutenção na indústria de semicondutor. Para resolução do problema, foi proposto um *framework* com duas camadas hierárquicas, estando o planejamento de longo prazo na camada externa e o planejamento de curto prazo na camada interna. O foco do trabalho foi a resolução do problema de curto prazo e foi proposto um modelo de programação inteira mista (MIP, das iniciais em inglês de *Mixed Integer Programming*) para a programação de 29 ordens de manutenção a serem executadas em 11 ferramentas diferentes, em uma janela de tempo pré-determinada. O escalonamento de várias ordens de manutenção para a mesma ferramenta é equivalente à programação de várias ordens de manutenção de equipamento simultaneamente no problema em estudo no presente trabalho. O objetivo é maximizar a disponibilidade geral da ferramenta e minimizar a indisponibilidade nos períodos nos quais é esperada uma alta carga de trabalho para a ferramenta. Assim como no problema foco deste trabalho, a função objetivo não leva em consideração os custos da manutenção. Para a análise dos resultados foi feito um estudo de caso baseado em um modelo de simulação já utilizado e validado pela empresa. Os resultados foram, então, comparados com o planejamento atual da empresa. Das 11 ferramentas, 10 não tiveram um resultado estatisticamente diferente. Houve, então, uma ferramenta para a qual o modelo teve um resultado 13,9% melhor. Apesar de não ser estatisticamente diferente, houve uma melhora média observada de aproximadamente 1,6%, o que representou um ganho financeiro de U$39 milhões por ano.

Adhikary et al. (2016) propuseram um algoritmo genético multiobjetivo para o escalonamento da manutenção preventiva. Os objetivos envolvem a maximização da disponibilidade e a minimização do custo de manutenção de uma planta de operação contínua em série. Foi realizado um estudo de caso de um gerador de uma unidade de geração de energia de 210 MW. Os resultados apresentados pelos autores mostram que o algoritmo genético consegue melhorar a disponibilidade em 3% em conjunto com uma redução do custo de manutenção de 437 milhões de rúpias indianas (aproximadamente U$6 milhões).

Saraiva et al. (2011) propuseram um modelo de programação inteira mista (MIP) e uma abordagem baseada na meta-heurística *Simulated Annealing* (SA) para a programação de ordens de manutenção de geradores térmicos. O estudo de caso apresentado

envolve 29 geradores e 29 ordens de manutenção, sendo uma para cada gerador. Assim como neste trabalho, cada ordem de manutenção tem uma janela de tempo pré-determinada. O objetivo é minimizar as perdas financeiras relacionadas às paradas dos geradores. Foi proposto o SA com uma penalização para cada restrição que pudesse ser violada. Algumas restrições são consideradas fortes e não podem ser violadas. Foi realizado então dois estudos de caso. Nesses estudos o SA alcançou uma solução sem a violação de nenhuma restrição em um tempo de 15 minutos.

3.3 Conclusões Parciais

Neste Capítulo foi feita uma revisão de literatura de problemas de otimização de manutenção e de problemas relacionados. Nos trabalhos pesquisados não foi encontrado um problema que pudesse ser diretamente aplicado ao PPOMPLP. No entanto, esses trabalhos foram utilizados como base para a formulação matemática proposta, assim como para o desenvolvimento dos algoritmos meta-heurísticos.

Capítulo 4

Uma Formulação de Programação Matemática para o PPOMPLP

Como não foi encontrada nenhuma abordagem para a resolução do problema em estudo, foi proposta uma formulação matemática a fim de descrevê-lo formalmente. A formulação proposta é descrita a seguir.

4.1 Notação

O Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP) pode ser descrito como a seguir. Considere $\mathcal{T} = \{1, 2, ..., n\}$ o conjunto de n manutenções a serem realizadas por um conjunto $\mathcal{W} = \{1, 2, ..., m\}$ de m equipes de trabalho.

A cada manutenção $i \in \mathcal{T}$ é associado um conjunto $\mathcal{W}_i \subseteq \mathcal{W}$ de equipes de trabalho capazes de realizá-la. Também são associados a cada manutenção $i \in \mathcal{T}$; um tempo p_i necessário para executá-la; equipamento A_i no qual a manutenção deve ser prestada; e janela de tempo $[\, e_i, \, l_i \,]$, que especifica o intervalo de tempo em que a manutenção i pode ser realizada. A penalidade por não realizar a manutenção $i \in \mathcal{T}$ é dada por w_i. O valor da penalidade foi definido empiricamente como sendo o tempo de processamento p_i vezes um valor correspondente à prioridade da ordem de manutenção. Na empresa em estudo as manutenções são divididas em 4 níveis de prioridade. Para as manutenções mais prioritárias, o valor do fator multiplicador foi 40, seguido por 30, 20 e 10 para as menos prioritárias.

Cada equipe de trabalho $k \in \mathcal{W}$ tem disponibilidade de prestar h_k horas de mão de obra. O conjunto $\mathcal{T}_k \subseteq \mathcal{T}$ indica as manutenções que podem ser realizadas pela equipe de trabalho $k \in \mathcal{W}$, ou seja, $\mathcal{T}_k$ indica as manutenções que a equipe de trabalho k tem habilidade para executar.

O objetivo do problema é determinar o plano de execução das manutenções que minimiza o número de equipes de trabalho necessários às manutenções, respeitando-se todas as restrições do problema, definidas formalmente no modelo matemático apresentado a seguir.

4.2 Modelo

A partir das informações descritas anteriormente, foram definidos cinco conjuntos de variáveis de decisão com o objetivo de modelar matematicamente o problema. Estes conjuntos de variáveis são detalhados a seguir:

- $x_{ij}^k = 1$, se a manutenção i precede imediatamente a manutenção j executada pela equipe de trabalho k; 0, caso contrário.
- $y_{ik} = 1$, se a manutenção i será executada pela equipe de trabalho k; 0, caso contrário.
- $z_k = 1$, se a equipe de trabalho k é utilizada; 0, caso contrário.
- $c_{ik} \geq 0$, tempo de conclusão da manutenção i pela equipe de trabalho k.
- $r_{ij} = 1$, se a manutenção i precede imediatamente a manutenção j para o caso em que ambas as manutenções referem-se ao mesmo equipamento ($A_i = A_j$) e que ambas são executadas por equipes diferentes; 0, caso contrário.

Com essas variáveis é possível descrever a seguinte formulação de otimização linear inteira mista:

$$\min \sum_{k \in \mathcal{W}} z_k \; + \; \sum_{i \in \mathcal{T}_k} w_i \left(1 - \sum_{k \in \mathcal{W}_i} y_{ik}\right) \tag{4.1}$$

$$\sum_{k \in \mathcal{W}_i} y_{ik} \leq 1 \qquad \forall i \in \mathcal{T} \qquad (4.2)$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{j\}} x_{ij}^k = y_{jk} \qquad \forall j \in \mathcal{T},\, k \in \mathcal{W}_j \qquad (4.3)$$

$$\sum_{j \in \mathcal{T}_k} x_{0j}^k = z_k \qquad \forall k \in \mathcal{W} \qquad (4.4)$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{il}^k = \sum_{j \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{lj}^k \qquad \forall k \in \mathcal{W},\, l \in \mathcal{T}_k \qquad (4.5)$$

$$c_{0k} = 0 \qquad \forall k \in \mathcal{W} \qquad (4.6)$$

$$c_{jk} \geq c_{ik} + p_j - M'_{ij} \cdot (1 - x_{ij}^k) \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \cup \{0\},\, j \in \mathcal{T}_k \qquad (4.7)$$

$$c_{ik} \geq (e_i + p_i) \cdot y_{ik} \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \qquad (4.8)$$

$$c_{ik} \leq l_i \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \qquad (4.9)$$

$$c_{jk'} \leq c_{ik} - p_i + M''_{ij} \cdot r_{ij} \qquad \forall k \in \mathcal{W},\, k' \in \mathcal{W},\, i \in \mathcal{T}_k,\, j \in \mathcal{T}_{k'},\ | \ k \neq k', i < j, A_i = A_j \qquad (4.10)$$

$$c_{jk'} \geq c_{ik} + p_j - M'_{ij} \cdot (1 - r_{ij}) \qquad \forall k \in \mathcal{W},\, k' \in \mathcal{W},\, i \in \mathcal{T}_k,\, j \in \mathcal{T}_{k'},\ | \ k \neq k', i < j, A_i = A_j \qquad (4.11)$$

$$c_{ik} \leq h_k \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \qquad (4.12)$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \cup \{0\},\, j \in \mathcal{T}_k \cup \{0\} \qquad (4.13)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \cup \{0\} \qquad (4.14)$$

$$z_k \in \{0, 1\} \qquad \forall k \in \mathcal{W} \qquad (4.15)$$

$$c_{ik} \geq 0 \qquad \forall k \in \mathcal{W},\, i \in \mathcal{T}_k \qquad (4.16)$$

$$r_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{T},\, j \in \mathcal{T} \mid i < j, A_i = A_j \qquad (4.17)$$

A função objetivo (4.1) busca reduzir a quantidade de mão de obra necessária para realizar o maior número possível de manutenções. O balanceamento entre os dois termos da função objetivo é realizado com base em uma penalidade w_i por não execução de cada manutenção. O conjunto de restrições (4.2) garante que cada manutenção será executada

por no máximo uma equipe de trabalho. O conjunto de restrições (4.3) garante que toda manutenção executada terá uma mão de obra alocada. As restrições (4.4) garantem que uma determinada equipe de trabalho só será alocada se ele for utilizada para execução de alguma manutenção. As restrições (4.5) garantem o sequenciamento das ordens de manutenção executadas por cada equipe de trabalho. O conjunto de restrições (4.6) impõe que o tempo de conclusão de uma manutenção fictícia, criada para facilitar a modelagem, é nulo para todas as equipes de trabalho. As restrições (4.7), (4.10) e (4.11) garantem que a equipe de trabalho não pode executar mais de uma ordem de manutenção simultaneamente e cada equipamento não pode ter mais de uma ordem de manutenção sendo executada simultaneamente. Mais especificamente, as restrições (4.7) são ativadas apenas quando a manutenção j for realizada imediatamente após a manutenção i pela equipe k. Por sua vez, as restrições (4.10) são ativadas quando a manutenção j precede imediatamente a manutenção i para o caso em que ambas as manutenções referem-se ao mesmo equipamento ($A_i = A_j$) e executadas por equipes diferentes. As restrições (4.11) são similares às (4.10) quando a manutenção i precede imediatamente a manutenção j. Para ativar ou desativar essas restrições, as constantes M'_{ij} e M''_{ij} devem assumir valores maiores ou iguais à $l_i + p_j$ e $l_j + p_i$, respectivamente. As restrições (4.8) e (4.9) garantem que cada manutenção é executada dentro de sua janela de tempo, enquanto as restrições (4.12) garantem que o número de horas alocadas para uma equipe de trabalho seja menor ou igual ao número de horas disponíveis. Por fim, as restrições (4.13)-(4.17) indicam o domínio e escopo das variáveis de decisão.

4.3 Conclusões Parciais

Este Capítulo apresentou uma formulação de programação matemática para descrever formalmente o PPOMPLP. Essa formulação é útil para o entendimento do problema e para validar os algoritmos meta-heurísticos desenvolvidos.

Capítulo 5

Algoritmos Heurísticos para o PPOMPLP

A formulação MILP, desenvolvida no Capítulo anterior, foi muito importante para o entendimento do problema. Entretanto, dado o fato de o PPOMPLP ser da classe NP-difícil, também foram propostos cinco algoritmos meta-heurísticos baseados em *Simulated Annealing* (SA), *Variable Neighborhood Search* (VNS) e *Biased Random-Key Genetic Algorithm* (BRKGA). Essas meta-heurísticas têm sido utilizadas com sucesso na resolução de vários problemas combinatórios (Cordone and Lulli, 2016; Ferreira and de Queiroz, 2018; Zudio et al., 2018). Tais algoritmos são descritos neste Capítulo.

5.1 Representação da Solução

Em todos os cinco algoritmos propostos neste trabalho, uma solução s do problema é representada de forma indireta por uma permutação $s = \langle s_1, s_2, \cdots, s_n \rangle$ das n ordens de manutenção. A posição em que cada ordem de manutenção s_i aparece na permutação s define a prioridade da manutenção s_i a ser considerada no algoritmo de alocação, descrito na seção seguinte.

Optou-se por utilizar esta representação indireta, uma vez que dessa forma a geração de soluções iniciais, bem como a exploração do espaço de soluções viáveis por meio dos operadores de vizinhança, torna-se um processo muito simplificado, quando comparado à representação direta da solução, ou seja, uma lista de ordens de manutenção para cada equipe de trabalho.

5.2 Algoritmo de Alocação de Ordens de Manutenção

A partir de uma solução s do problema é aplicado um algoritmo simples, que busca alocar as ordens de manutenção às equipes. Este algoritmo serve, também, para avaliar o custo de uma solução. Os passos do referido processo de alocação são descritos em alto nível no Algoritmo 1.

Algoritmo 1: Alocação de Ordens de Manutenção

```
1  custo ← 0
2  para cada equipe k ← 1 até m faça
3  |  z_k ← false
4  fim
5  para cada ordem de manutenção s_i da posição i ← 1 até n faça
6  |  alocou ← false
7  |  para cada equipe k ← 1 até m faça
8  |  |  se s_i ∈ T_k então
9  |  |  |  Construa um controle com as janelas de tempo consolidadas (conforme
   |  |  |  Figuras 5.1 e 5.2)
10 |  |  |  para cada intervalo faça
11 |  |  |  |  se intervalo compatível então
12 |  |  |  |  |  Aloque a ordem de manutenção s_i à equipe k
13 |  |  |  |  |  Aloque a ordem de manutenção s_i em seu respectivo equipamento
14 |  |  |  |  |  alocou ← true
15 |  |  |  |  |  se z_k = false então
16 |  |  |  |  |  |  z_k ← true
17 |  |  |  |  |  |  custo ← custo + 1
18 |  |  |  |  |  fim
19 |  |  |  |  |  Vá para linha 5
20 |  |  |  |  fim
21 |  |  |  fim
22 |  |  fim
23 |  fim
24 |  se alocou = false então
25 |  |  custo ← custo + w_{s_i}
26 |  fim
27 fim
28 retorna custo
```

O Algoritmo 1 funciona como segue. Inicialmente (linhas 1 a 4), o custo e o controle de alocação das equipes são inicializados. Em seguida, as ordens de manutenção são percorridas sequencialmente, com o objetivo de tentar alocá-las (linha 5). A linha 6 inicializa um controle para verificação se a ordem de manutenção foi alocada. Para cada manutenção, as equipes de trabalho são percorridas sequencialmente (linha 7), da

primeira até a última, até encontrar uma equipe de trabalho capacitada para a execução da ordem de manutenção. Na linha 8 é verificada se a equipe é capacitada a executar a manutenção. Caso seja capacitada, na linha 9 é construída uma janela de tempo consolidada, assim chamada uma janela que leva em consideração tanto a disponibilidade da equipe quanto a do equipamento no qual a manutenção será executada. Os intervalos disponíveis na janela de tempo consolidada são percorridos sequencialmente (linha 10). Se a ordem de manutenção puder ser alocada dentro do intervalo disponível (linha 11), a ordem de manutenção é alocada tanto no controle das equipes (linha 12), quanto no controle do respectivo equipamento (linha 13) e o controle é atualizado para indicar que a manutenção foi alocada (linha 14). Se a equipe de trabalho ainda não foi alocada (linha 15), o controle de alocação da equipe é atualizado (linha 16) e o custo é acrescido em uma unidade (linha 17). A linha 19 sai dos laços de tentativa de alocação da ordem de manutenção se ela já tiver sido alocada. Se, ao fim de todas as tentativas de alocação, não for possível alocar a ordem de manutenção, a manutenção não será executada e uma penalidade por sua não execução (w_{s_i}) é acrescida ao custo da respectiva solução, conforme mostra a linha 25 do Algoritmo 1.

