



**Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração (PROFICAM)**
**Escola de Minas, Universidade Federal de Ouro Preto (UFOP)**
**Associação Instituto Tecnológico Vale (ITV)**

**Dissertação**

# PLANEJAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA POR MEIO DE SIMHEURÍSTICA

**Diego Gomes Coelho**

**Ouro Preto**
**Minas Gerais, Brasil**
**2023**

**Diego Gomes Coelho**

# PLANEJAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA POR MEIO DE SIMHEURÍSTICA

Dissertação apresentada ao Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração da Universidade Federal de Ouro Preto e do Instituto Tecnológico Vale, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Controle e Automação.

Orientador: Prof. Marcone Jamilson Freitas Souza, D.Sc.
Coorientador: Prof. Luciano Perdigão Cota, D.Sc.

Ouro Preto
2023

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

C672p Coelho, Diego Gomes.
Planejamento de ordens de manutenção preventiva por meio de SIMHEURÍSTICA. [manuscrito] / Diego Gomes Coelho. - 2023.
51 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Coorientador: Prof. Dr. Luciano Perdigão Cota.
Dissertação (Mestrado Profissional). Universidade Federal de Ouro Preto. Programa de Mestrado Profissional em Instrumentação, Controle e Automação de Processos de Mineração. Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração.
Área de Concentração: Engenharia de Controle e Automação de Processos Minerais.

1. Planejamento. 2. Ordem de Serviço de Manutenção (OS). 3. Otimização estrutural. 4. Programação heurística. I. Souza, Marcone Jamilson Freitas. II. Cota, Luciano Perdigão. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5:622.2

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

**FOLHA DE APROVAÇÃO**

**Diego Gomes Coelho**

**Planejamento de ordens de manutenção preventiva por meio de simheurística**

Dissertação apresentada ao Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração (PROFICAM), Convênio Universidade Federal de Ouro Preto / Associação Instituto Tecnológico Vale (UFOP/ITV), como requisito parcial para obtenção do título de Mestre em Engenharia de Controle e Automação na área de concentração em Instrumentação, Controle e Automação de Processos de Mineração.

Aprovada em 26 de maio de 2023.

Membros da banca

Dr. Marcone Jamilson Freitas Souza - Orientador (Universidade Federal de Ouro Preto)
Dr. Luciano Perdigão Cota - Coorientador (Instituto Tecnológico Vale)
Dra. Simone de Lima Martins - (Universidade Federal Fluminense)
Dr. George Henrique Godim da Fonseca - (Universidade Federal de Ouro Preto)

[Marcone Jamilson Freitas Souza], orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 10/08/2023

Documento assinado eletronicamente por **Marcone Jamilson Freitas Souza**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 10/08/2023, às 19:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0571955** e o código CRC **3453E005**.



R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

*Dedico esse trabalho à minha filha Catarina, por ter sido a catalisadora de tantas mudanças positivas na minha vida desde sua chegada.*

# Agradecimentos

Agradeço a todos que estiveram envolvidos na elaboração desta dissertação, direta ou indiretamente. A sua contribuição foi fundamental para a sua concretização.

A minha família que sempre foi meu porto seguro, meus pais Luciano e Eva e meus irmãos Lorene e Douglas. Obrigado pelo amor e apoio incondicionais.

A minha companheira e esposa Elisa que percorreu comigo cada etapa dessa jornada me dando apoio, incentivando e acreditando em minha capacidade quando eu mesmo não acreditava. Obrigado por tudo e por tanto, sem você esse trabalho não seria possível.

Aos meus sogros Sr. Luiz e Sra. Tereza pelo carinho e por terem proporcionado uma rede de apoio tão necessária.

Aos meus orientadores, Professor Doutor Marcone Jamilson Freitas Souza e Professor Doutor Luciano Perdigão Cota, pela disponibilidade, atenção e grandes contribuições na elaboração deste trabalho.

Agradeço também a empresa Stellantis por conceder a oportunidade de fazer esse mestrado. Em especial à equipe de Auditoria e Compliance e à equipe de Manutenção da planta de motores em Betim. Deixo aqui meu agradecimento aos gestores Vanderlei Oliveira e Milton Moreira pelo grande apoio no início desse processo.

À Universidade Federal de Ouro Preto por propiciar um ensino público, gratuito e de qualidade.

Ao ITV por propiciar esse programa através da parceria com a UFOP.

Aos meus colegas de turma do mestrado, pela boa convivência e apoio mútuo.

**Resumo**

Resumo da Dissertação apresentada ao Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PLANEJAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA POR MEIO DE SIMHEURÍSTICA

Diego Gomes Coelho

Maio/2023

Orientadores: Marcone Jamilson Freitas Souza
Luciano Perdigão Cota

Este trabalho aborda o problema de alocação de tarefas de manutenção preventiva em um planejamento de 52 semanas. Dado um conjunto de tarefas de manutenção a serem realizadas em um conjunto de máquinas, um conjunto de equipes e um horizonte de planejamento, o problema consiste em designar cada tarefa a uma equipe em um determinado instante do horizonte de planejamento visando a minimizar o número de equipes necessárias e maximizar o número de tarefas não executadas. Para tratá-lo, inicialmente foi desenvolvido um algoritmo construtivo capaz de testar automaticamente 384 regras de construção e retornar a melhor regra para cada instância. Comparado com a literatura, esse algoritmo construtivo gerou soluções de boa qualidade em tempo computacional muito menor que os algoritmos meta-heurísticos existentes. Foi desenvolvido também o algoritmo meta-heurístico *Iterated Local Search* (ILS) para aprimorar as soluções geradas pelo método construtivo. O algoritmo ILS atingiu o melhor resultado em 97% das instâncias avaliadas. No entanto, tanto o método ILS quanto os demais métodos da literatura para o problema são determinísticos e não levam em consideração as incertezas que podem ocorrer durante a execução de uma tarefa, o que pode resultar em um planejamento insuficiente com um tempo de execução total mais longo do que o previsto. Para contornar essa situação, foi proposto um algoritmo simheurístico, nomeado SIM-ILS. Embora ele apresente um custo levemente superior ao método ILS determinístico, o SIM-ILS é capaz de captar as incertezas da operação de manutenção, tornando suas soluções mais aderentes ao ambiente industrial.