Note que, como o algoritmo de alocação trata todas as restrições do problema, qualquer solução s é uma solução viável para o problema. Essa é uma característica interessante para problemas com muitas restrições.

Para ilustrar o funcionamento do procedimento de alocação das ordens de manutenção deste Algoritmo, considere uma ordem de manutenção de duração igual a 2 unidades de tempo, que tenha uma janela de tempo de execução no intervalo [2, 9]. A Figura 5.1 representa na cor verde o controle das janelas de tempo da manutenção, assim como de uma dada equipe capacitada a executar essa ordem de manutenção e do equipamento em que ela será executada.

Para este exemplo, o procedimento verifica primeiramente que a equipe em teste possui uma disponibilidade de execução nos períodos [1, 2], [5, 6] e [8, 10]; o equipamento nos períodos [1, 4] e [6, 10]. A Figura 5.2 consolida todas essas disponibilidades de execução dessa ordem de manutenção, isto é, há disponibilidade nos períodos [2, 2], [6, 6] e [8, 9]. Como a manutenção tem uma duração de 2 unidades de tempo contínuas, não é possível alocá-la aos períodos [2, 2] e [6, 6]; neste caso, ela é alocada ao período [8, 9].

Ao final da aplicação do algoritmo, ter-se-á um conjunto de ordens de manutenção alocadas a um conjunto de equipes, bem como, possivelmente, um conjunto de ordens

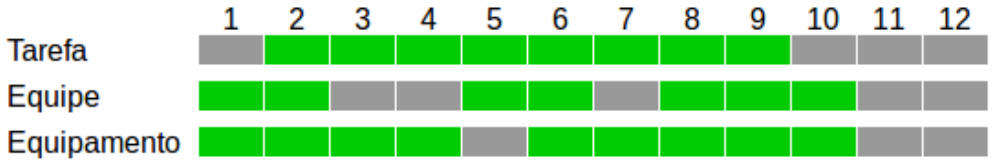

Figura 5.1: Controle das janelas de tempo.

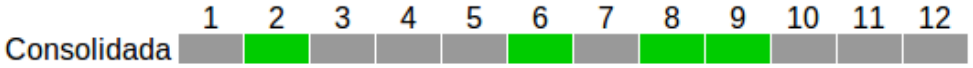

Figura 5.2: Consolidação dos controles das janelas de tempo

de manutenção não alocadas.

Para ilustrar o funcionamento do algoritmo completo, considere a instância representada pelas Tabelas 5.1 e 5.2 com 6 ordens de manutenção preventivas envolvendo 3 equipamentos. Observa-se na Tabela 5.1 que há 3 equipes diferentes com duas especialidades (mecânica e elétrica) que podem ser alocadas para a execução dessas ordens de manutenção. As ordens de manutenção 1, 2, 3 e 6 podem ser executadas pelas equipes de mecânica (1 ou 3) e as ordens de manutenção 4 e 5 pela equipe de elétrica (2). As ordens de manutenção 1, 3 e 5 devem ser executadas no carro, as ordens de manutenção 2 e 6 devem ser executadas no caminhão e a ordem de manutenção 3 deve ser executada na moto. As janelas de tempo de cada ordem de manutenção devem começar nos horários indicados pela coluna Início e terminar nos horários indicados pela coluna Fim. O tempo de execução previsto para cada manutenção é informado na coluna Duração e a penalidade por não execução na coluna Penalidade.

Tabela 5.1: Equipes de trabalho

Equipe	Especialidade
1) Mecânica A	Mecânica
2) Elétrica A	Elétrica
3) Mecânica B	Mecânica

A Figura 5.3 mostra o controle de alocações das ordens de manutenção, dos equipamentos e das equipes. Na cor verde é possível ver a janela de alocação de cada ordem de manutenção e na cor cinza os períodos que não podem ser alocados. A alocação

Tabela 5.2: Ordens de manutenção.

Tarefa de Manutenção	Equipamento	Especialidade	Início	Fim	Duração	Penalidade
1 (Alinhamento)	Carro	Mecânica	0	4	1	20
2 (Alinhamento)	Caminhão	Mecânica	2	7	2	30
3 (Revisão de motor)	Carro	Mecânica	3	9	3	40
4 (Revisão elétrica)	Moto	Elétrica	2	6	1	20
5 (Revisão elétrica)	Carro	Elétrica	3	8	2	30
6 (Revisão de motor)	Caminhão	Mecânica	4	7	1	40

para a solução $s = \langle 1, 2, 3, 4, 5, 6 \rangle$ pode ser visualizada na Figura 5.4. Nessa Figura, na cor vermelha está o período alocado e o número indica a ordem de manutenção que foi alocada. A seguir é explicado o passo a passo dessa alocação de acordo com o algoritmo proposto:

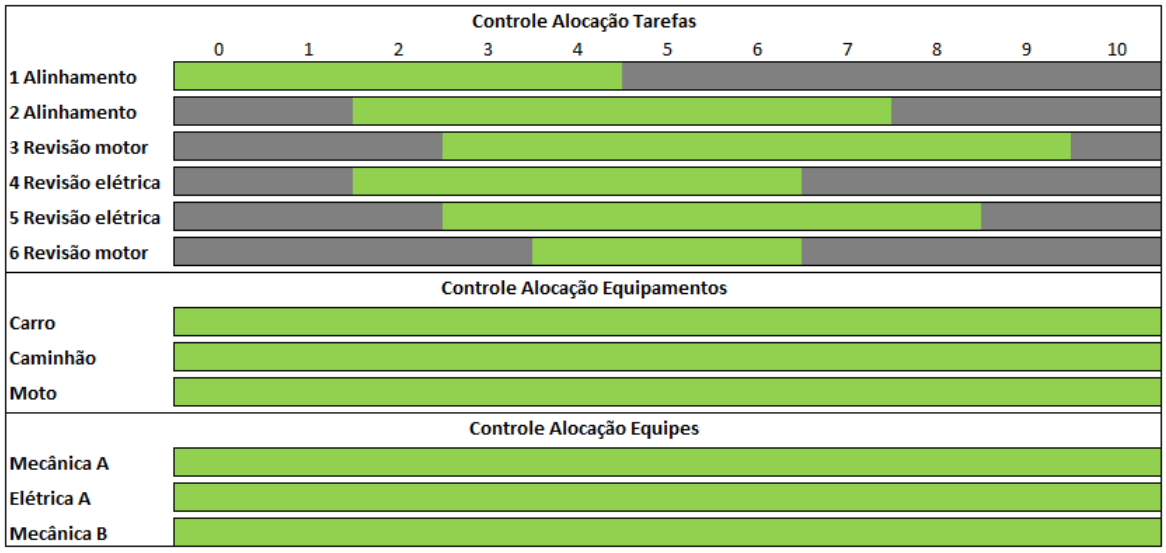

Figura 5.3: Controles de alocação

- Alocação da manutenção 1: A janela consolidada de alocação, considerando a primeira equipe compatível (Mecânica A), começa no período 0 e vai até o período 4. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 0 e tem duração de 1 unidade. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 1.

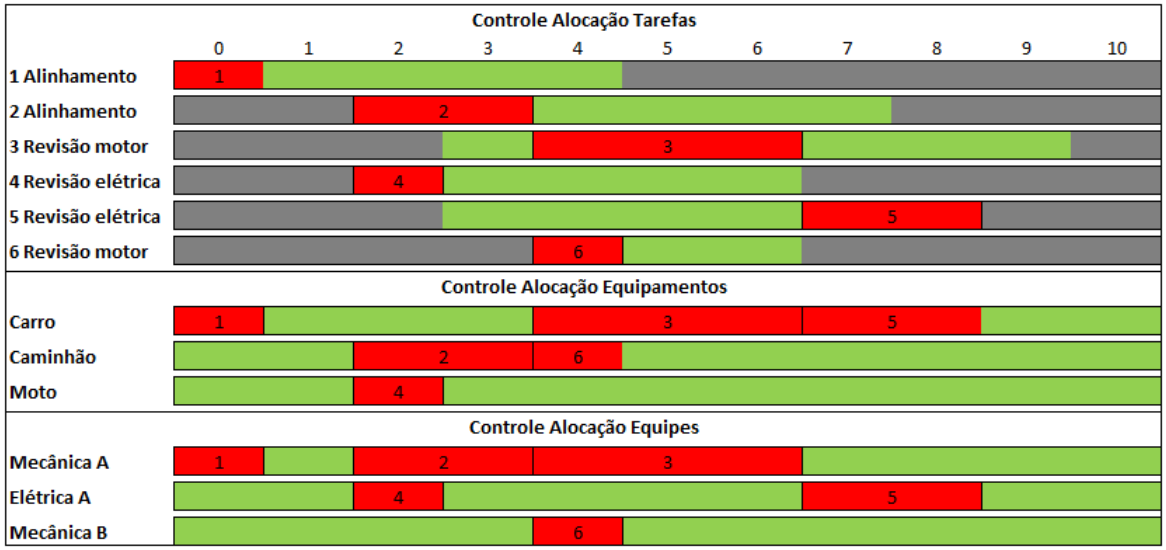

Figura 5.4: Alocação inicial

- Alocação da manutenção 2: A janela consolidada de alocação, considerando a primeira equipe compatível (Mecânica A), começa no período 2 e vai até o período 7. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 2 e tem duração de 2 unidades. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 2.

- Alocação da manutenção 3: A janela consolidada de alocação, considerando a primeira equipe compatível (Mecânica A), começa no período 4 e vai até o período 9. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 4 e tem duração de 3 unidades. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 3.

- Alocação da manutenção 4: A janela consolidada de alocação, considerando a primeira equipe compatível (Elétrica A), começa no período 2 e vai até o período 6. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 2 e tem duração de 1 unidade. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 4.

- Alocação da manutenção 5: A janela consolidada de alocação, considerando a primeira equipe compatível (Elétrica A), começa no período 7 e vai até o período 8. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 7 e tem duração de 2 unidades. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 5.

- Alocação da manutenção 6: Não existe uma janela consolidada de alocação considerando a primeira equipe compatível (Mecânica A). Considerando a segunda equipe compatível (Mecânica B), a janela consolidada de alocação começa no período 4 e vai até o período 6. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 4 e tem duração de 1 unidade. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 6.

Com a aplicação desse procedimento de alocação de ordens de manutenção, conclui-se que a solução $s = \langle 1, 2, 3, 4, 5, 6 \rangle$ do exemplo considerado tem um custo igual a 3.

5.3 Solução Inicial e Estrutura de Vizinhança

Antes de apresentar os algoritmos heurísticos propostos é necessário fazer algumas definições que são comuns aos algoritmos SA, VNS e MSVNS, baseados em busca local. Todos esses algoritmos são iniciados com uma solução criada de forma completamente aleatória, escolhendo-se uma permutação s qualquer das n ordens de manutenção.

Para explorar o espaço de soluções é definida uma estrutura de vizinhança simples, que consiste na troca de duas posições da permutação s. Todos os vizinhos de uma solução s são representados por $\mathcal{N}(s)$. É importante observar que, com esse movimento, é possível explorar todo o espaço de soluções do problema partindo-se de qualquer solução inicial.

A Figura 5.5 mostra uma solução s e um de seus vizinhos s' para uma instância com 6 ordens de manutenção. Nessa solução, a ordem de manutenção 6 é trocada com a 3 na sequência de execução.

$$s = \left\langle\ 1,\ 2,\ \mathbf{3},\ 4,\ 5,\ \mathbf{6}\ \right\rangle \qquad s' = \left\langle\ 1,\ 2,\ \mathbf{6},\ 4,\ 5,\ \mathbf{3}\ \right\rangle$$

Figura 5.5: Uma solução s e uma solução vizinha s'.

A Figura 5.6 mostra a alocação da solução s'. Seguindo o algoritmo proposto para a alocação das ordens de manutenção, a alocação das manutenções 1 e 2 é a mesma. A seguir é explicado, passo a passo, como são feitas as alocações subsequentes, isto é, as alocações a partir da alocação da manutenção 6 (terceira a ser alocada no vetor s'):

- Alocação da manutenção 6: a janela consolidada de alocação, considerando a primeira equipe compatível (Mecânica A), começa no período 4 e vai até o período 6. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 4 e tem duração de 1 unidade. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 6.
- Alocação da manutenção 4: a janela consolidada de alocação, considerando a primeira equipe compatível (Elétrica A), começa no período 2 e vai até o período 6. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 2 e tem duração de 1 unidade. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 4.

- Alocação da manutenção 5: a janela consolidada de alocação, considerando a primeira equipe compatível (Elétrica A), começa no período 3 e vai até o período 8. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 3 e tem duração de 2 unidades. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 5.

- Alocação da manutenção 3: a janela consolidada de alocação, considerando a primeira equipe compatível (Mecânica A), começa no período 5 e vai até o período 9. Como a janela tem uma duração suficiente para alocação da manutenção, ela é alocada ao período 5 e tem duração de 3 unidades. Dessa forma, os 3 controles de alocação são atualizados em vermelho com a manutenção 3.

Com a alocação feita pelo procedimento acima, conclui-se que a solução s' tem custo 2. No caso, ela é ótima, visto que não houve penalidade alguma e que eram necessárias duas equipes com capacitações diferentes para realizar as 6 manutenções.

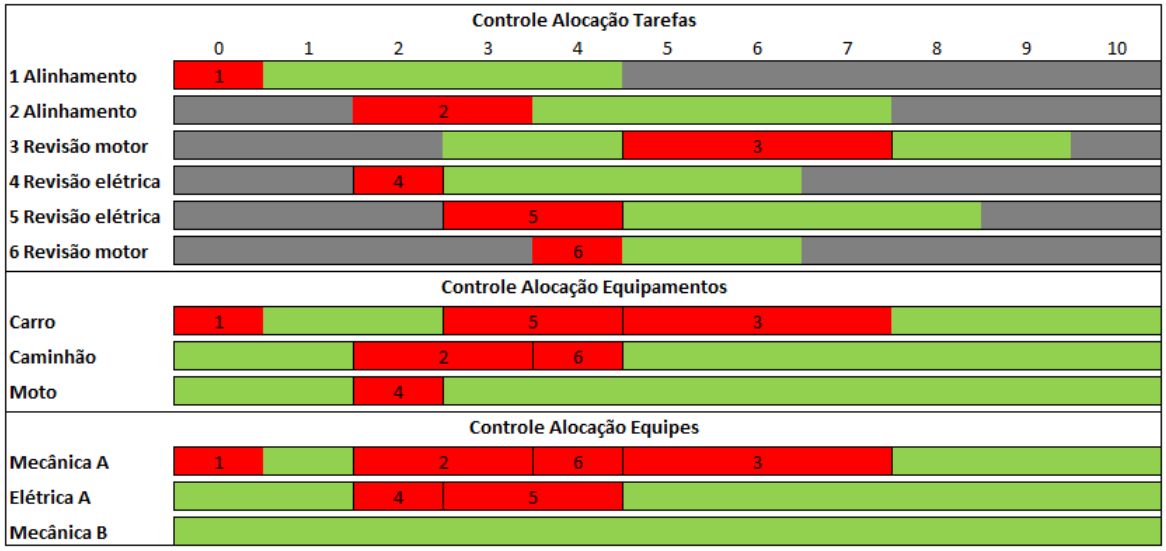

Figura 5.6: Alocação depois da troca das ordens de manutenção 3 e 6.

5.4 *Variable Neighborhood Search*

Variable Neighborhood Search – VNS (Hansen and Mladenović, 2014; Mladenović and Hansen, 1997) é um método que explora o espaço de soluções de um problema de otimização por meio de buscas locais e de mudanças sistemáticas de estruturas de vizinhança. Essas mudanças de estruturas têm o objetivo de capacitar o algoritmo a não ficar preso em ótimos locais.

O procedimento SHAKE, descrito pelo Algoritmo 2, é responsável por gerar as perturbações nos ótimos locais correntes e, assim, permitir que o algoritmo não fique preso nesses ótimos locais. A partir de uma solução s o procedimento faz k perturbações (*swaps*) sucessivas, em que k define o nível da perturbação.

O procedimento BUSCA-LOCAL, descrito pelo Algoritmo 3, é responsável pelo refinamento da solução perturbada. Esse procedimento analisa os vizinhos da solução s (considerando a vizinhança de permutação) e escolhe o melhor vizinho s". Se esse vizinho for melhor que a melhor solução corrente s', então a solução é aceita. O procedimento é repetido até que não exista mais um vizinho capaz de melhorar a solução corrente, situação que configura a existência de um ótimo local com relação à vizinhança de troca.

O Algoritmo 4 mostra o pseudocódigo do VNS básico.

Algoritmo 2: SHAKE(s, k)

```
1 s' ← s
2 i ← 1
3 enquanto i ≤ k faça
4 |   Selecione aleatoriamente um vizinho s'' ∈ N(s')
5 |   s' ← s''
6 |   i ← i + 1
7 fim
8 retorna s'
```

A solução inicial (linha 1 do Algoritmo 4) é gerada aleatoriamente e é considerada como a melhor solução até então. Enquanto a estrutura de vizinhança corrente não atingir o número máximo de estruturas a serem analisadas (linha 11), o algoritmo progressivamente gera soluções intermediárias s' pela introdução de perturbações na solução corrente s (linha 4). Para cada solução perturbada s' é aplicada uma busca local (linha 5). Se uma solução melhor é encontrada (linha 6), ela é aceita e a estrutura de vizinhança volta para a primeira (respectivamente, linhas 7 e 8). Caso contrário, a estrutura de vizinhança é incrementada em uma unidade (linha 10). Quando o laço de

repetição é interrompido pelo número máximo de estruturas de vizinhança, a melhor solução encontrada durante a busca é retornada (linha 13 do Algoritmo 4).

Algoritmo 3: BUSCA-LOCAL(s)

```
1  s' ← s
2  repita
3  |  Encontre o melhor vizinho s'' ∈ N(s')
4  |  se f(s'') < f(s') então
5  |  |  s' ← s''
6  |  |  otimo_local ← false
7  |  senão
8  |  |  otimo_local ← true
9  |  fim
10 até otimo_local = true ;
11 retorna s'
```

Algoritmo 4: VNS

```
1  s ← Solução inicial aleatória
2  k ← 1
3  repita
4  |  s' ← SHAKE(s, k) // Algoritmo 2
5  |  s'' ← BUSCA-LOCAL(s') // Algoritmo 3
6  |  se f(s'') < f(s) então
7  |  |  s ← s''
8  |  |  k ← 1
9  |  senão
10 |  |  k ← k + 1
11 |  fim
12 até k > k_max ;
13 retorna s
```

5.5 *Multi-Start Variable Neighborhood Search*

Os algoritmos *Multi-Start* são iniciados em diversos pontos diferentes do espaço de busca e retornam o melhor valor das soluções refinadas (Martí et al., 2013). O *Multi-Start Variable Neighborhood Search* (MSVNS) é praticamente igual ao algoritmo VNS explicado na Seção anterior, porém iniciado diversas vezes com uma solução aleatória. O Algoritmo 5 mostra seu funcionamento, considerando como critério de parada um tempo máximo prefixado de processamento, dado por $tempo_{\max}$.

Algoritmo 5: MSVNS

```
1  f(s*) ← ∞
2  enquanto tempo < tempo_max faça
3  |  s ← Solução inicial aleatória
4  |  k ← 1
5  |  repita
6  |  |  s' ← SHAKE(s, k) // Algoritmo 2
7  |  |  s'' ← BUSCA-LOCAL(s') // Algoritmo 3
8  |  |  se f(s'') < f(s) então
9  |  |  |  s ← s''
10 |  |  |  k ← 1
11 |  |  senão
12 |  |  |  k ← k + 1
13 |  |  fim
14 |  até k > k_max ;
15 |  se f(s) < f(s*) então
16 |  |  s* ← s
17 |  fim
18 fim
19 retorna s*
```

5.6 *Simulated Annealing*

Simulated Annealing – SA (Kirkpatrick et al., 1983) é um algoritmo meta-heurístico probabilístico inspirado no processo metalúrgico termodinâmico de recozimento de metais.

No processo metalúrgico, o metal é aquecido a uma temperatura suficiente para que ele chegue próximo à temperatura de fusão. Nesta temperatura, o metal irá ter uma alta ductilidade, permitindo que a estrutura física do metal se altere facilmente. À medida que o metal vai se resfriando, vai ficando mais rígido, sendo então cada vez mais difícil alterar a sua estrutura física. Quando o metal chega na temperatura final, a alteração física não é mais conseguida.

O SA simula a dinâmica deste processo metalúrgico, fazendo-se uma analogia entre o metal e a solução do problema em questão. No SA, quanto maior a temperatura, maior é a probabilidade de o algoritmo aceitar uma solução de piora.

Na simulação computacional, o sistema começa, então, com uma temperatura alta e, baseado em um fator de resfriamento, vai diminuindo gradativamente até uma temperatura baixa pré-determinada. Para cada temperatura, é buscado um vizinho qualquer por um número de máximo de iterações pré-determinado. Em cada iteração, se a solução vizinha for melhor, ela é aceita; se for pior, ela poderá ser aceita de acordo com uma função de probabilidade, que é diretamente proporcional à temperatura. Assim, quanto maior a temperatura, maior a chance de aceitação de uma solução de piora e quanto menor a temperatura, menor também é a chance de se aceitar uma solução de piora.

O Algoritmo 6 mostra o pseudocódigo do SA para a resolução do problema de alocação de ordens de manutenção, explicado a seguir.

A solução inicial é gerada aleatoriamente e é considerada a melhor solução até então (linha 2 do Algoritmo 6). Enquanto a temperatura mínima não é atingida e nem o número máximo de iterações é atingido, o algoritmo seleciona aleatoriamente um vizinho da solução corrente (linha 7). Essa solução pode ser aceita de acordo com a função probabilística implementada (linha 14). Essa solução é aceita como a melhor solução se ela for uma solução melhor que a melhor solução encontrada até então. Depois de *maxIterations* iterações, a temperatura é diminuída por um fator de resfriamento $\alpha \in [0, 1]$ (linha 18). O algoritmo termina quando a temperatura mínima é atingida, situação em que se retorna a melhor solução encontrada durante a busca.

Algoritmo 6: Simulated Annealing (SA)

```
1  s ← Solução inicial aleatória
2  melhorSolucao ← s
3  t ← maxTemperature
4  enquanto t > minTemperature faça
5  |  iter ← 0
6  |  enquanto iter < maxIterations faça
7  |  |  s' ← Selecione um vizinho aleatório s' ∈ N(s)
8  |  |  Δ ← f(s') − f(s)
9  |  |  se Δ < 0 então
10 |  |  |  s ← s'
11 |  |  |  se f(s') < f(melhorSolucao) então
12 |  |  |  |  melhorSolucao ← s'
13 |  |  |  fim
14 |  |  senão se rand(0,1) < e^{−Δ/t} então
15 |  |  |  s ← s'
16 |  |  iter ← iter + 1
17 |  fim
18 |  t ← t × (1 − α)
19 fim
20 retorna melhorSolucao
```

5.7 *Biased Random-Key Genetic Algorithm*

Biased Random-Key Genetic Algorithm – BRKGA (Martinez et al., 2011) é uma variante do *Random-Key Genetic Algorithm* – RKGA (Bean, 1994) e tem sido alvo de estudos na literatura desde 2004. Ambos são algoritmos genéticos (AGs).

Os algoritmos genéticos (Holland, 1975) fazem uma busca populacional baseados nos processos biológicos de evolução. Nesses algoritmos são implementados operadores de evolução, como cruzamento e mutação, analogamente ao que acontece nos processos naturais de evolução. Neste paralelo, cada solução é considerada um indivíduo e, portanto, é codificada de forma similar a um cromossomo (conjunto de genes) de forma a permitir a aplicação de algoritmos que simulam o processo biológico.

Após a aplicação dos operadores, é feita uma seleção de quais indivíduos devem sobreviver, baseados em uma probabilidade de sobrevivência na qual os indivíduos melhores têm maiores chances de sobreviver e uma nova geração é criada, com características da geração anterior. Espera-se a reprodução de filhos cada vez melhores ao longo das gerações.

Bean (1994) introduziu os algoritmos genéticos de chaves aleatórias ou *Random-Key Genetic Algorithm* – RKGA para problemas de sequenciamento. No RKGA a representação de cada indivíduo é feita por um vetor de chaves aleatórias no intervalo contínuo [0,1). Devido a essa representação genérica, faz-se necessário aplicar um algoritmo de decodificação, a fim de traduzir um vetor de chaves aleatórias para uma solução propriamente dita.

$$s' = \langle\ 0.12,\ 0.22,\ 0.99,\ 0.68,\ 0.88,\ 0.33\ \rangle \quad s = \langle\ 1,\ 2,\ 6,\ 4,\ 5,\ 3\ \rangle$$

Figura 5.7: Uma solução codificada s' e sua representação decodificada s.

A partir desse princípio de decodificação proposto, uma população inicial é facilmente gerada, simplesmente criando vetores de números aleatórios no intervalo contínuo [0,1). Para a reprodução, é realizado um *crossover* uniforme no qual dois pais são aleatoriamente selecionados e para cada gene do filho é feito um sorteio de qual pai o gene será herdado. A Figura 5.8 mostra um exemplo de reprodução nos quais os genes 1, 2 e 5 foram herdados do pai P1 e os genes 3, 4 e 6 foram herdados do pai P2. A Figura 5.9 mostra essas mesmas soluções de forma decodificada.

$$\begin{aligned} P1 &= \langle\ \mathbf{0.12},\ \mathbf{0.22},\ 0.99,\ 0.68,\ \mathbf{0.88},\ 0.33\ \rangle \\ P2 &= \langle\ 0.52,\ 0.62,\ \mathbf{0.19},\ \mathbf{0.18},\ 0.22,\ \mathbf{0.01}\ \rangle \\ F &= \langle\ \mathbf{0.12,\ 0.22,\ 0.19,\ 0.18,\ 0.88,\ 0.01}\ \rangle \end{aligned}$$

Figura 5.8: Exemplo de reprodução.

$$\begin{aligned} P1 &= \langle\ 1,\ 2,\ 6,\ 4,\ 5,\ 3\ \rangle \\ P2 &= \langle\ 6,\ 4,\ 3,\ 5,\ 1,\ 2\ \rangle \\ F &= \langle\ 6,\ 1,\ 4,\ 3,\ 2,\ 5\ \rangle \end{aligned}$$

Figura 5.9: Exemplo de reprodução decodificada.

No RKGA o conceito original de mutação dos GAs não é utilizado. Ao invés de mutação é utilizado o conceito de mutantes. Os mutantes são novos indivíduos inseridos

na população utilizando o mesmo processo de geração da população original, ou seja, são indivíduos que são criados codificados por um vetor de chaves aleatórias no intervalo contínuo [0,1).

A Figura 5.10 exemplifica o processo de evolução. Para realizar a evolução no RKGA são computados os custos de todas as soluções da geração atual e selecionado um grupo de elite com as melhores soluções e adicionados à próxima geração. Um outro grupo de mutantes é gerado e adicionado à próxima geração. Até que a população da próxima geração seja completada, é realizada a reprodução entre dois indivíduos aleatórios da geração atual e o filho é adicionado à nova geração.

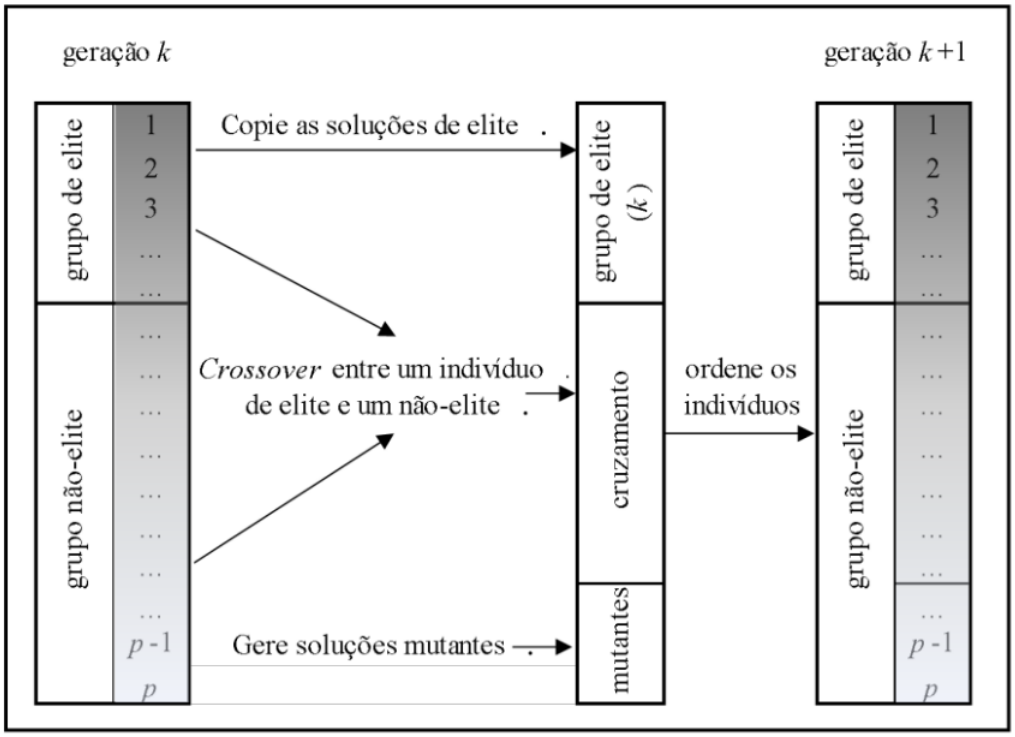

Figura 5.10: Evolução do BRKGA (Mainieri, 2014)

Os algoritmos genéticos com chaves aleatórias tendenciosas ou *Biased Random-Key Genetic Algorithm* – BRKGA diferem do RKGA na forma em que os pais são selecionados para a reprodução e como a reprodução acontece (Gonçalves and Resende, 2011). No RKGA, todos os indivíduos da população têm a mesma probabilidade de serem escolhidos e a reprodução ocorre conforme explicado anteriormente. Já no BRKGA, um dos indivíduos é escolhido aleatoriamente sempre a partir do conjunto elite (conjunto formado com as melhores indivíduos) e o outro é escolhido aleatoriamente do conjunto não elite.