**Palavras-chave:** Planejamento, Ordens de Manutenção, Otimização, Heurísticas.

**Macrotema:** Logística; **Linha de Pesquisa:** Tecnologias da Informação, Comunicação e Automação Industrial; **Tema:** Melhoria de Procedimentos de Manutenção.

**Abstract**

Abstract of Dissertation presented to the Graduate Program on Instrumentation, Control and Automation of Mining Process as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PREVENTIVE MAINTENANCE ORDER PLANNING THROUGH SIMHEURISTIC

Diego Gomes Coelho

May/2023

Advisors: Marcone Jamilson Freitas Souza
Luciano Perdigão Cota

This work addresses the problem of assigning preventive maintenance jobs in a 52-week planning horizon. Given a set of maintenance jobs to be performed in a set of machines, a set of work teams, and a planning horizon, the problem consists in assigning each job to a team in a given instant of the planning horizon, aiming to minimize the number of necessary teams and maximize the number of jobs not performed. Initially, we developed a constructive algorithm capable of automatically testing 384 construction rules and returning the best rule for each instance. Compared to the literature, this constructive algorithm generated good-quality solutions in much less computational time than existing meta-heuristic algorithms. We also developed an Iterated Local Search (ILS) algorithm to improve the solutions generated by the constructive method. The ILS algorithm achieved the best result in 97% of the evaluated instances. However, both the ILS method and other methods in the literature for the problem are deterministic and do not consider the uncertainties that may occur during the execution of a job, which can result in insufficient planning with a total execution time longer than expected. This work proposes a simheuristic algorithm named SIM-ILS, to overcome this situation. Although it presents a cost slightly higher than the deterministic ILS method, the SIM-ILS is able to capture the uncertainties of the maintenance operation, making its solutions more adherent to the industrial environment.

**Keywords:** Scheduling, Maintenance Order, Optimization, Heuristics.

**Macrotheme:** Logistics; **Research Line:** Information Technologies, Communication and Industrial Automation; **Theme:** Maintenance Planning and Control.

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

**ALNS** *Adaptive Large Neighborhood Search*

**BRKGA** *Biased Random-Key Genetic Algorithm*

**BRKMA** *Biased Random-Key Memetic Algorithm*

**CMMS** *Computerized Maintenance Management System*

**ILS** *Iterated Local Search*

**Irace** *Iterated Racing for Automatic Algorithm Configuration*

**JPA** *Job Positioning Algorithm*

**MS** *Multi-Start*

**SA** *Simulated Annealing*

**VNS** *Variable Neighborhood Search*

# Sumário

# 1. Introdução

Este capítulo está organizado como segue. Na Seção 1.1, contextualiza-se o problema objeto de pesquisa deste trabalho, descrevendo o cenário em que o problema ocorre e os principais desafios envolvidos. Na Seção 1.2 discute-se a motivação para a realização deste estudo, destacando a importância da pesquisa e o que a torna relevante para a comunidade científica e a indústria. A Seção 1.3 apresenta os objetivos do trabalho, descrevendo o que se pretende alcançar por meio desta pesquisa. Por fim, a Seção 1.4 apresenta a organização deste trabalho.

## 1.1. Contextualização

O planejamento e controle da manutenção são cruciais para o setor industrial, pois garantem a disponibilidade dos ativos com o menor custo possível. Para minimizar a probabilidade de falhas nesses ativos, é frequentemente utilizada a manutenção preventiva. As manutenções preventivas são intervenções programadas de acordo com as recomendações do fabricante ou experiência das equipes de manutenção em ativos (VIANA, 2006).

Para gerenciar as tarefas de manutenção, as indústrias utilizam um Sistema Computadorizado de Gestão da Manutenção, (do inglês *Computerized Maintenance Management System* (CMMS)), que centraliza as informações e facilita o controle das operações de manutenção. Nele são cadastrados os planos de manutenção dos ativos, contendo as informações relativas às manutenções preventivas que devem ser executadas, sua periodicidade, sua duração e o tipo de habilidade requerida pela equipe para executá-las.

A partir dessas informações do CMMS é possível gerar um planejamento de longo prazo, conhecido como mapa de 52 semanas. Esse mapa é utilizado como uma ferramenta de planejamento e permite que a empresa possa se antecipar às necessidades de material e mão de obra requeridas para execução das manutenções preventivas. A importância desse mapa é garantir que todas as tarefas de manutenção preventiva sejam realizadas no prazo correto, reduzindo a probabilidade de falhas nos equipamentos e aumentando a vida útil dos ativos. Além disso, ele permite que as equipes de manutenção possam ser planejadas com antecedência, garantindo que as habilidades necessárias para cada atividade estejam disponíveis quando necessário.

No entanto, a elaboração do mapa de 52 semanas pode ser uma tarefa desafiadora. Primeiro, nem todas as tarefas de manutenção preventiva são iguais em termos de complexidade e tempo de execução, o que pode levar a uma alocação inadequada de recursos e atrasos na realização das tarefas mais complexas. Outro ponto é que ele é gerado com base em suposições estáticas e não considera as mudanças dinâmicas na demanda de produção, mudanças na disponibilidade de recursos e outras variáveis que podem afetar o planejamento de manutenção. Isso pode levar a uma falta de flexibilidade no planejamento e dificuldade em lidar com imprevistos. Levando a um aumento dos custos de manutenção, devido à necessidade de horas extras e reagendamento de tarefas não realizadas, e redução da disponibilidade dos ativos, devido à falta

de manutenção adequada. Por isso, é importante o desenvolvimento de ferramentas que auxiliem no planejamento, que possam tratar todas as restrições e se adaptar a diferentes cenários industriais possibilitando um controle eficiente das tarefas de manutenção preventiva.