Na reprodução, para cada gene do filho a ser gerado é feito um sorteio tendencioso de qual pai ele herdará esse gene. Um parâmetro de probabilidade é então necessário para definir o quão tendencioso será a escolha do grupo de elite. A Figura 5.11 mostra um exemplo de reprodução tendenciosa na qual a probabilidade de escolha do gene do pai proveniente do grupo de elite (P1) é 0.7. Observa-se nessa Figura que, quando o número aleatório sorteado é menor que 0.7, o gene é herdado do P1 e quando é maior que 0.7, o gene é herdado do pai P2.

P1	0.12	0.22	0.99	0.68	0.88	0.33
P2	0.52	0.62	0.19	0.18	0.22	0.01
Número aleatório	0.14	0.42	0.73	0.34	0.54	0.84
Parâmetro de probabilidade = 0.7	$<$	$<$	$>$	$<$	$<$	$>$
Herda de	P1	P1	P2	P1	P1	P2
F	0.12	0.22	0.19	0.68	0.88	0.01

Figura 5.11: Exemplo de reprodução tendenciosa.

Neste trabalho foi definida uma forma simples de decodificação, que é o ordenamento não decrescente do vetor de chaves aleatórias contínuas na qual a ordem de manutenção é representada pela posição relativa da chave no vetor. A Figura 5.7 mostra um indivíduo s' e sua decodificação s para a instância com 6 ordens de manutenção. No exemplo, a ordem de manutenção 3, representada pela chave 0.99 seria a última ordem a ser executada, como pode ser observado na solução s.

O Algoritmo 7 mostra o pseudocódigo do BRKGA, baseado no código apresentado por Faria et al. (2017). Na linha 1, a função é inicializada com um valor muito alto. O algoritmo segue gerando uma população de n chaves aleatórias, sendo n o número de manutenções (linha 2). Enquanto o critério de parada não for satisfeito (no caso, o limite de tempo do teste), o algoritmo continua evoluindo de acordo com os passos a seguir. Cada nova solução da população é decodificada e avaliada (linha 4). A população é dividida em um conjunto elite e outro não elite (linha 5). A melhor solução é buscada (linha 6) e o seu índice armazenado (linha 7). Se o valor da solução encontrada é menor que o da melhor solução (linha 8), tanto o índice quanto a melhor solução são atualizados com os valores encontrados (linhas 9 e 10). Na linha 12 a próxima geração é inicializada com todos os indivíduos da população elite. Na linha 13 são gerados os conjuntos de população mutantes e adicionados à próxima geração (linha 14). A população é, então, completada até o número máximo de indivíduos de acordo com os passos a seguir.

Um pai a é selecionado do grupo de elite (linha 16) e outro pai b do grupo não elite (linha 16). É, então, realizado um sorteio tendencioso, com a probabilidade maior de escolher o pai do grupo de elite (linha 18). O filho escolhido é, então, adicionado ao conjunto dos filhos (linhas 20 e 22). Após a escolha dos filhos, eles são adicionados à próxima população (linha 25) e a população é atualizada (linha 27). Quando o critério de parada é satisfeito, a melhor solução é retornada (linha 29). A Figura 5.12 mostra um fluxograma do funcionamento do algoritmo.

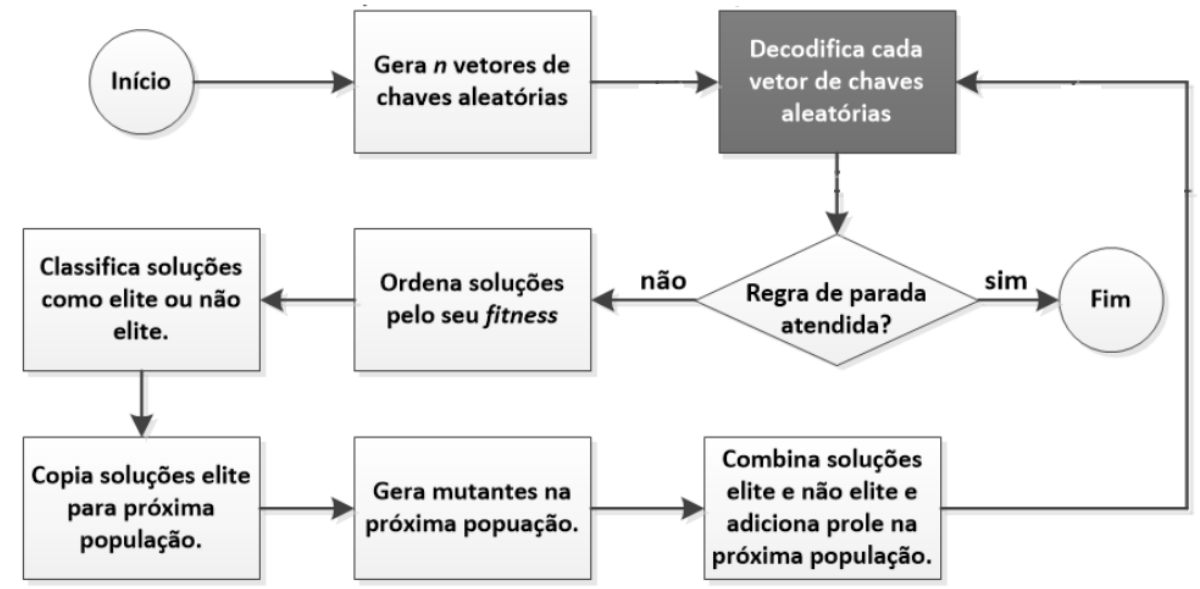

Figura 5.12: Fluxograma do BRKGA. Adaptado de Frinhani et al. (2015)

5.8 *Biased Random-Key Memetic Algorithm*

De acordo com Neri and Cotta (2012), os algoritmos meméticos (*Memetic Algorithms* – MA) foram, inicialmente, uma modificação dos algoritmos genéticos (*Genetic Algorithms* – GA) desenvolvidos para resolver o Problema do Caixeiro Viajante (*Traveling Salesman Problem* – TSP). A modificação foi a inclusão de um operador de busca local nos pais antes da operação de cruzamento.

O *Biased Random-Key Memetic Algorithm* (BRKMA) aplica uma ideia semelhante ao do algoritmo BRKGA, exceto pelo fato de que a cada L gerações é aplicada a busca local (Algoritmo 3) nas soluções decodificadas relativas aos indivíduos elite, ou seja,

Algoritmo 7: Biased Random-Key Genetic Algorithm (BRKGA)

```
1  f* ← ∞
2  Gere uma população P com n vetores de chaves aleatórias
3  enquanto critério de parada não satisfeito faça
4  |  Decodifique e avalie o custo de cada nova solução em P
5  |  Divida P em dois conjuntos: P_e e P_ē
6  |  Encontre a melhor solução s+ de P_e
7  |  s+ ← argmin{f(s) | s ∈ P_e}
8  |  se f(s+) < f* então
9  |  |  s* ← s+
10 |  |  f* ← f(s*)
11 |  fim
12 |  Inicialize a população da próxima geração: P+ ← P_e
13 |  Gere um conjunto de |P_m| mutantes, cada mutante com n chaves aleatórias
14 |  Adicione P_m à próxima geração: P+ ← P+ ∪ P_m
15 |  para cada i ← 1 a |P| − |P_e| − |P_m| faça
16 |  |  Selecione aleatoriamente um Pai a de P_e  Selecione aleatoriamente um Pai
   |  |  b de P_ē
17 |  |  para cada j ← 1 a n faça
18 |  |  |  Jogue uma moeda viciada com probabilidade ρ_α > 0,5 de sair cara
19 |  |  |  se cara então
20 |  |  |  |  c[j] ← a[j]
21 |  |  |  senão
22 |  |  |  |  c[j] ← b[j]
23 |  |  |  fim
24 |  |  fim
25 |  |  Adicione o filho c à população da próxima geração: P+ ← P+ ∪ {c}
26 |  fim
27 |  Atualize a população: P ← P+
28 fim
29 retorna s*
```

somente naqueles pertencentes à P_e. O objetivo da busca local é encontrar os ótimos locais da população elite, com isso, espera-se que o algoritmo encontre soluções que podem ser mais promissoras.

O Algoritmo 8 mostra o pseudocódigo do BRKMA, adaptado do código apresentado por Faria et al. (2017). Este algoritmo difere do BRKGA (apresentado pelo Algoritmo 29) apenas pela adição da busca local (Algoritmo 3) aos elementos do grupo elite (linha 13) a cada L gerações (linha 12).

Algoritmo 8: Biased Random-Key Memetic Algorithm (BRKMA)

$f^* \leftarrow \infty$
Gere uma população P com n vetores de chaves aleatórias
enquanto *critério de parada não satisfeito* **faça**
    Decodifique e avalie o custo de cada nova solução em P
    Divida P em dois conjuntos: P_e e $P_{\overline{e}}$
    Encontre a melhor solução s^+ de P_e
    $s^+ \leftarrow argmin\{f(s) \,|\, s \in P_e\}$
    se $f(s^+) < f^*$ **então**
        $s^* \leftarrow s^+$
        $f^* \leftarrow f(s^*)$
    fim
    se *Número de gerações é múltiplo de* L **então**
        $P_e \leftarrow \{\text{Busca-Local}(e)\ \forall e \in P_e\}$
    fim
    Inicialize a população da próxima geração: $P^+ \leftarrow P_e$
    Gere um conjunto de P_m mutantes, cada mutante com n chaves aleatórias
    Adicione P_m na próxima geração: $P^+ \leftarrow P^+ \cup P_m$
    para *cada* $i \leftarrow 1$ a $|P| - |P_e| - |P_m|$ **faça**
        Selecione aleatoriamente um Pai a de P_e
        Selecione aleatoriamente um Pai b de $P_{\overline{e}}$
        para *cada* $j \leftarrow 1$ a n **faça**
            Jogue uma moeda viciada com probabilidade $\rho_\alpha > 0{,}5$ de sair *cara*
            se *cara* **então**
                $c[j] \leftarrow a[j]$
            senão
                $c[j] \leftarrow b[j]$
            fim
        fim
        Adicione o filho c à população da próxima geração: $P^+ \leftarrow P^+ \cup \{c\}$
    fim
    Atualize a população: $P \leftarrow P^+$
fim
retorna s^*

5.9 *Conclusões Parciais*

Este Capítulo apresentou cinco diferentes algoritmos meta-heurísticos como alternativas para solução do problema em estudo. Esses algoritmos são baseados nas meta-heurísticas *Simulated Annealing* (SA), *Variable Neighborhood Search* (VNS), *Multi-Start* (MS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Biased Random-Key Memetic Algo-*

rithm (BRKMA). Inicialmente, mostrou-se como representar uma solução, como avaliá-la por meio de um algoritmo de alocação das ordens de manutenção e como explorar o espaço de soluções por meio de uma estrutura de vizinhança baseada em troca da ordem de realização das manutenções. Em seguida, cada algoritmo proposto foi detalhado e, considerando a simplicidade da representação da solução, os algoritmos foram implementados tal como propostos na literatura, sem a necessidade de adaptações.

Capítulo 6

Experimentos e Resultados Computacionais

A formulação MILP proposta na Seção 4.2, à página 20, foi programada na linguagem C++ utilizando o Concert Technology Library do CPLEX, versão 12.5 acadêmica, na configuração padrão, com exceção do tempo de execução, que foi limitado a uma hora de processamento. Os algoritmos SA, VNS, MSVNS, BRKGA e BRKMA foram programados na linguagem C++.

Os experimentos foram divididos em dois conjuntos de testes. O primeiro é relativo às três primeiras fases propostas, mencionadas na Seção 1.4, à página 5. Neste primeiro conjunto são analisados os resultados do MILP, do SA e do VNS. O segundo conjunto é relativo à quarta fase proposta, também mencionada na Seção 1.4. Neste último conjunto foram analisados os resultados do MSVNS, do BRKGA e do BRKMA, visto que as instâncias desse conjunto têm dimensões maiores e, por isso, foram propostos algoritmos com processamento paralelo para tratá-las.

Para garantir uma análise coerente dos dados com o objetivo de determinar qual algoritmo apresentou melhores resultados, foi aplicado um teste estatístico. Como os resultados não seguem uma distribuição normal, foi escolhido o teste de Friedman (Lowry, 2018), considerando um nível de confiança de 95%. Desta forma, o valor-p dado pelo teste tem que ser menor que 0,05 para confirmar que há diferença estatística entre os algoritmos. Se houver tal diferença, é necessário testar se todos os pares de algoritmos são estatisticamente diferentes. Esses testes são chamados de *post-hoc* (Pohlert, 2014). Os testes foram realizados utilizando a calculadora estatística, que já faz os testes *post-hoc*:

Friedman rank sum (omnibus) test calculator.[1]

6.1 Descrição das Instâncias

A equipe de engenharia de manutenção da indústria estudada forneceu os dados referentes a todas as ordens de manutenção do ano de 2016. Esses dados contêm informações de 33484 ordens de manutenção preventivas, envolvendo 1032 equipamentos e 145 equipes de trabalho.

Nos testes iniciais foi verificado que a formulação MILP só era capaz de resolver instâncias com um número muito pequeno de ordens de manutenção. Por esse motivo, foi gerado um conjunto de 100 instâncias menores e diferentes, que são decomposições da instância original. Esse conjunto foi gerado variando-se o número de ordens de manutenção preventivas (20 a 80), número de equipamentos (2 a 5) e o número de equipes de trabalho (3 a 14).

Esse primeiro conjunto de instâncias foi utilizado para a validação do MILP, assim como para comparar os resultados produzidos pelo MILP, VNS e o SA, descritos na Seção 6.2.

Para a quarta fase do trabalho foi necessário adicionar instâncias de dimensões maiores para verificar o desempenho dos algoritmos para instâncias de interesse prático. Para isso, foram adicionados dados reais de cinco outras unidades operacionais da empresa em estudo, totalizando, então 6 instâncias com dados reais. A partir dos dados de cada uma dessas unidades operacionais foram geradas outras 6 instâncias de dimensões menores (com 150, 300, 600, 1200, 2400 e 4800 ordens de manutenção), totalizando 36 instâncias. Para a primeira das instâncias reais disponibilizadas pela equipe de engenharia de manutenção, ainda foi gerada uma instância com 9600 e outra com 19200 ordens de manutenção, totalizando assim 38 instâncias. Para a geração das instâncias derivadas das 6 instâncias com dados reais, foram selecionados subconjuntos de manutenções relativos a períodos iguais. Como exemplo, suponha um mapa de 52 semanas com 5200 manutenções e cada semana com 100 manutenções. Para esse exemplo, uma instância com 600 manutenções considera todas as manutenções relativas a 6 semanas consecutivas. Somando as 38 novas instâncias com a instância real de 33484 ordens de manutenção e com as 100 instâncias anteriores, totalizam-se 139 instâncias para análise

[1] Disponível em `http://astatsa.com/FriedmanTest/`

na Seção 6.3.