## 1.2. Motivação

Aquino *et al.* (2019) introduziram o problema de alocação de tarefas de manutenção preventiva de máquinas industriais às equipes de trabalho para um planejamento de longo prazo, conhecido como mapa de 52 semanas. Os autores desenvolveram um modelo matemático e algoritmos meta-heurísticos para gerar um planejamento mais eficiente quando comparado com o feito pela equipe de planejamento da empresa. No entanto, detectamos pelo menos cinco oportunidades de pesquisa, as quais motivaram o desenvolvimento do presente trabalho.

A primeira motivação foi o tempo de execução alto dos algoritmos propostos por Aquino *et al.* (2019) para gerar uma boa solução para a instância real do problema. Como a manutenção é uma atividade sabidamente dinâmica, muitas vezes uma manutenção programada pode, por vários motivos, não ser realizada, exigindo uma reprogramação das atividades ainda não realizadas. Desta forma, é importante ter à disposição um algoritmo que requeira um menor tempo de processamento para realizar essa reprogramação.

A segunda motivação foi o fato de que os autores partem de uma solução representada por uma sequência aleatória de ordens de manutenção. Assim, eles não exploram características específicas do problema. Explorar tais características pode reduzir o espaço de soluções do problema e, indiretamente, o tempo de processamento do algoritmo.

A terceira motivação foi o fato de o algoritmo de posicionamento de tarefas de Aquino *et al.* (2019) não ter sido explorado com outras alternativas de alocação. No algoritmo desses autores, cada tarefa de manutenção é alocada a partir do instante inicial de sua janela de execução. Começar a alocação pelo instante final é uma estratégia que não foi analisada e que pode gerar um resultado melhor.

O desenvolvimento de um algoritmo meta-heurístico baseado em *Iterated Local Search* (ILS) para tratar o problema é também outra motivação. De nosso conhecimento, apesar do sucesso do ILS no tratamento de vários outros problemas de natureza combinatória, ele foi aplicado apenas em um problema similar ao tratado neste trabalho, porém menos abrangente.

A quinta motivação está relacionada ao aspecto estocástico do tempo de execução das tarefas de manutenção. O tempo planejado para executar uma tarefa pode ser diferente daquele realmente gasto na sua execução. Por exemplo, muitas vezes, as tarefas de manutenção envolvem desmontagem, reparo e montagem de peças, o que pode levar mais tempo (dependendo da equipe e da condição de operação) do que o previsto inicialmente. Isso pode ocasionar atrasos no cronograma de manutenção, resultando em equipamentos que não estarão disponíveis quando necessário ou que sofram falhas prematuras. Para tratar esses aspectos estocásticos do problema, o uso de simheurística é uma alternativa ainda não explorada para este problema. O

sucesso desta técnica no tratamento de outros problemas, faz-nos acreditar na proposição de uma solução que seja mais aderente ao planejamento do que aquela realizada por um algoritmo que considera apenas tempos determinísticos de execução de tarefas.

## 1.3. Objetivos

O presente trabalho tem como objetivo geral implementar uma ferramenta de suporte para o planejamento e controle da manutenção industrial baseado em uma simheurística. Para tanto, os seguintes objetivos específicos foram considerados:

- Realização de uma revisão bibliográfica sobre otimização do planejamento da manutenção e programação de tarefas, levando em consideração as possíveis restrições que são inerentes ao problema tratado;
- Proposição de um novo algoritmo de posicionamento de ordens de manutenção dada uma sequência de manutenção;
- Desenvolvimento de algoritmos heurísticos construtivos para tratar o problema;
- Desenvolvimento de um algoritmo meta-heurístico de busca local para aprimorar as soluções geradas pelos algoritmos construtivos;
- Desenvolvimento de um algoritmo meta-heurístico de busca local integrado a um simulador para considerar aspectos estocásticos da duração das ordens de manutenção;
- Desenvolvimento de uma ferramenta de visualização do planejamento das ordens de manutenção para auxiliar o controle da manutenção;
- Validação da abordagem proposta utilizando dados reais;
- Comparação dos resultados dos algoritmos determinístico e estocástico desenvolvidos com os da literatura.

## 1.4. Organização do Trabalho

Os capítulos seguintes deste trabalho estão estruturados conforme descrição a seguir:

- **Capítulo 2 - Revisão Bibliográfica:** apresenta uma revisão bibliográfica sobre o tema de otimização na manutenção;
- **Capítulo 3 - Caracterização do Problema:** apresenta e detalha o problema tratado;

- **Capítulo 4 - Algoritmos Propostos:** descreve como uma solução é representada e avaliada, além de apresentar os algoritmos de posicionamento das ordens de manutenção, os algoritmos heurísticos construtivos e os algoritmos meta-heurísticos propostos para tratar as versões determinística e estocástico do problema.
- **Capítulo 5 - Experimentos Computacionais:** apresenta e detalha os experimentos computacionais realizados;
- **Capítulo 6 - Conclusão:** traz as considerações finais e indica propostas de continuidade deste trabalho.

# 2. Revisão Bibliográfica

A gestão e melhoria da manutenção é um tema recorrente na literatura, como apresentado em Simões *et al.* (2011) e Sharma *et al.* (2011). Em grande parte dos trabalhos, a abordagem se dá na otimização do uso de recursos e na redução do custo de manutenção. Contudo, o problema de alocação de ordens de manutenção preventiva ainda é pouco explorado na literatura.

Saraiva *et al.* (2011) utilizam uma abordagem com a meta-heurística *Simulated Annealing* (SA) (KIRKPATRICK *et al.*, 1983) para a programação de manutenção em geradores de energia e o estudo foi realizado em um sistema com 29 grupos geradores. O objetivo de minimizar o custo operacional ao longo do período de agendamento e uma penalidade por energia não fornecida. O artigo detalha as restrições do problema e inclui variáveis binárias para indicar quando um gerador está em manutenção em uma determinada semana. No estudo em questão, é discutido o planejamento de longo prazo, entretanto, não se aprofunda na aplicação de algoritmos construtivos para a formação de soluções iniciais e tampouco utiliza simheurísticas.

Em Almakhlafi e Knowles (2015) é proposto um algoritmo meta-heurístico baseado no método *Iterated Local Search* (ILS) (LOURENÇO *et al.*, 2001) para resolver o problema de programação de manutenção preventiva em geradores de energia. O trabalho apresenta extensões do método ILS fazendo combinações com outros métodos, para obter uma melhor performance na busca pela solução do problema. O objetivo é aumentar a confiabilidade do sistema de fornecimento de energia. O trabalho aborda o planejamento de longo prazo, porem não explora o uso de algoritmos construtivos para construção de soluções iniciais e não faz uso de simheuristicas.

Aquino *et al.* (2018) e Aquino *et al.* (2019) tratam o problema de alocação de ordens de manutenção preventiva de máquinas industriais às equipes de trabalho disponíveis ao longo de um planejamento de 52 semanas. Neste problema, o objetivo é alocar a maior quantidade de ordens de manutenção utilizando a menor quantidade de equipes de trabalho. Trata-se de um problema real encontrado em indústrias de mineração. Para resolver o problema são desenvolvidos um modelo de programação linear inteira mista e métodos heurísticos. Os métodos heurísticos são baseados nas meta-heurísticas *Simulated Annealing* (SA) (KIRKPATRICK *et al.*, 1983), *Variable Neighborhood Search* (VNS) (MLADENOVIĆ e HANSEN, 1997), *Multi-Start* (MS) (MARTÍ *et al.*, 2013) e o *Biased Random-Key Genetic Algorithm* (BRKGA) (MARTINEZ *et al.*, 2011) e *Biased Random-Key Memetic Algorithm* (BRKMA) (NERI e COTTA, 2012). Dentre estes, o MS e o BRMKA obtiveram os melhores resultados. Observou-se que as heurísticas construtivas e o algoritmo responsável pelo posicionamento de tarefas nos algoritmos meta-heurísticos propostos, bem como o uso de técnicas de simheurísticas, não foram explorados.

Woller e Kulich (2021) abordam um problema de planejamento de ordens de manutenção em redes de transmissão de energia. Neste problema o objetivo é maximizar o número de manutenções a serem realizadas em janelas de paradas programadas. Para tratá-lo, eles propõem

um algoritmo baseado na meta-heurística *Adaptive Large Neighborhood Search* (ALNS) (PISINGER e ROPKE, 2010) que é adaptada ao problema de programação de manutenção de transmissão. O artigo apresenta uma lista de heurísticas de destruição e reparação e operadores de busca local para refinamento adicional de um cronograma reorganizado pelas heurísticas. Os resultados mostram que o método é capaz de produzir soluções consistentes com as melhores soluções conhecidas para todos os casos de teste. Embora o artigo tenha apresentado um método baseado em meta-heurísticas para resolver o problema de programação de manutenção de transmissão, é importante ressaltar que ele não abordou o planejamento de longo prazo e não explorou o uso de algoritmos construtivos ou simheurísticas.

Viveros *et al.* (2021) propõem um modelo matemático de programação linear inteira mista para um outro problema de alocação de ordens de manutenção em janelas de oportunidade em uma planta de tratamento de água. O objetivo é minimizar a indisponibilidade do sistema devido a paradas de manutenção. Para isso, propõe uma estratégia de agrupamento oportunista com tolerâncias de janelas de tempo e tempos de execução não negligenciáveis, visando otimizar os planos de manutenção preventiva. Neste trabalho, é abordado o planejamento de curto prazo sem o uso de meta-heurísticas ou simheuristicas.

Em Mena *et al.* (2021), os autores apresentam uma metodologia para o planejamento de manutenção preventiva de uma única máquina a médio prazo, utilizando um modelo matemático de programação inteira mista que determina os instantes de execução das atividades de manutenção, considerando tolerâncias de janela de tempo para avançar ou adiar as execuções e, assim, maximizar o agrupamento de execuções e minimizar o número de paradas necessárias para cumprir uma política de manutenção preventiva. O trabalho tem como foco o planejamento de médio prazo em uma única máquina e não utiliza algoritmos construtivos, meta-heurísticas ou simheurísticas.

A Tabela 2.1 agrega as principais características dos trabalhos revisados indicando qual foi o horizonte de planejamento, os objetivos e os métodos de solução propostos para cada um deles.

Tabela 2.1: Principais características presentes em alguns estudos da literatura.

| Trabalho | Horizonte de planejamento | | | Objetivos | | | | | Métodos de solução | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Curto | Médio | Longo | [1] | [2] | [3] | [4] | [5] | Exato | Construtivo | Meta-heurística | Simheurística |
| Almakhlafi e Knowles (2015) | - | - | ✓ | - | ✓ | - | - | ✓ | - | - | ✓ | - |
| Saraiva *et al.* (2011) | - | - | ✓ | - | ✓ | ✓ | - | - | - | - | ✓ | - |
| Aquino *et al.* (2019) | - | - | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - | ✓ | - |
| Woller e Kulich (2021) | - | - | ✓ | ✓ | - | - | - | ✓ | - | - | ✓ | - |
| Viveros *et al.* (2021) | ✓ | - | - | ✓ | ✓ | - | ✓ | - | ✓ | - | - | - |
| Mena *et al.* (2021) | - | ✓ | - | ✓ | - | - | ✓ | - | ✓ | - | - | - |
| Nossa Proposta | - | - | ✓ | ✓ | ✓ | ✓ | - | - | - | ✓ | ✓ | ✓ |

**Legenda:**
[1]: Mais de uma equipe; [2]: Mais de um equipamento; [3]: Penalidade por não execução; [4]: Tempo de parada; [5]: Confiabilidade do sistema

Fonte: Compilação do autor.