6.2 Comparação entre MILP, VNS e SA

Todos os testes foram realizados em um computador Intel Xeon CPU E3-1225 v5 @ 3.30GHz $\times$ 4, 32GB RAM, sob sistema operacional Ubuntu 16.04.3 LTS 64 bits. Cada algoritmo foi aplicado 10 vezes em cada instância.

O algoritmo VNS (Algoritmo 4) tem apenas um parâmetro ($k_{\max}$), que foi determinado empiricamente em n, isto é, o número de ordens de manutenção preventivas.

Por sua vez, o algoritmo SA, dado pelo Algoritmo 6, tem quatro parâmetros:

- *maxTemperature*: temperatura máxima
- *minTemperature*: temperatura mínima
- *maxIterations*: número máximo de iterações
- α: fator de resfriamento

Para fazer uma calibração justa desses parâmetros foi utilizada uma ferramenta automatizada, de nome Irace (*Iterated Racing for Automatic Algorithm Configuration*), desenvolvida por López-Ibáñez et al. (2016). Esse algoritmo foi desenvolvido para retornar o conjunto mais apropriado de parâmetros dentro de uma faixa pré-determinada e baseados em uma série de testes automatizados realizados em um número pré-determinado de instâncias de treinamento. O Irace é executado dentro de um pacote do software R (R Development Core Team, 2008).

Os experimentos para a determinação dos valores dos parâmetros foram realizados utilizando-se 20 instâncias de treinamento. Para garantir a representatividade dessas instâncias, foram selecionadas combinações com parâmetros em toda a faixa de variação deles. Os conjuntos de valores dos parâmetros foram definidos conforme a seguir:

- $maxTemperature \in \{10, 20, 50, 100, 500, 1000\}$
- $minTemperature \in \{0.1, 1.0, 5.0\}$
- $maxIterations \in \{1n, 2n, 5n, 10n\}$
- $\alpha \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$.

Após executar o Irace, foram retornados os seguintes valores para os parâmetros: *maxTemperature*: 500, *minTemperature*: 0.1, *maxIterations*: $2n$ e α: 0.002. Os resultados encontrados pela formulação MILP e pelos algoritmos VNS e SA são descritos em detalhes na Tabela 6.1.

Na Tabela 6.1, as primeiras quatro colunas descrevem as instâncias. As colunas *ID*, n, m e Q, informam, respectivamente, o número de identificação da instância, o número de ordens de manutenção preventivas, o número de equipes de trabalho disponíveis e o número de equipamentos. Os resultados obtidos pela formulação MILP são descritos nas colunas *Obj*, $\#T$, $\#P$, *Gap* e *Time(s)*. A coluna *Obj* mostra o valor da solução obtida ao final do processamento, $\#T$ informa o número de equipes de trabalho alocada para a execução das $\#P$ ordens de manutenção preventivas. A coluna *Gap* mostra o *gap* relativo entre o limite superior (*Obj*) e o *LB* calculado como (*Obj* - *LB*)/*Obj*, sendo *LB* o limite inferior obtido no final do processamento. A coluna *Tempo*(s) informa o tempo de processamento, em segundos, gasto pela formulação MILP. Nas próximas colunas são apresentados os resultados do VNS e do SA. A coluna *Obj* mostra a melhor solução encontrada em um total de 10 execuções do algoritmo. A coluna σ informa o desvio-padrão do valor de todas as 10 soluções. As colunas $\#T$, $\#P$ têm o mesmo significado que os das colunas da formulação MILP, se referindo à melhor solução encontrada pelo algoritmo heurístico. Finalmente, a coluna *Time*(s) mostra o tempo de processamento total para encontrar todas as 10 soluções pelo respectivo algoritmo heurístico. Note que para destacar a melhor solução encontrada, os melhores resultados estão em negrito.

Tabela 6.1: Comparação MILP X SA x VNS

Instância				**MILP**					**SA**					**VNS**				
ID	**n**	**m**	**Q**	**Obj**	**#T**	**#P**	***Gap***	**T(s)**	**Obj**	**#T**	**#P**	$\boldsymbol{\sigma}$	**T(s)**	**Obj**	**#T**	**#P**	$\boldsymbol{\sigma}$	**T(s)**
1	20	2	2	**434**	2	18	0,00	0,28	**434**	2	18	0,00	4,98	**434**	2	18	0,00	0,20
2	20	3	2	**219**	3	19	0,00	7,21	**219**	3	19	0,00	5,38	**219**	3	19	0,00	0,19
3	20	4	2	**219**	3	19	0,00	81,96	**219**	3	19	0,00	5,69	**219**	3	19	0,00	0,20
4	20	5	2	**219**	3	19	0,00	85,5	**219**	3	19	0,00	5,69	**219**	3	19	0,00	0,23
5	20	10	2	**219**	3	19	0,00	28,79	**219**	3	19	0,00	6,07	**219**	3	19	0,00	0,29
6	30	2	2	**434**	2	28	0,00	2,04	**434**	2	28	0,00	13,55	**434**	2	28	91,07	1,39
7	30	3	2	**219**	3	29	0,00	10,7	**219**	3	29	0,00	15,32	**219**	3	29	0,00	1,23
8	30	4	2	**219**	3	29	0,00	333,35	**219**	3	29	0,00	15,28	**219**	3	29	0,00	1,24
9	30	5	2	220	4	29	0,01	1h	**219**	3	29	0,00	16,36	**219**	3	29	0,00	1,49
10	30	10	2	**219**	3	29	0,01	1h	**219**	3	29	0,00	16,96	**219**	3	29	0,00	1,32
11	40	2	2	**434**	2	38	0,00	57,18	650	2	37	0,00	27,12	650	2	37	91,07	4,78
12	40	3	2	**219**	3	39	0,00	186,27	**219**	3	39	0,00	28,01	**219**	3	39	0,00	4,62
13	40	4	2	220	4	39	0,01	1h	**219**	3	39	0,00	29,57	**219**	3	39	0,00	4,73
14	40	5	2	220	4	39	0,99	1h	**219**	3	39	0,00	29,43	**219**	3	39	0,00	4,79

Continua na próxima página

Tabela 6.1 – (continuação) Comparação MILP X SA x VNS

Instância				**MILP**					**SA**					**VNS**				
ID	**n**	**m**	**Q**	**Obj**	**#T**	**#P**	***Gap***	**T(s)**	**Obj**	**#T**	**#P**	σ	**T(s)**	**Obj**	**#T**	**#P**	σ	**T(s)**
15	40	10	2	221	5	39	0,02	1h	**219**	3	39	0,00	30,09	**219**	3	39	0,00	5,20
16	60	2	2	**434**	2	58	0,00	67,67	**434**	2	58	91,07	76,75	650	2	57	198,49	26,65
17	60	3	2	**219**	3	59	0,00	1358,16	**219**	3	59	0,00	78,62	**219**	3	59	91,07	28,88
18	60	4	2	220	4	59	1,00	1h	**219**	3	59	0,00	79,97	**219**	3	59	68,31	29,93
19	60	5	2	221	5	59	0,99	1h	**219**	3	59	0,00	80,73	**219**	3	59	91,28	29,55
20	60	10	2	652	4	57	1,00	1h	**219**	3	59	0,00	83,23	**219**	3	59	91,07	30,75
21	80	2	2	**866**	2	76	0,50	1h	1082	2	75	0,00	139,27	1082	2	75	245,22	102,66
22	80	3	2	**219**	3	79	0,99	1h	**219**	3	79	0,00	142,81	435	3	78	151,03	109,48
23	80	4	2	436	4	78	1,00	1h	**219**	3	79	0,00	150,23	435	3	78	204,92	112,71
24	80	5	2	653	5	77	1,00	1h	**219**	3	79	0,00	149,12	435	3	78	145,79	119,89
25	80	10	2	653	5	77	1,00	1h	**219**	3	75	0,00	159,53	435	3	78	145,79	120,70
26	20	3	3	**579**	3	17	0,00	2,07	**579**	3	17	0,00	4,91	**579**	3	17	0,00	0,23
27	20	4	3	**220**	4	19	0,00	14,65	**220**	4	19	0,00	5,26	**220**	4	19	0,00	0,22
28	20	5	3	**220**	4	19	0,00	129,85	**220**	4	19	0,00	5,46	**220**	4	19	0,00	0,20
29	20	6	3	**220**	4	19	0,00	3076,37	**220**	4	19	0,00	5,65	**220**	4	19	0,00	0,22
30	20	12	3	**220**	4	19	0,01	1h	**220**	4	19	0,00	6,41	**220**	4	19	0,00	0,23
31	30	3	3	**579**	3	27	0,00	6,33	**579**	3	27	0,00	11,83	**579**	3	27	48,60	1,21
32	30	4	3	**220**	4	29	0,00	38,31	**220**	4	29	0,00	12,39	**220**	4	29	0,00	1,03
33	30	5	3	**220**	4	29	0,00	316,8	**220**	4	29	0,00	12,55	**220**	4	29	0,00	1,04
34	30	6	3	**220**	4	29	0,01	1h	**220**	4	29	0,00	13,18	**220**	4	29	0,00	1,06
35	30	12	3	223	7	29	0,99	1h	**220**	4	29	0,00	14,32	**220**	4	29	0,00	1,23
36	40	3	3	**579**	3	37	0,00	807,32	**579**	3	37	69,56	25,58	867	3	35	97,19	4,46
37	40	4	3	**220**	4	39	0,00	238,45	**220**	4	39	0,00	27,26	**220**	4	39	0,00	4,27
38	40	5	3	221	5	39	0,01	1h	**220**	4	39	0,00	27,87	**220**	4	39	0,00	4,33
39	40	6	3	221	5	39	0,01	1h	**220**	4	39	0,00	28,25	**220**	4	39	0,00	4,29
40	40	12	3	225	9	39	0,99	1h	**220**	4	39	0,00	29,76	**220**	4	39	0,00	4,91
41	60	3	3	**1515**	3	51	0,76	1h	1659	3	50	60,72	65,51	2019	3	48	106,25	24,58
42	60	4	3	**220**	4	59	0,00	1619,31	**220**	4	59	0,00	78,53	**220**	4	59	91,07	29,82
43	60	5	3	221	5	59	0,01	1h	**220**	4	59	0,00	85,47	**220**	4	59	0,48	29,17
44	60	6	3	222	6	59	0,99	1h	**220**	4	59	0,00	80,09	**220**	4	59	0,42	30,29
45	60	12	3	225	9	59	1,00	1h	**220**	4	59	0,00	84,74	**220**	4	59	0,42	31,30
46	80	3	3	3891	3	59	0,32	1h	**1803**	3	69	74,36	115,97	2163	3	67	135,76	92,79
47	80	4	3	2524	4	68	0,13	1h	**220**	4	79	0,00	136,67	**220**	4	79	79,68	104,33
48	80	5	3	2525	5	68	0,26	1h	**220**	4	79	0,00	140,41	**220**	4	79	93,35	105,77
49	80	6	3	2886	6	66	0,42	1h	**220**	4	79	0,00	143,59	**220**	4	79	75,55	116,49
50	80	12	3	12895	7	16	0,90	1h	**220**	4	79	0,00	149,34	**220**	4	79	0,53	121,10
51	20	3	4	**723**	3	16	0,00	530,62	**723**	3	16	0,00	4,79	**723**	3	16	86,20	0,27
52	20	4	4	**220**	4	19	0,00	1862,58	**220**	4	19	0,00	5,49	**220**	4	19	0,00	0,22
53	20	5	4	**220**	4	19	0,01	1h	**220**	4	19	0,00	5,66	**220**	4	19	0,00	0,21
54	20	6	4	**220**	4	19	0,01	1h	**220**	4	19	0,00	5,73	**220**	4	19	0,00	0,23
55	20	12	4	**220**	4	19	0,01	1h	**220**	4	19	0,00	6,51	**220**	4	19	0,00	0,23
56	30	3	4	**723**	3	26	0,40	1h	**723**	3	26	0,00	11,60	939	3	25	97,49	1,24

Continua na próxima página

Tabela 6.1 – (continuação) Comparação MILP X SA x VNS

Instância				MILP					SA					VNS				
ID	n	m	Q	Obj	#T	#P	*Gap*	T(s)	Obj	#T	#P	σ	T(s)	Obj	#T	#P	σ	T(s)
57	30	4	4	**220**	4	29	0,01	1h	**220**	4	29	0,00	12,53	**220**	4	29	0,00	1,14
58	30	5	4	221	5	29	0,02	1h	**220**	4	29	0,00	12,96	**220**	4	29	0,00	1,09
59	30	6	4	**220**	4	29	0,01	1h	**220**	4	29	0,00	13,07	**220**	4	29	0,00	1,15
60	30	12	4	222	6	29	0,99	1h	**220**	4	29	0,00	14,10	**220**	4	29	0,00	1,35
61	40	3	4	**867**	3	35	1,00	1h	**867**	3	35	69,56	22,36	1083	3	34	123,55	4,18
62	40	4	4	**220**	4	39	0,01	1h	**220**	4	39	0,00	24,75	**220**	4	39	0,00	3,95
63	40	5	4	221	5	39	0,02	1h	**220**	4	39	0,00	25,28	**220**	4	39	0,00	3,98
64	40	6	4	222	6	39	0,21	1h	**220**	4	39	0,00	25,58	**220**	4	39	0,00	3,72
65	40	12	4	223	7	39	1,00	1h	**220**	4	39	0,00	27,39	**220**	4	39	0,00	4,47
66	60	3	4	**2163**	3	47	1,00	1h	2307	3	46	0,00	62,18	2667	3	45	130,57	23,05
67	60	4	4	652	4	57	0,73	1h	**220**	4	59	0,00	75,01	**220**	4	59	102,11	28,53
68	60	5	4	221	5	59	0,61	1h	**220**	4	59	0,00	77,99	**220**	4	59	0,52	30,02
69	60	6	4	222	6	59	0,99	1h	**220**	4	59	0,00	78,92	**220**	4	59	0,53	29,18
70	60	12	4	1086	6	55	1,00	1h	**220**	4	59	0,00	83,57	**220**	4	59	0,52	35,27
71	80	3	4	3171	3	62	1,00	1h	**3099**	3	58	60,72	98,83	3747	3	59	81,74	72,56
72	80	4	4	**220**	4	79	0,01	1h	**220**	4	79	0,00	129,94	**220**	4	79	127,00	95,36
73	80	5	4	221	5	79	0,99	1h	**220**	4	79	0,00	131,94	**220**	4	79	0,53	103,66
74	80	6	4	222	6	79	0,99	1h	**220**	4	79	0,00	142,62	**220**	4	79	0,48	117,94
75	80	12	4	8215	7	42	1,00	1h	**220**	4	79	0,00	143,99	221	5	79	0,00	102,02
76	20	4	5	**724**	4	16	0,00	62,93	**724**	4	16	0,00	4,83	**724**	4	16	98,66	0,28
77	20	5	5	**221**	5	19	0,00	277,79	**221**	5	19	0,00	5,37	**221**	5	19	0,00	0,21
78	20	6	5	**221**	5	19	0,01	1h	**221**	5	19	0,00	5,59	**221**	5	19	0,00	0,21
79	20	7	5	**221**	5	19	0,01	1h	**221**	5	19	0,00	6,01	**221**	5	19	0,00	0,21
80	20	14	5	**221**	5	19	0,01	1h	**221**	5	19	0,00	6,32	**221**	5	19	0,00	0,25
81	30	4	5	**724**	4	26	0,20	1h	**724**	4	26	0,00	10,35	796	4	26	74,36	1,05
82	30	5	5	**221**	5	29	0,01	1h	**221**	5	29	0,00	11,26	**221**	5	29	0,00	1,09
83	30	6	5	**221**	5	29	0,01	1h	**221**	5	29	0,00	11,72	**221**	5	29	0,00	1,10
84	30	7	5	**221**	5	29	0,01	1h	**221**	5	29	0,00	12,00	**221**	5	29	0,00	1,08
85	30	14	5	225	9	29	0,98	1h	**221**	5	29	0,00	13,67	**221**	5	29	0,00	1,24
86	40	4	5	**868**	4	35	1,00	1h	**868**	4	35	0,00	20,15	940	4	35	135,98	3,53
87	40	5	5	**221**	5	39	0,01	1h	**221**	5	39	0,00	21,84	**221**	5	39	0,00	3,42
88	40	6	5	222	6	39	0,02	1h	**221**	5	39	0,00	22,56	**221**	5	39	0,00	3,72
89	40	7	5	222	6	39	0,02	1h	**221**	5	39	0,00	22,96	**221**	5	39	0,00	3,45
90	40	14	5	223	7	39	1,00	1h	**221**	5	39	0,00	24,97	**221**	5	39	0,00	3,92
91	60	4	5	**1588**	4	50	1,00	1h	**1588**	4	50	0,00	55,46	1948	4	49	178,96	21,54
92	60	5	5	**221**	5	59	0,02	1h	**221**	5	59	0,00	63,69	**221**	5	59	68,31	22,96
93	60	6	5	222	6	59	0,02	1h	**221**	5	59	0,00	65,13	**221**	5	59	0,32	24,85
94	60	7	5	223	7	59	0,99	1h	**221**	5	59	0,00	65,34	**221**	5	59	0,00	27,40
95	60	14	5	225	9	59	1,00	1h	**221**	5	59	0,00	72,82	**221**	5	59	0,48	28,93
96	80	4	5	3388	4	61	0,92	1h	**3244**	4	61	0,00	91,90	3604	4	60	143,20	67,88
97	80	5	5	509	5	78	0,43	1h	**221**	5	78	0,00	127,32	**221**	5	79	108,40	82,65
98	80	6	5	510	6	78	0,43	1h	**221**	5	77	0,00	126,07	**221**	5	79	68,34	86,15

Continua na próxima página

Tabela 6.1 – (continuação) Comparação MILP X SA x VNS

Instância				**MILP**					**SA**					**VNS**				
ID	**n**	**m**	**Q**	**Obj**	**#T**	**#P**	***Gap***	**T(s)**	**Obj**	**#T**	**#P**	σ	**T(s)**	**Obj**	**#T**	**#P**	σ	**T(s)**
99	80	7	5	511	7	78	0,43	1h	**221**	5	78	0,00	121,09	**221**	5	79	0,48	98,64
100	80	14	5	3102	6	67	0,91	1h	**221**	5	79	0,00	129,10	**221**	5	79	0,57	101,16

De acordo com os valores apresentados na coluna σ, pode ser notado que os algoritmos SA e VNS apresentaram uma boa repetibilidade, pois o desvio-padrão para as 10 soluções obtidas é zero ou próximo a zero para 94% das instâncias para o SA e 65% para o VNS.

Note que, para um dado Q e n, a formulação MILP se comporta melhor para os menores valores de m, uma vez que quanto menor esse valor, menor é a complexidade combinatória do problema. A formulação MILP obteve melhores soluções que o VNS e o SA apenas para as instâncias com número menores de equipes, considerando as instâncias com números iguais de ordens de manutenção preventivas e números iguais de equipamentos. Observa-se que o MILP superou o VNS em apenas 6 instâncias e o VNS em apenas 14.

A Tabela 6.2 apresenta o resultado do desvio percentual (calculado conforme a Equação 6.1) em relação ao melhor resultado para cada abordagem proposta, considerando as instâncias e resultados descritos na Tabela 6.1 . O teste de Friedman retornou um valor-p igual a $1,49 \times 10^{-11}$, o que indica que há uma diferença estatística entre os resultados. O teste *post-hoc* (Tabela 6.3) mostrou que há diferença estatística entre todos os pares de algoritmos. Note que, considerando o desvio médio relativo ao MILP e os resultados da Tabela 6.1, os algoritmos heurísticos propostos obtiveram, em geral, soluções com qualidade superior às das soluções encontradas pela formulação MILP. Pode-se, assim, concluir que tanto o SA quanto o VNS foram melhores que o MILP para esse conjunto de 100 instâncias, sendo que o SA foi o melhor deles.

$$Desvio = \frac{f_{Medio} - f_{Melhor}}{f_{Melhor}} \times 100 \tag{6.1}$$

Foi realizada, também, uma comparação com o indicador que é controlado pela equipe de engenharia de manutenção da empresa. Este indicador é o número percentual de ordens de manutenção que são executadas em relação às programadas. O valor médio histórico é próximo a 50%, mesmo após o uso de horas extras e contratação de mão de obra terceirizada. Com o SA, foi possível alocar 90% das ordens de manutenção e com o

Tabela 6.2: Comparação MILP × SA × VNS

	MILP	**SA**	**VNS**
Média	158,29	**0,91**	7,73

Tabela 6.3: Teste de Friedman para todos os resultados

	MILP	**SA**
SA	$2{,}97 \times 10^{-103}$	
VNS	$2{,}95 \times 10^{-72}$	$1{,}56 \times 10^{-34}$

VNS, 92,5%. O MILP não foi capaz de resolver a instância real. É importante observar que esses percentuais podem ser ainda maiores se forem utilizadas horas extras e/ou mão de obra terceirizada. Cabe destacar que, apesar de o VNS ter alocado uma solução com maior número de ordens de manutenção do que o do algoritmo SA, o valor dessa solução segundo a função objetivo dada pela Equação (4.1) é menor do que a do SA. Isso se deve ao fato de que o VNS alocou muitas ordens de manutenção de menor prioridade.

Pelos resultados observa-se que, com o limite de uma hora de processamento, a formulação MILP encontrou a solução ótima (coluna *Gap* com o valor 0) para 26 instâncias (26% do total). Observa-se que o SA também obteve os ótimos para 26 instâncias e o VNS para 24 instâncias. Observa-se também que o SA obteve os melhores resultados para 96% das instâncias, o VNS para 81% e o MILP para 53%. Interessante observar que o tempo médio para o resultado do SA foi de 50,6 segundos e o do VNS de 27,1 segundos; tempo muito inferior ao limite de 1 hora que foi configurado para o MILP.

Os resultados dos testes validaram o uso dos algoritmos heurísticos para a resolução de problemas de interesse prático. A formulação MILP, apesar de só resolver problemas muito pequenos, foi importante para validar o correto funcionamento e desempenho dos algoritmos heurísticos. Após a validação do SA e do VNS, foram estudadas alternativas e diversos testes preliminares foram executados até a proposição de três novos algoritmos heurísticos, sendo um baseado no próprio VNS e dois em algoritmos genéticos. Os resultados são apresentados e discutidos na próxima Seção.

6.3 Comparação entre MSVNS, BRKGA e BRKMA

Assim como o VNS, o parâmetro $k_{\max}$ do MSVNS foi fixado empiricamente em n, i.e, no número de ordens de manutenção preventivas. Como o objetivo é gerar uma ferramenta que possa ser utilizada para o planejamento pela área de engenharia de manutenção da empresa, o tempo de processamento do MSVNS precisa ser limitado a um valor factível. Assim, considerando que: 1) a equipe de manutenção tem um prazo de aproximadamente dois meses para gerar 14 mapas de 52 semanas envolvendo um total de aproximadamente 500.000 ordens de manutenção; 2) a heurística precisa ser executada pelo menos 5 vezes; 3) o número de ordens de manutenção a serem processadas seria na ordem de 2,5 milhões apenas para a geração de uma opção do mapa de manutenção; 4) um mês tem aproximadamente 2,6 milhões de segundos; então o tempo de teste foi fixado em n segundos, o que permite gerar, em média, dois mapas de 52 semanas para cada unidade operacional.

Os algoritmos BRKGA e BRKMA considerados neste trabalho utilizam o *framework*[2] desenvolvido por Toso and Resende (2011). No BRKGA foi necessário apenas implementar a função de codificação definida para o PPOMPLP (veja a Seção 5.2), enquanto que para o BRKMA também foi necessário incluir a busca local definida e discutida na Seção 5.4.

O BRKGA apresenta quatro parâmetros, enquanto o BRKMA, além dos parâmetros do BRKGA, utiliza um parâmetro L para controlar a frequência em que a busca local é aplicada. Os parâmetros comuns a ambos os algoritmos são listados abaixo:

- P: tamanho da população;
- *pe*: fração da população para ir para o grupo de elite;
- *pm*: fração da população para ser substituída por mutantes;
- *rhoe*: probabilidade de herdar do grupo elite;

Note que o BRKMA é equivalente ao BRKGA quando $L = 0$, ou seja, quando a busca local não é considerada. Dessa forma, a fim de determinar o melhor algoritmo (dentre o BRKGA e BRKMA) e os seus melhores parâmetros, foi feita uma calibração utilizando a ferramenta Irace, já mencionada anteriormente.

[2]Disponível em `http://mauricio.resende.info/src/brkgaAPI/`

Os experimentos para a determinação dos parâmetros foram realizados utilizando-se 16 instâncias de treinamento. O tamanho da população foi definido para ser proporcional ao tamanho dos cromossomos, que nesse caso é igual ao número de ordens de manutenção. Dessa forma, a população P ficou definida como: $P = a \cdot n$. Apesar de ser aconselhado por Toso and Resende (2011) valores de P maiores ou iguais ao do tamanho do cromossomo, um valor de $P \geq n$, testes preliminares mostraram que este valor permitia apenas algumas gerações para a instância real completa. Para verificar o desempenho com diferentes valores de P, primeiro o Irace foi executado com valores de $a \in \{1.0, 1.5, 2.0, 4.0\}$. A melhor configuração sugerida pelo Irace foi com $a = 1$. Interessante observar que o valor de a está no limite inferior dos valores testados. Então um segundo teste foi realizado com valores de $a \in \{1.00, 0.01, 0.005, 0.0025, 0.00125\}$. A melhor configuração sugerida pelo Irace foi com $a = 0.01$. Dessa forma, verificou-se que para as instâncias teste, o BRKGA teve um desempenho melhor para valores $P \leq n$. Com base nesses resultados, foi decidido utilizar uma fração do valor de n. Para que o tamanho da população não ficasse muito pequeno, o valor mínimo de P foi definido como 20. Para a definição final dos parâmetros, os valores considerados foram definidos conforme a seguir:

- $a \in \{0.01, 0.005, 0.0025, 0.00125\}$
- $pe \in \{0.1, 0.15, 0.2, 0.25\}$
- $pm \in \{0.1, 0.15, 0.2, 0.25, 0.3\}$
- $rhoe \in \{0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\}$
- L: $\in \{0, 10, 20, 40, 80, 160\}$

A melhor configuração retornada pela ferramenta Irace não utiliza o procedimento de busca local, ou seja, o algoritmo BRKGA foi considerado melhor nessa calibração, uma vez que $L = 0$. Os demais parâmetros assumem os seguintes valores: $a = 0.01$, $pe = 0.15$, $pm = 0.15$ e $rhoe = 0.8$.

Dentre as configuração que utilizam o procedimento de busca local, a melhor retornada pela ferramenta Irace tem os seguintes valores para os parâmetros: $a = 0.00125$, $pe = 0.1$, $pm = 0.2$, $rhoe = 0.7$ e $L = 80$.

Tanto o BRKGA quanto o BRKMA foram configurados conforme os valores retornados pelo Irace.

Como o SA foi o algoritmo que obteve os melhores resultados na comparação com o MILP e o VNS, foram realizados testes para comparação dos resultados do BRKGA e do BRKMA com o SA. Para uma comparação justa, os algoritmos foram configurados para utilizar apenas um núcleo de processamento. Nestes testes foram utilizadas as mesmas 100 instâncias da Seção 6.2. Os resultados deste teste estão reportados na Tabela A.1 do Apêndice A. Neste conjunto de testes, o SA teve um desvio médio percentual de 1,94%, o BRKGA de 1,59%, e o BRKMA de 1,56%. O teste de Friedman retornou um valor-p de 0,85428, valor que confirma que os resultados são estatisticamente iguais. Na comparação quantitativa de melhores resultados, o SA obteve os melhores resultados para 94 instâncias, o BRKGA para 97, e o BRKMA para 98. A partir da análise dos resultados, pode-se concluir que tanto o BRKGA quanto o BRKMA são iguais ou melhores que o SA para as instâncias testadas.

Para a comparação do MSVNS, BRKGA e BRKMA, foram realizados testes em um computador Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz x 40, com 384GB de memória RAM sob o sistema operacional CentOS Release 6.8 (Final) Kernel Linux 2.6.32-642.1.1.el6.x86_64. Este computador pertence ao Departamento de Computação (DECOM) da Universidade Federal de Ouro Preto (UFOP). Foram utilizados 39 *threads* para o processamento paralelo. Cada instância foi executada 5 vezes para cada algoritmo proposto. Cada algoritmo foi limitado para ser executado por no máximo n segundos, ou seja, tempo em segundos igual ao número de ordens de manutenção.

A Tabela 6.4 mostra os resultados obtidos por todos os algoritmos desenvolvidos para solucionar o problema. As 4 primeiras colunas têm o mesmo significado da Tabela 6.1. A coluna *Media* informa o resultado médio; a coluna *Melhor*, o melhor resultado entre o MSVNS, BRKGA e BRKMA; a coluna *Desvio*, o desvio médio percentual, calculado conforme Equação (6.1) entre as 5 vezes que o algoritmo foi executado.