A literatura sobre o uso de meta-heurísticas para problemas de manutenção aponta para a necessidade de soluções mais robustas diante dos desafios dinâmicos do ambiente industrial.

Nesse sentido, a revisão de literatura sobre simheurísticas se destaca por apresentar uma abordagem híbrida que combina a simulação do ambiente industrial com a otimização por meio de meta-heurísticas, permitindo uma maior adaptabilidade às mudanças imprevisíveis do ambiente. Em Juan *et al.* (2022), é apresentado um tutorial introdutório sobre o conceito de simheurística, que consiste na combinação de métodos de simulação e algoritmos de otimização meta-heurísticos para resolver problemas de otimização estocásticos em diversos setores, como manufatura, serviços, transporte, telecomunicações e seguros. Dados os aspectos estocásticos desses problemas e como muitos deles são de difícil resolução em instâncias de grande porte, a abordagem simheurística se torna uma alternativa eficiente para tratá-los. Na sequência são revisados alguns trabalhos que fizeram uso dessa técnica.

Guimarans *et al.* (2018) apresentam o problema de roteamento de veículos em duas dimensões com demandas compostas por conjuntos de itens não empilháveis e tempos de viagem estocásticos, além de propor um algoritmo simheurístico híbrido para resolvê-lo. O algoritmo combina simulação de Monte Carlo, busca local iterada e heurísticas de roteamento e empacotamento com aleatorização enviesada. O método foi testado em um *benchmark* extenso que estende o problema determinístico com carregamento irrestrito e não orientado.

Em Santos *et al.* (2020), os autores buscam determinar a quantidade adequada de equipamentos ativos em um circuito de britagem de minério para garantir a eficiência e a taxa de produção. Para otimizar a quantidade de equipamentos ativos, um sistema de suporte à decisão baseado em uma abordagem simheurística é proposto. Nesse sistema, um modelo de planta simulada é usado para avaliar a taxa de produção. Experimentos computacionais foram realizados com cenários reais de produção em uma mina brasileira e os resultados mostraram que as soluções simheurísticas geram uma taxa de produção maior e resultam em menor consumo de energia, com aumento de até 9% na produção e redução do consumo de energia em até 59%.

Keenan *et al.* (2021) tratam o Problema de Roteamento de Arcos com Capacidade de Tempo (TCARP) com demandas estocásticas. O objetivo é encontrar um plano de roteamento que minimize o tempo total esperado necessário para atender todos os clientes, considerando a capacidade de tempo do veículo e a variabilidade das demandas e dos tempos de serviço. Para resolver esse problema, propõe-se um algoritmo simheurístico de oscilação estratégica. Essa variabilidade afeta os tempos de serviço, que também se tornam variáveis aleatórias. O principal objetivo é encontrar um plano de roteamento que minimize o tempo total esperado necessário para atender todos os clientes. O desempenho do algoritmo é testado em experimentos numéricos.

Kizys *et al.* (2022) tratam o problema de otimização de carteira, que busca minimizar o risco para um retorno esperado alocando pesos aos ativos. O problema se torna NP-difícil quando o conjunto de ativos investíveis cresce e restrições adicionais são impostas. O artigo propõe um algoritmo simheurístico que integra um algoritmo meta-heurístico com simulação de Monte Carlo para lidar com retornos estocásticos e covariâncias ruidosas modeladas como variáveis aleatórias. Os experimentos computacionais mostram as vantagens dessa abordagem

e analisam como as soluções mudam em resposta a diferentes níveis de aleatoriedade e retornos mínimos requeridos.

# 3. Caracterização do Problema

A seguir, descrevem-se os conjuntos, parâmetros de entrada, índices, as variáveis de decisão e o modelo de programação linear inteira mista proposto por Aquino *et al.* (2019) para representar o problema em estudo.

- Conjuntos:

$\mathscr{E}$: Conjunto de máquinas industriais nas quais as ordens de manutenção serão executadas. $\mathscr{E} = \{1,2,\ldots,Q\}$;

$\mathscr{T}$: Conjunto de ordens de manutenção preventiva. $\mathscr{T} = \{1,2,\cdots,N\}$;

$\mathscr{W}$: Conjunto de equipes de manutenção disponíveis para a realização das ordens de manutenção. $\mathscr{W} = \{1,2,\cdots,M\}$;

- Parâmetros de entrada:

$Q$ : Número de máquinas industriais;
$N$ : Número de ordens de manutenção;
$\mathcal{W}_i$ : Habilidade requerida para execução da ordem de manutenção $i \in \mathscr{T}$;
$E_i$ : Início da janela de tempo de execução da ordem de manutenção $i \in \mathscr{T}$;
$L_i$ : Início da janela de tempo de execução da ordem de manutenção $i \in \mathscr{T}$;
$P_i$ : Duração da ordem de manutenção $i \in \mathscr{T}$;
$\omega_i$ : Penalidade por não execução da ordem de manutenção $i \in \mathscr{T}$;
$M$ : Número de equipes de trabalho;
$H_k$ : Instante máximo de disponibilidade da equipe de trabalho $k \in \mathscr{W}$;
$\mathcal{T}_k$ : Habilidade da equipe de trabalho $k \in \mathscr{W}$;

- Índices:

$i, j$ : Índices para ordens de manutenção;
$k$ : Índice para equipes de trabalho;

- Variáveis de decisão:

$x_{ij}^k$: 1 se a manutenção $i$ é executada imediatamente antes da ordem de manutenção $j$ pela equipe $k$; 0, caso contrário;

$y_{ik}$: 1 se a manutenção $i$ é executada pela equipe de trabalho $k$; 0, caso contrário;

$z_k$: 1 se a equipe de trabalho $k$ é utilizada; 0, caso contrário;

$c_{ik}$: instante de conclusão da manutenção $i$ quando ela é executada pela equipe de trabalho $k$;

$r_{ij}$: 1 se a manutenção $i$ é executada antes da manutenção $j$ e 0, caso contrário.

A seguir, apresenta-se a formulação de programação linear inteira mista para o problema, expressa pelas equações (3.1)-(3.17):

$$\min \sum_{k \in \mathscr{W}} z_k + \sum_{i \in \mathscr{T}} \omega_i \left( 1 - \sum_{k \in \mathcal{W}_i} y_{ik} \right) \tag{3.1}$$

$$\sum_{k \in \mathcal{W}_i} y_{ik} \leq 1 \qquad i \in \mathscr{T} \tag{3.2}$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{j\}} x_{ij}^k = y_{jk} \qquad j \in \mathscr{T}, k \in \mathcal{W}_j \tag{3.3}$$

$$\sum_{j \in \mathcal{T}_k} x_{0j}^k = z_k \qquad k \in \mathscr{W} \tag{3.4}$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{il}^k = \sum_{j \in \mathcal{T}_k \cup \{0\} \setminus \{l\}} x_{lj}^k \qquad k \in \mathscr{W}, l \in \mathcal{T}_k \tag{3.5}$$

$$c_{0k} = 0 \qquad k \in \mathscr{W} \tag{3.6}$$

$$c_{jk} \geq c_{ik} + P_j - M'_{ij}(1 - x_{ij}^k) \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \tag{3.7}$$

$$c_{ik} \geq (E_i + P_i) y_{ik} \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \tag{3.8}$$

$$c_{ik} \leq L_i \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \tag{3.9}$$

$$c_{jk'} \geq c_{ik} + P_j - M'_{ij}(1 - r_{ij}) \qquad k \in \mathscr{W}, k' \in \mathscr{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, \mid k \neq k', i < j, Q_i = Q_j \tag{3.10}$$

$$c_{jk'} \leq c_{ik} - P_i + M''_{ij} r_{ij} \qquad k \in \mathscr{W}, k' \in \mathscr{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, \mid k \neq k', i < j, Q_i = Q_j \tag{3.11}$$

$$c_{ik} \leq H_k \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \tag{3.12}$$

$$x_{ij}^k \in \{0,1\} \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \cup \{0\} \tag{3.13}$$

$$y_{ik} \in \{0,1\} \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \cup \{0\} \tag{3.14}$$

$$z_k \in \{0,1\} \qquad k \in \mathscr{W} \tag{3.15}$$

$$c_{ik} \geq 0 \qquad k \in \mathscr{W}, i \in \mathcal{T}_k \tag{3.16}$$

$$r_{ij} \in \{0,1\} \qquad \forall i \in \mathscr{T}, j \in \mathscr{T}, i < j, Q_i = Q_j \tag{3.17}$$

Nesta formulação foi definida uma tarefa fictícia (0), que precede imediatamente a primeira tarefa de manutenção preventiva e segue imediatamente a última tarefa executada de cada equipe de trabalho.

A função objetivo (3.1) busca reduzir o número de equipes de manutenção necessárias para executar o maior número possível de ordens de manutenção preventiva, através da minimização da aplicação de uma penalidade a cada tarefa não executada. As restrições (3.2) garantem que cada tarefa seja executada por no máximo uma equipe. As restrições (3.3) asseguram que se a equipe de trabalho $k$ executar a tarefa $j$, essa tarefa deve constar no agendamento de trabalho da equipe $k$. Para o conjunto de restrições da equação (3.4), se pelo menos uma tarefa for

programada para a equipe de trabalho $k$, essa equipe será utilizada. As restrições (3.5) asseguram o sequenciamento das tarefas de manutenção executadas por uma equipe de trabalho. As restrições (3.6) garantem que as equipes terminam a tarefa fictícia 0 no instante 0. As restrições (3.7) garantem que uma tarefa $j$ termina somente após a conclusão da tarefa imediatamente predecessora $i$ mais a sua duração. As restrições (3.8) e (3.9) garantem que toda manutenção seja executada na sua respectiva janela de tempo. As restrições (3.10) e (3.11) asseguram que duas ou mais manutenções não são executadas ao mesmo tempo na mesma máquina. Nota-se que estas duas últimas restrições só podem ser aplicadas entre equipes de trabalho diferentes, porque não haverá sobreposição da execução da manutenção pela mesma equipe de trabalho, sendo esta assegurada pelas restrições anteriores. As restrições (3.12) asseguram que o instante de conclusão da tarefa $i$ não exceda o instante limite de disponibilidade da equipe $k$. Finalmente, as restrições (3.13) a (3.17) definem o domínio das variáveis de decisão.

# 4. Algoritmos Propostos

A seguir, nas seções 4.1 e 4.2 são apresentados detalhes sobre como as soluções são representadas e avaliadas. Na Seção 4.3 são apresentados os algoritmos construtivos que são responsáveis por construir soluções iniciais viáveis para os algoritmos de refinamento. Na Seção 4.4, é apresentado o algoritmo meta-heurístico baseado no método ILS (*Iterated Local Search*) para tratar a versão determinística do problema. Por fim, na Seção 4.5, apresenta-se o algoritmo SIM-ILS para tratar a versão estocástica do problema.

## 4.1. Representação da Solução

Neste trabalho é utilizada a representação de solução do problema proposta por Aquino *et al.* (2019). Uma solução é representada de forma indireta por meio de uma sequência $s = \langle s_1, s_2, ..., s_n \rangle$ das $n$ tarefas de manutenção a serem alocadas. A sequência em que as tarefas de manutenção são inseridas é importante, pois define a prioridade de alocação; quanto mais à frente está uma dada tarefa na sequência, maior a sua prioridade.

Esse tipo de representação indireta da solução simplifica o processo de avaliação da solução proposta bem como a utilização de operadores de vizinhança para análise do espaço de soluções.

## 4.2. Avaliação da Solução

Uma solução $s$ é avaliada por meio do *Job Positioning Algorithm* (JPA), definido pelo Algoritmo 1. Além de calcular o custo da solução, ele também tem a função de fazer o posicionamento das tarefas de manutenção no horizonte de planejamento. Isto é, o JPA define se cada tarefa será executada ou não, e, se executada, qual será o instante de início de sua execução, bem como a equipe designada para executá-la.