Tabela 6.4: Comparação MSVNS x BRKGA x BRKMA

Instância				**MSVNS**			**BRKGA**			**BRKMA**		
ID	**n**	**m**	**Q**	**Média**	**Melhor**	**Desvio**	**Média**	**Melhor**	**Desvio**	**Média**	**Melhor**	**Desvio**
1	20	2	2	434	**434**	0,0	434	**434**	0,0	434	**434**	0,0
2	20	3	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
3	20	4	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
4	20	5	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
5	20	10	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
6	30	2	2	434	**434**	0,0	434	**434**	0,0	434	**434**	0,0
7	30	3	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
8	30	4	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0

Continua na próxima página

Tabela 6.4 – (continuação) Comparação MSVNS x BRKGA x BRKMA

Instância				MSVNS			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
9	30	5	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
10	30	10	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
11	40	2	2	650	650	49,8	520	**434**	19,8	607	**434**	39,9
12	40	3	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
13	40	4	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
14	40	5	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
15	40	10	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
16	60	2	2	650	**650**	0,0	650	**650**	0,0	650	**650**	0,0
17	60	3	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
18	60	4	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
19	60	5	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
20	60	10	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
21	80	2	2	1082	**1082**	0,0	1082	**1082**	0,0	1082	**1082**	0,0
22	80	3	2	219	**219**	0,0	262	**219**	19,6	219	**219**	0,0
23	80	4	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
24	80	5	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
25	80	10	2	219	**219**	0,0	219	**219**	0,0	219	**219**	0,0
26	20	3	3	579	**579**	0,0	579	**579**	0,0	579	**579**	0,0
27	20	4	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
28	20	5	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
29	20	6	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
30	20	12	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
31	30	3	3	579	**579**	0,0	579	**579**	0,0	579	**579**	0,0
32	30	4	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
33	30	5	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
34	30	6	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
35	30	12	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
36	40	3	3	723	723	24,9	694	**579**	19,9	694	**579**	19,9
37	40	4	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
38	40	5	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
39	40	6	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
40	40	12	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
41	60	3	3	1760	1731	22,0	1616	**1443**	12,0	1573	**1443**	9,0
42	60	4	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
43	60	5	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
44	60	6	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
45	60	12	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
46	80	3	3	2033	1947	40,9	1659	1659	15,0	1659	**1443**	15,0
47	80	4	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
48	80	5	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
49	80	6	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
50	80	12	3	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0

Continua na próxima página

Tabela 6.4 – (continuação) Comparação MSVNS x BRKGA x BRKMA

Instância				MSVNS			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
51	20	3	4	723	**723**	0,0	723	**723**	0,0	723	**723**	0,0
52	20	4	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
53	20	5	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
54	20	6	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
55	20	12	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
56	30	3	4	723	**723**	0,0	723	**723**	0,0	723	**723**	0,0
57	30	4	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
58	30	5	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
59	30	6	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
60	30	12	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
61	40	3	4	1011	1011	16,6	939	**867**	8,3	953	**867**	9,9
62	40	4	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
63	40	5	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
64	40	6	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
65	40	12	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
66	60	3	4	2350	2307	12,4	2134	**2091**	2,1	2149	**2091**	2,8
67	60	4	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
68	60	5	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
69	60	6	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
70	60	12	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
71	80	3	4	3416	3315	15,6	3056	**2955**	3,4	3099	**2955**	4,9
72	80	4	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
73	80	5	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
74	80	6	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
75	80	12	4	220	**220**	0,0	220	**220**	0,0	220	**220**	0,0
76	20	4	5	724	**724**	0,0	724	**724**	0,0	724	**724**	0,0
77	20	5	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
78	20	6	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
79	20	7	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
80	20	14	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
81	30	4	5	724	**724**	0,0	724	**724**	0,0	724	**724**	0,0
82	30	5	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
83	30	6	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
84	30	7	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
85	30	14	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
86	40	4	5	868	**868**	0,0	868	**868**	0,0	882	**868**	1,6
87	40	5	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
88	40	6	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
89	40	7	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
90	40	14	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
91	60	4	5	1847	1804	16,3	1631	**1588**	2,7	1602	**1588**	0,9
92	60	5	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0

Continua na próxima página

Tabela 6.4 – (continuação) Comparação MSVNS x BRKGA x BRKMA

Instância				MSVNS			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
93	60	6	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
94	60	7	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
95	60	14	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
96	80	4	5	3359	3316	10,9	3086	**3028**	1,9	3114	**3028**	2,8
97	80	5	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
98	80	6	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
99	80	7	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
100	80	14	5	221	**221**	0,0	221	**221**	0,0	221	**221**	0,0
101	150	148	120	1756	**1756**	0,0	1756	**1756**	0,0	1756	**1756**	0,0
102	150	75	129	30	**30**	0,0	30	**30**	0,0	30	**30**	0,0
103	150	102	91	3273	**3273**	0,0	3273	**3273**	0,0	3273	**3273**	0,0
104	150	57	126	24	**24**	0,0	24	**24**	0,0	24	**24**	0,0
105	150	92	93	49	**49**	0,0	49	**49**	0,0	49	**49**	0,0
106	150	71	101	106	**106**	0,0	106	**106**	0,0	106	**106**	0,0
107	300	158	179	1881	**1776**	5,9	1969	1932	10,9	2006	1932	13,0
108	300	221	239	2452	**2452**	0,0	2452	**2452**	0,0	2452	**2452**	0,0
109	300	112	177	3361	**3360**	0,0	3362	3361	0,1	3363	3362	0,1
110	300	75	181	36	**35**	2,9	38	37	8,6	38	37	8,6
111	300	121	162	281	**281**	0,0	283	282	0,7	283	282	0,7
112	300	119	176	308	**308**	0,0	309	**308**	0,3	309	**308**	0,3
113	600	165	329	4484	**4409**	1,7	4577	4545	3,8	4698	4609	6,6
114	600	256	388	3385	**3384**	0,0	3387	**3384**	0,1	3398	3388	0,4
115	600	120	288	8579	**8578**	0,0	8581	8580	0,0	8593	8584	0,2
116	600	77	215	42	**42**	0,0	106	43	152,4	84	43	100,0
117	600	126	279	414	**414**	0,0	416	415	0,5	417	416	0,7
118	600	120	292	5664	**5663**	0,0	5665	5664	0,0	5667	5665	0,1
119	1200	186	519	10908	**10784**	1,1	11297	11035	4,8	11271	11077	4,5
120	1200	263	666	9535	**9527**	0,1	9634	9587	1,1	9632	9575	1,1
121	1200	122	470	21932	**21930**	0,0	22268	22075	1,5	22434	22318	2,3
122	1200	88	252	324	**309**	4,9	1016	765	228,8	664	390	114,9
123	1200	130	420	528	**526**	0,4	665	532	26,4	779	679	48,1
124	1200	122	403	6842	**6841**	0,0	6852	**6841**	0,2	6850	6842	0,1
125	2400	188	738	26991	26777	1,1	27797	**26705**	4,1	27835	27179	4,2
126	2400	283	869	24590	**24553**	0,2	24921	24732	1,5	24843	24713	1,2
127	2400	130	603	38293	**38094**	0,5	41268	39614	8,3	40402	38764	6,1
128	2400	90	278	2273	2000	43,4	2333	2026	47,2	1953	**1585**	23,2
129	2400	132	701	945	**816**	15,8	2319	1768	184,2	2244	1608	175,0
130	2400	126	561	8280	**8259**	0,3	9454	8839	14,5	9363	8775	13,4
131	4800	197	1128	60525	**59580**	1,6	62883	62395	5,5	62649	61138	5,2
132	4800	78	1286	42215	**41925**	0,7	43650	42643	4,1	43981	42781	4,9
133	4800	130	720	95815	94780	3,8	97950	95629	6,1	95958	**92320**	3,9
134	4800	91	294	208749	207390	5,2	200655	199117	1,2	199368	**198340**	0,5

Continua na próxima página

Tabela 6.4 – (continuação) Comparação MSVNS x BRKGA x BRKMA

Instância				MSVNS			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
135	4800	135	990	6803	6194	13,2	7279	**6008**	21,2	7831	6088	30,3
136	4800	129	713	17181	**16745**	2,6	19738	19062	17,9	18138	17749	8,3
137	9600	132	816	51712	50272	44,5	39650	**35778**	10,8	39445	38004	10,2
138	19200	132	908	208419	198838	102,8	108320	**102773**	5,4	108290	103863	5,4
139	33484	145	1032	621953	616479	182,6	237002	223752	7,7	222586	**220048**	1,2

A Tabela 6.5 mostra o desvio médio percentual de todas as instâncias para cada algoritmo. O teste de Friedman retornou um valor-p de 0,69, valor que confirma que os resultados são estatisticamente iguais. Apesar de haver uma diferença de valores médios, por essa análise, pode-se concluir que, dos algoritmos testados, não houve um algoritmo que se destacou em relação aos demais.

Tabela 6.5: Desvio médio global dos algoritmos MSVNS, BRKGA e BRKMA

MSVNS	BRKGA	BRKMA
4,64	6,36	5,04

Foi, então, realizada uma outra análise para verificar se há alguma diferença dos algoritmos com o crescimento do tamanho das instâncias. Os resultados foram agrupados em 4 conjuntos, os quais variam de acordo com a quantidade de ordens de manutenção, conforme a seguir:

- P: instâncias de 20 a 80 ordens de manutenção
- M: instâncias de 150 a 600 ordens de manutenção
- G: instâncias de 1200 a 4800 ordens de manutenção
- GG: instâncias de 9600 a 33484 ordens de manutenção

A Tabela 6.6 e a Figura 6.1 mostram os resultados dos desvios para cada conjunto mencionado anteriormente. Conforme será discutido a seguir, observa-se que para cada conjunto de instâncias há um algoritmo que possui um melhor desempenho.

Tabela 6.6: Desvio médio dos algoritmos MSVNS, BRKGA e BRKMA por conjunto de instâncias

Conjunto	MSVNS	BRKGA	BRKMA
P	2,09	**1,05**	1,07
M	**0,59**	9,85	7,25
G	**5,27**	32,14	24,85
GG	109,99	7,97	**5,59**

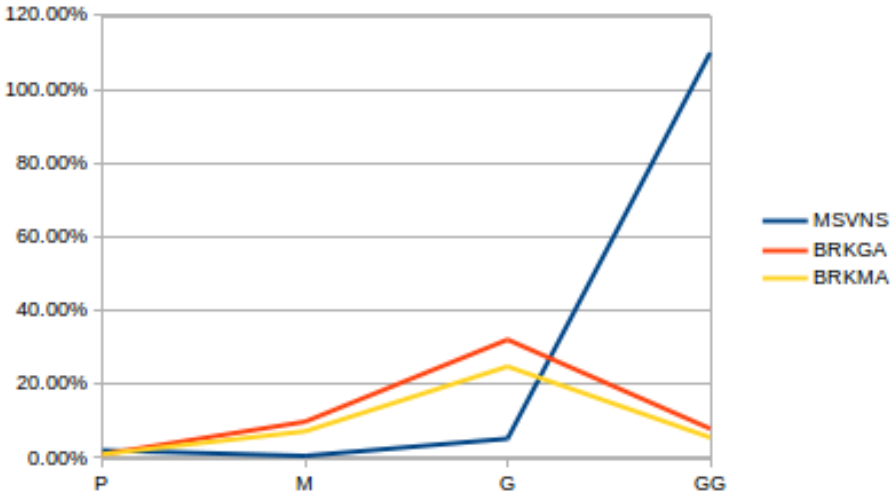

Figura 6.1: Análise comparativa por conjunto de instâncias.

Nas instâncias de dimensões pequenas (linha P da Tabela 6.6), o teste de Friedman retornou um valor-p de 0,006572, valor que confirma que os resultados são estatisticamente diferentes. Os testes *post-hoc*, mostrados na Tabela 6.7, revelam que todos os algoritmos possuem resultados estatisticamente diferentes. Por essa análise, pode-se concluir que, dos algoritmos testados, o MSVNS é o pior deles nesse conjunto de instâncias (com desvio de 2,09%) e os algoritmos BRKGA e BRKMA, apesar de serem estatisticamente diferentes, estão praticamente empatados com os melhores resultados (desvios de 1,05% e 1,07%, respectivamente).

Nas instâncias de dimensões médias (linha M da Tabela 6.6), o teste de Friedman retornou um valor-p de 0,009862, valor que confirma que os resultados são estatisticamente diferentes. Os testes *post-hoc* (Tabela 6.8) revelam que os algoritmos BRKGA e BRKMA possuem resultados estatisticamente iguais, e o MSVNS tem um resultado

Tabela 6.7: Resultados do Teste de Friedman nas instâncias de dimensões pequenas

	MSVNS	**BRKGA**
BRKGA	$5{,}24 \times 10^{-42}$	
BRKMA	$2{,}97 \times 10^{-30}$	0,000122

estatisticamente diferente dos demais. Por essa análise pode-se concluir que, dos algoritmos testados, o MSVNS é o melhor algoritmo para o conjunto de instâncias médias (com desvio de 0,59%) e os algoritmos BRKGA e BRMKA são os piores (com desvios de 9,85% e 7,25%, respectivamente).

Tabela 6.8: Resultados do Teste de Friedman nas instâncias de dimensões médias

	MSVNS	**BRKGA**
BRKGA	$4{,}45 \times 10^{-6}$	
BRKMA	$1{,}71 \times 10^{-8}$	0,050286

Nas instâncias de dimensões grandes (linha G da Tabela 6.6), o teste de Friedman retornou um valor-p de 0,000064, valor que confirma que os resultados são estatisticamente diferentes. Os testes *post-hoc* (Tabela 6.9) mostram que todos os pares de algoritmos possuem resultados estatisticamente diferentes. Por essa análise pode-se concluir que, dos algoritmos testados, o MSVNS é o melhor algoritmo para o conjunto de instâncias grandes (com desvio de 5,27%) e BRKGA é o pior (com desvio de 32,14%).

Tabela 6.9: Resultados do Teste de Friedman nas instâncias de dimensões grandes

	MSVNS	**BRKGA**
BRKGA	$3{,}96 \times 10^{-14}$	
BRKMA	$1{,}46 \times 10^{-10}$	0,001137

Nas instâncias de dimensões gigantes (linha GG da Tabela 6.6), entre as quais está a instância real completa com 33484 ordens de manutenção, a variação nos resultados foi

expressiva. O teste de Friedman deu um valor-p de 0,000006, indicando que os resultados são estatisticamente diferentes. Os testes *post-hoc* (Tabela 6.10) mostraram que todos os pares de algoritmos têm resultados estatisticamente diferentes. O melhor algoritmo foi o BRKMA, com desvio de 5,59%. O MSVNS foi o pior deles, com desvio de 109,99%.

Tabela 6.10: Resultados do Teste de Friedman nas instâncias de dimensões gigantes

	MSVNS	**BRKGA**
BRKGA	$8{,}25 \times 10^{-12}$	
BRKMA	$6{,}01 \times 10^{-15}$	0.000224

Todos os testes de Friedman mostraram haver diferença estatística nos resultados dos algoritmos. Nos testes *post-hoc*, apenas não houve diferença estatística em um dos pares de algoritmos para as instâncias de dimensões médias (BRKGA e BRKMA). Esse resultado estatístico facilita a interpretação gráfica dos resultados. Como pode ser observado na Figura 6.1, apesar da diferença estatística dos resultados, todos os algoritmos tiveram um desempenho semelhante para as instâncias de dimensões pequenas. Para as instâncias de dimensões médias e grandes, observa-se um aumento no desvio do BRKGA e do BRKMA; dessa forma, pode ser concluído que para essas instâncias o MSVNS teve um desempenho melhor. Para as instâncias de dimensões gigantes, houve uma inversão dos resultados e uma convergência dos resultados do BRKGA e do BRKMA, os quais se destacam como os dois melhores algoritmos para as instâncias de tamanho real ou próxima ao real. Observa-se um aumento significativo do desvio médio do MSVNS, mostrando uma tendência de piora expressiva com o aumento da dimensão das instâncias. Observa-se, também, que, apesar da diferença estatística dos resultados, os algoritmos BRKGA e BRKMA produzem resultados muito próximos, e seguem um mesmo comportamento para todos os conjuntos de dimensões de instâncias.

Na comparação com o indicador que é controlado pela equipe de engenharia de manutenção da empresa (% de ordens de manutenções que são executadas em relação às programadas), observa-se que o BRKMA conseguiu atender a 95% das ordens de manutenção da instância real, superando o algoritmo VNS (que atendeu a 92,5%) e a equipe que faz essa tarefa na empresa, que consegue atender apenas cerca de 50% dos pedidos.

Capítulo 7

Conclusões e Trabalhos Futuros

Este trabalho propôs soluções para o Problema de Alocação de Ordens de Manutenção de uma empresa de beneficiamento de minério de ferro.

Inicialmente foi desenvolvida uma formulação de programação linear inteira mista (MILP). Entretanto, com essa formulação não foi possível resolver uma instância real, que envolvia mais de 30 mil ordens de manutenção. Apesar de não resolver problemas de dimensões grandes, essa formulação foi essencial para o entendimento formal do problema, assim como para ser utilizada para validar os algoritmos heurísticos propostos. Para resolver as instâncias de interesse prático, foram propostos, inicialmente, algoritmos baseados nas meta-heurísticas *Simulated Annealing* (SA) e *Variable Neighborhood Search* (VNS). Após a validação desses algoritmos, foram propostos três outros algoritmos heurísticos: *Multi-Start Variable Neighborhood Search* (MSVNS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Biased Random-Key Memetic Algorithm* (BRKMA).

Foi fornecida pela equipe de manutenção da empresa em foco uma instância com 33484 ordens de manutenção. Essa instância foi decomposta em 100 instâncias menores de forma homogênea para a realização do primeiro conjunto de testes para analisar os resultados do MILP, do SA e do VNS. O SA e o VNS tiveram um resultado muito melhor que o MILP, com destaque para o SA, que teve um desvio de apenas 0,91% se comparado com um desvio de 7,73% do VNS e 158,20% do MILP.

Para o segundo conjunto de testes (envolvendo os algoritmos MSVNS, BRKGA e BRKMA) foram adicionadas novas instâncias. Para isso, foram obtidas 5 novas instâncias reais, das quais foram geradas mais instâncias, somando um total de 139 instâncias. O teste realizado com todas as instâncias indicou que não houve um algoritmo estatis-

ticamente melhor para a resolução do problema. Para verificar se esse desempenho é homogêneo para qualquer tamanho de instância, elas foram divididas em 4 grupos e as instâncias classificadas como de dimensão pequena (0 a 80 ordens de manutenção), de dimensão média (150 a 600 ordens de manutenção), de dimensão grande (1200 a 4800 ordens) e de dimensão gigante (9600 a 33484 ordens). Foi observado que, dependendo da dimensão da instância, um algoritmo diferente tem o melhor desempenho. Para as instâncias de dimensões pequenas, o algoritmo BRKGA obteve os melhores resultados. Nas instâncias de dimensões médias e grandes o MSVNS teve o melhor resultado. O BRKMA, por sua vez, foi o melhor nas instâncias de dimensões gigantes. Tendo em vista o interesse prático de resolução do mapa de 52 semanas, este resultado da instância gigante é o que deve ser considerado na empresa em estudo.

Este trabalho contribuiu tanto com uma formulação de programação matemática para um problema de interesse prático na indústria, o de de alocação de ordens de manutenção, quanto para a proposição de algoritmos heurísticos para resolver esse problema. Os resultados obtidos mostraram que os algoritmos desenvolvidos são capazes de produzir soluções melhores do que aquelas empregadas pela empresa, e em um tempo adequado para a tomada de decisão.

Como sugestões de trabalhos futuros, são propostas as seguintes atividades:

- Teste de longa duração das demais 5 instâncias reais completas. Essas instâncias foram utilizadas para gerar os conjuntos de 150 a 4800 ordens de manutenção para comparar os resultados das heurísticas propostas;
- Estudo e implementação de novos algoritmos utilizando processamento paralelo;
- Proposição e teste de outras representações da solução;
- Consideração do fato de que as equipes podem executar uma mesma ordem de manutenção com tempos diferentes. Apesar de a empresa em estudo considerar tempos iguais, a coleta, armazenamento e uso dessas informações pode contribuir para melhorar a assertividade do mapa de 52 semanas;
- Desenvolvimento de uma interface para coleta das informações e geração do mapa de 52 semanas;
- Pesquisa e coleta de dados de outras empresas para adequação do modelo e dos algoritmos heurísticos propostos;

- Discussão e proposição de novas funções objetivo;
- Estudo e implementação de penalizações pela não execução das ordens de manutenção em função de estimativas de perdas financeiras.

7.1 Publicações

A seguir são listadas as duas produções geradas até então com os resultados intermediários obtidos durante a execução deste trabalho:

1. **Título**: A Mixed-Integer Linear Programming Model and a Simulated Annealing Algorithm for the LongTerm Preventive Maintenance Scheduling Problem
   Autores: Roberto Dias Aquino, Jonatas Batista Costa das Chagas e Marcone Jamilson Freitas Souza
   Evento: International Conference on Intelligent Systems Design and Applications - ISDA 2017
   Local: Delhi, India
   Qualis CC: B1
   DOI: 10.1007/978-3-319-76348-4_150
   Período: 14 a 16 de dezembro de 2017

2. **Título**: A Variable Neighborhood Search Algorithm for the Long-Term Preventive Maintenance Scheduling Problem
   Autores: Roberto Dias Aquino, Jonatas Batista Costa das Chagas e Marcone Jamilson Freitas Souza
   Evento: 20th International Conference on Enterprise Information Systems (ICEIS)
   Local: Funchal, Portugal
   Qualis CC: B2
   DOI: 10.5220/0006689703030310
   Período: 21 a 24 de março de 2018

Referências Bibliográficas

Adhikary, D. D., Bose, G. K., Jana, D. K., Bose, D., and Mitra, S. (2016). Availability and cost-centered preventive maintenance scheduling of continuous operating series systems using multi-objective genetic algorithm: a case study. *Quality Engineering*, 28(3):352–357.

Agustiady, T. and Cudney, E. (2015). *Total Productive Maintenance: Strategies and Implementation Guide*. CRC Press.

Aquino, R. D., Chagas, J. B. C., and Souza, M. J. F. (2017). Mixed-integer linear programming model and a simulated annealing algorithm for the longterm preventive maintenance scheduling problem. In *International Conference on Intelligent Systems Design and Applications*.

Aquino, R. D., Chagas, J. B. C., and Souza, M. J. F. (2018). A variable neighborhood search algorithm for the long-term preventive maintenance scheduling problem. In 20^{th} *International Conference on Enterprise Information System*.

Barlow, R. and Hunter, L. (1960). Optimum preventive maintenance policies. *Operations Research*, 8(1):90–100.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Brucker, P. (2007). *Scheduling algorithms*, volume 3. Springer.

Cordone, R. and Lulli, G. (2016). Multimode extensions of combinatorial optimization problems. *Electronic Notes in Discrete Mathematics*, 55:17–20.

Faria, H, D., Resende, M. G. C., and Ernst, D. (2017). A biased random key genetic algorithm applied to the electric distribution network reconfiguration problem. *Journal of Heuristics*, 23(6):533–550.

Ferreira, K. M. and de Queiroz, T. A. (2018). Two effective simulated annealing algorithms for the location-routing problem. *Appl. Soft Comput.*, 70:389–422.

Frinhani, R. d. M. D., Lacerda, R. M., Silva, R. M. A., and Mateus, G. R. (2015). Brkga para auto-parametrizaÇÃo do grasp com path-relinking no agrupamento de dados. *XLVII Simpósio Brasileiro de Pesquisa Operacional*.

Gonçalves, J. F. and Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Hansen, P. and Mladenović, N. (2014). Variable neighborhood search. In *Search methodologies*, pages 313–337. Springer.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI. second edition, 1992.

Jonge, B. (2017). *Maintenance Optimization based on Mathematical Modeling.* PhD thesis, University of Groningen.

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

Lafraia, J. (2001). *Manual de confiabilidade, mantenabilidade e disponibilidade.* Qualitymark.

Lowry, R. (2018). Concepts applications of inferential statistics. Disponnível em: `http://vassarstats.net/textbook/`. Último acesso em: 06 Nov. 2018.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Mainieri, G. B. (2014). *Meta-heurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido.* PhD thesis, Escola Politécnica da Universidade de São Paulo.

Martí, R., Resende, M. G. C., and Ribeiro, C. C. (2013). Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, 226:1–8.

Martinez, C., Loiseau, I., Resende, M. G. C., and Rodriguez, S. (2011). BRKGA algorithm for the capacitated arc routing problem. *Electr. Notes Theor. Comput. Sci.*, 281:69–83.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Moreira, D. A. (2012). *Administração da produção e operações.* Cengage Learning.

Moubray, J. (1997). *Reliability centered maintenance.* Industrial Press Inc.

Márquez, A. C., Díaz, V. G.-P., and Fernández, J. F. G. (2018). *Advanced Maintenance Modelling for Asset Manegement: Techniques and Methods for Complex Industrial Systems.* Springer.

Neri, F. and Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1 – 14.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall international series in industrial and systems engineering. Springer.

Pinto, A. and Nascif, J. (2009). *Manutenção: função estratégica*. Qualitymark.

Pohlert, T. (2014). *The Pairwise Multiple Comparison of Mean Ranks Package (PMCMR)*. R package.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Saraiva, J. T., Pereira, M. L., Mendes, V. T., and Sousa, J. C. (2011). A simulated annealing based approach to solve the generator maintenance scheduling problem. *Electric Power Systems Research*, 81(7):1283–1291.

Sharma, A., Yadava, G., and Deshmukh, S. (2011). A literature review and future perspectives on maintenance optimization. *Journal of Quality in Maintenance Engineering*, 17(1):5–25.

Simões, J. M., Gomes, C. F., and Yasin, M. M. (2011). A literature review of maintenance performance measurement: A conceptual framework and directions for future research. *Journal of Quality in Maintenance Engineering*, 17(2):116–137.

Toso, R. and Resende, M. G. C. (2011). A c++ application programming interface for biased random-key genetic algorithms. Disponível em: <`http://mauricio.resende.info/src/brkgaAPI/`>. Último acesso em: 06 Nov. 2018.

Wendy, T. and Victor, A. (2018). *Mean Time Between Failure: Explanation and Standards*.

Yamayee, Z., Sidenblad, K., and Yoshimura, M. (1983). A computationally efficient optimal maintenance scheduling method. *IEEE Transactions on Power Apparatus and Systems*, 102(2):330–338.

Yao, X., Fernández-Gaucherand, E., Fu, M. C., and Marcus, S. I. (2004). Optimal preventive maintenance scheduling in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 17(3):345–356.

Zudio, A., da Silva Costa, D. H., Masquio, B. P., Coelho, I. M., and Pinto, P. E. D. (2018). BRKGA/VND hybrid algorithm for the classic three-dimensional bin packing problem. *Electronic Notes in Discrete Mathematics*, 66:175–182.

Apêndice A

Comparação entre os algoritmos SA, BRKGA e BRKMA

As tabelas nas próximas páginas mostram os resultados dos experimentos computacionais entre os algoritmos SA, BRKGA e BRKMA.

Na Tabela A.1, as primeiras quatro colunas descrevem a instância, sendo que as colunas *ID*, *n*, *m* e *Q*, informam, respectivamente, o número de identificação da instância, o número de ordens de manutenção preventivas, o número de equipes de trabalho disponíveis e o número de equipamentos. A coluna *Media* informa o resultado médio; a *Melhor*, o melhor resultado; a *Desvio*, o desvio percentual entre a média dos resultados e o melhor resultado (conforme descrito na Equação 6.1).

Tabela A.1: Comparação SA x BRKGA x BRKMA

Instância				**SA**			**BRKGA**			**BRKMA**		
ID	**n**	**m**	**Q**	**Média**	**Melhor**	**Desvio**	**Média**	**Melhor**	**Desvio**	**Média**	**Melhor**	**Desvio**
1	20	2	2	434	**434**	0.0	434	**434**	0.0	434	**434**	0.0
2	20	3	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
3	20	4	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
4	20	5	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
5	20	10	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
6	30	2	2	434	**434**	0.0	434	**434**	0.0	434	**434**	0.0
7	30	3	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
8	30	4	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
9	30	5	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
10	30	10	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
11	40	2	2	650	650	49.8	520	**434**	19.8	607	**434**	39.9
12	40	3	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0

Continua na próxima página

Tabela A.1 – (continuação) Comparação SA x BRKGA x BRKMA

Instância				SA			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
13	40	4	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
14	40	5	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
15	40	10	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
16	60	2	2	606.8	**434**	39.8	650	650	49.8	650	650	49.8
17	60	3	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
18	60	4	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
19	60	5	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
20	60	10	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
21	80	2	2	1082	**1082**	0.0	1082	**1082**	0.0	1082	**1082**	0.0
22	80	3	2	219	**219**	0.0	262	**219**	19.6	219	**219**	0.0
23	80	4	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
24	80	5	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
25	80	10	2	219	**219**	0.0	219	**219**	0.0	219	**219**	0.0
26	20	3	3	579	**579**	0.0	579	**579**	0.0	579	**579**	0.0
27	20	4	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
28	20	5	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
29	20	6	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
30	20	12	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
31	30	3	3	579	**579**	0.0	579	**579**	0.0	579	**579**	0.0
32	30	4	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
33	30	5	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
34	30	6	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
35	30	12	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
36	40	3	3	679.8	**579**	17.4	694	**579**	19.9	694	**579**	19.9
37	40	4	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
38	40	5	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
39	40	6	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
40	40	12	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
41	60	3	3	1702.2	1659	18.0	1616	**1443**	12.0	1573	**1443**	9.0
42	60	4	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
43	60	5	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
44	60	6	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
45	60	12	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
46	80	3	3	1889.4	1803	30.9	1659	1659	15.0	1659	**1443**	15.0
47	80	4	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
48	80	5	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
49	80	6	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
50	80	12	3	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
51	20	3	4	723	**723**	0.0	723	**723**	0.0	723	**723**	0.0
52	20	4	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
53	20	5	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
54	20	6	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
55	20	12	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
56	30	3	4	723	**723**	0.0	723	**723**	0.0	723	**723**	0.0
57	30	4	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0

Continua na próxima página

Tabela A.1 – (continuação) Comparação SA x BRKGA x BRKMA

Instância				SA			BRKGA			BRKMA		
ID	n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
58	30	5	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
59	30	6	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
60	30	12	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
61	40	3	4	967.8	**867**	11.6	939	**867**	8.3	953	**867**	9.9
62	40	4	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
63	40	5	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
64	40	6	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
65	40	12	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
66	60	3	4	2307	2307	10.3	2134	**2091**	2.1	2149	**2091**	2.8
67	60	4	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
68	60	5	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
69	60	6	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
70	60	12	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
71	80	3	4	3214.2	3099	8.8	3056	**2955**	3.4	3099	**2955**	4.9
72	80	4	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
73	80	5	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
74	80	6	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
75	80	12	4	220	**220**	0.0	220	**220**	0.0	220	**220**	0.0
76	20	4	5	724	**724**	0.0	724	**724**	0.0	724	**724**	0.0
77	20	5	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
78	20	6	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
79	20	7	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
80	20	14	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
81	30	4	5	724	**724**	0.0	724	**724**	0.0	724	**724**	0.0
82	30	5	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
83	30	6	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
84	30	7	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
85	30	14	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
86	40	4	5	868	**868**	0.0	868	**868**	0.0	882	**868**	1.6
87	40	5	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
88	40	6	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
89	40	7	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
90	40	14	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
91	60	4	5	1588	**1588**	0.0	1631	**1588**	2.7	1602	**1588**	0.9
92	60	5	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
93	60	6	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
94	60	7	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
95	60	14	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
96	80	4	5	3244	3244	7.1	3086	**3028**	1.9	3114	**3028**	2.8
97	80	5	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
98	80	6	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
99	80	7	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0
100	80	14	5	221	**221**	0.0	221	**221**	0.0	221	**221**	0.0