No Algoritmo 1, inicialmente, o JPA identifica, para cada tarefa $i \in s$, se existe uma janela de tempo disponível na máquina onde ela deve ser executada (linha 4). Nas linhas 5-7, são identificadas equipes candidatas a executá-la, considerando suas habilidades e disponibilidades. Havendo uma equipe $k$ habilitada e disponível, a tarefa $i$ é alocada a ela na linha 8. O custo de utilização dessa equipe é atualizado na linha 11, caso ela ainda não tenha sido utilizada. Para cada tarefa $i$ não alocada, é imposta uma penalidade $\omega_i$ (linha 18) ao custo da solução. Por fim, o algoritmo retorna o custo total da solução $s$.

Para ilustrar o funcionamento do JPA, considere um exemplo com quatro tarefas, uma máquina e duas equipes de trabalho. A Tabela 4.1 descreve as quatro tarefas de manutenção para essa máquina e a sequência dessas tarefas. Já a Tabela 4.2 apresenta as características das duas equipes disponíveis para execução das tarefas.

**Algoritmo 1:** JPA - *Job Positioning Algorithm*

**Entrada:** Solução $s$, Conjunto de equipes ($\mathscr{W}$), Vetor de penalidades $\omega_i$ com $i \in s$
**Saída:** Custo

```
1  Custo ← 0;
2  para cada tarefa i ∈ s faça
3      alocou ← falso;
4      se existeJanela(i) então
5          para cada equipe k ∈ 𝒲 faça
6              se temHabilidade(i,k) então
7                  se JanelaCompativel(i,k) então
8                      Alocar tarefa i à equipe k;
9                      alocou ← verdadeiro;
10                     se equipe ainda não utilizada então
11                         Custo ← Custo + 1;
12                     fim
13                 fim
14             fim
15         fim
16     fim
17     se alocou = falso então
18         Custo ← Custo + ω_i;
19     fim
20 fim
21 retorna Custo
```

Tabela 4.1: Tarefas de Manutenção

| Máquina: Britador | | | | | |
|---|---|---|---|---|---|
| #Tarefa | Habilidade | Janela Início [$E$] | Janela Fim [$L$] | Duração | Peso [$\omega$] |
| 1 | Mecânica | 1 | 6 | 2 | 10 |
| 2 | Mecânica | 3 | 10 | 3 | 20 |
| 3 | Elétrica | 4 | 9 | 3 | 10 |
| 4 | Elétrica | 1 | 4 | 2 | 20 |

Fonte: Compilação do autor.

Tabela 4.2: Equipes de Manutenção

| Equipe | Habilidade [$\mathfrak{T}$] | Disp. Máxima [$H$] |
|---|---|---|
| 1 | Mecânica | 10 |
| 2 | Elétrica | 10 |

Fonte: Compilação do autor.

Na Tabela 4.3 reporta-se o resultado da aplicação do JPA para este exemplo. Nela, mostra-se a representação das janelas de tempo da máquina e das equipes de trabalho.

No resultado acima foi possível posicionar três das quatro tarefas de manutenção e foram utilizadas as duas equipes disponíveis. O valor da função objetivo é obtido somando-se o número de equipes usadas (no caso, duas) com a penalidade pela tarefa não executada, isto é, $Custo = 2 + 20 = 22$.

No JPA, a sequência em que as tarefas são submetidas tem influência direta no valor da função objetivo. No exemplo anterior, a sequência das tarefas foi a mesma descrita na Tabela 4.2: [1, 2, 3, 4]. Para verificar essa influência, se a coluna "Janela Fim [$L$]" da Tabela 4.1 for

Tabela 4.3: Posicionamento de tarefas

| | Horizonte de Planejamento | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Equipe 1 | 1 | 1 | 2 | 2 | 2 | | | | | |
| Equipe 2 | | | | | | 3 | 3 | 3 | | |
| Britador | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | | |

Fonte: Compilação do autor.

classificada de maneira crescente, então teremos uma nova sequência de tarefas: [4, 1, 3, 2].

Ao submeter essa nova sequência de tarefas ao JPA, tem-se uma nova representação das janelas de tempo da máquina e das equipes, que pode ser verificada na Tabela 4.4.

Tabela 4.4: Posicionamento de tarefas

| | Horizonte de Planejamento | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Equipe 1 | | | 1 | 1 | | | | 2 | 2 | 2 |
| Equipe 2 | 4 | 4 | | | 3 | 3 | 3 | | | |
| Britador | 4 | 4 | 1 | 1 | 3 | 3 | 3 | 2 | 2 | 2 |

Fonte: Compilação do autor.

Nessa nova sequência de tarefas foi possível realizar o posicionamento de todas elas, gerando um melhor valor para a função objetivo. Assim, a função objetivo terá apenas o custo de utilização das duas equipes, isto é, $Custo = 2$. Portanto, fica evidenciado que a ordem em que as tarefas são submetidas ao algoritmo JPA tem influência no valor da função objetivo.

A partir do JPA proposto, descrito pelo Algoritmo 1, foi proposta também uma nova variante deste JPA, cuja única diferença ocorre na opção de posicionamento da tarefa dentro do intervalo disponível, podendo ser no início ou no fim desse intervalo. As versões são aqui denominadas como JPA direto, para o JPA original (isto é, o JPA no qual as tarefas são posicionadas no início do intervalo), e JPA invertido para a nova variante (isto é, o JPA no qual as tarefas são posicionadas no fim do intervalo).

Para ilustrar as diferenças entre os algoritmos JPA direto e invertido, apresenta-se um exemplo de alocação de uma tarefa que tem como parâmetros: duração = 3, janela de início = 1 e janela de fim = 10.

O JPA direto faz o posicionamento das tarefas do exemplo conforme ilustrado na Figura 4.1, inserindo cada um delas no início do intervalo disponível.

Usando os mesmos dados do exemplo anterior, o JPA invertido faz o posicionamento das tarefas conforme a Figura 4.2, inserindo cada uma delas no final do intervalo disponível.

As duas versões do JPA foram desenvolvidas para verificar se existe e qual é o impacto dessas abordagens no valor da função objetivo do problema.

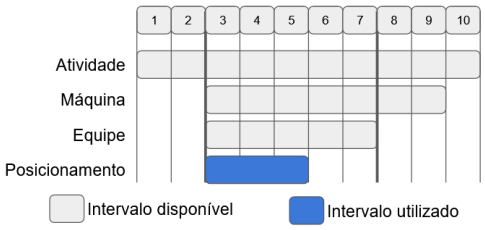



Figura 4.1: Posicionamento de tarefa com JPA Direto

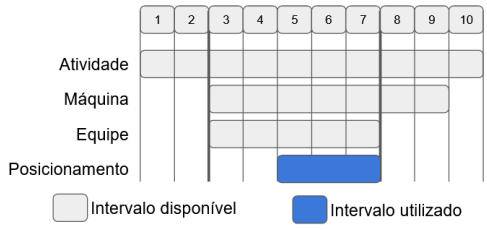



Figura 4.2: Posicionamento de tarefa com JPA Invertido

Fonte: Compilação do autor.

## 4.3. Algoritmos Heurísticos Construtivos

Para gerar uma solução inicial para o problema são utilizados algoritmos heurísticos construtivos, conhecidos em problemas de programação de tarefas como “regras de despacho”. Tratam-se de métodos para gerar sequências de tarefas, em que cada elemento de uma sequência é submetido a um conjunto de regras que verifica o benefício de sua inserção. Dessa forma, a cada passo do algoritmo, somente o elemento com maior beneficio é inserido na sequência.

No exemplo das tabelas 4.1 e 4.2, da seção anterior, foram aplicados dois algoritmos heurísticos construtivos simples. No primeiro, a regra utilizada é manter a sequência original das tarefas; já no segundo, a regra é determinar a sequência das tarefas de acordo com a menor “Janela Fim [$L$]”, isto é, priorizando as tarefas que encerram sua janela de execução primeiro. A solução para a primeira regra é $s = \langle 1,2,3,4 \rangle$ e para a segunda é $s' = \langle 4,1,3,2 \rangle$. Abaixo, apresenta-se o valor de função objetivo (ou Custo) das soluções geradas.

- $s = \langle 1,2,3,4 \rangle \rightarrow \ Custo = 22$;
- $s' = \langle 4,1,3,2 \rangle \rightarrow \ Custo = 2$.

Dessa forma, é possível desenvolver vários algoritmos heurísticos construtivos para gerar uma solução inicial, cada um deles baseado em uma regra. A seguir, apresenta-se na Tabela 4.5, um exemplo de instância com um conjunto maior de tarefas de manutenção a serem executadas.

Tabela 4.5: Características das tarefas de manutenção

| #Tarefa | Habilidade | Máquina | $[E]$ | $[L]$ | Duração | $[\omega]$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 100 | 0 | 10 | 5 | 10 |
| 2 | 2 | 100 | 7 | 20 | 7 | 20 |
| 3 | 1 | 200 | 7 | 20 | 5 | 10 |
| 4 | 2 | 200 | 15 | 30 | 7 | 20 |
| 5 | 1 | 100 | 0 | 10 | 3 | 30 |
| 6 | 2 | 200 | 15 | 30 | 2 | 30 |

Fonte: Compilação do autor.

Nesta tabela, verifica-se que para os vários parâmetros das tarefas de manutenção, há valores que se repetem, como o peso ($[\omega]$), a duração e janelas de início ($[E]$) e fim ($[L]$). Portanto, se utilizarmos um algoritmo heurístico construtivo simples, algumas tarefas podem "empatar" durante a avaliação do critério estabelecido. Para resolver essa questão, é possível utilizar um outro critério de desempate a ser avaliado, gerando, assim, algoritmos construtivos mais complexos. Tomando como base as tarefas da Tabela 4.5, podemos estabelecer, por exemplo, as seguintes regras para construir uma solução usando um algoritmo heurístico construtivo com critérios de desempate:

1. Inserir primeiro a tarefa que possui maior peso ($[\omega]$);
2. Em caso de empate na regra anterior, inserir a tarefa que possui a maior duração;
3. Em caso de empate na regra anterior, inserir a que possui menor janela fim ($[L]$).

Ao aplicar esse algoritmo construtivo nas tarefas da Tabela 4.5, chega-se à seguinte solução:

$$s'' = \langle 5,6,2,4,1,3 \rangle$$

Para facilitar o entendimento, podemos verificar na Tabela 4.6 os parâmetros das tarefas classificadas geradas na solução $s''$.

Tabela 4.6: Tarefas de manutenção ordenadas conforme regras estabelecidas

| #Tarefa | Habilidade | Máquina | $[E]$ | $[L]$ | Duração | $[\omega]$ |
|---|---|---|---|---|---|---|
| 5 | 1 | 100 | 0 | 10 | 3 | 30 |
| 6 | 2 | 200 | 15 | 30 | 2 | 30 |
| 2 | 2 | 100 | 7 | 20 | 7 | 20 |
| 4 | 2 | 200 | 15 | 30 | 7 | 20 |
| 1 | 1 | 100 | 0 | 10 | 5 | 10 |
| 3 | 1 | 200 | 7 | 20 | 5 | 10 |

Fonte: Compilação do autor.

Ao avaliarmos os parâmetros das tarefas de manutenção na Tabela 4.6, podemos separá-las em duas categorias: as discricionárias, que identificam as tarefas, e as chaves, que são os parâmetros de execução e o peso (ou prioridade).

- Colunas Discricionárias:
  - Coluna 0 - # Atv - Número da tarefa;
  - Coluna 1 - Habilidade - Tipo de habilidade requerida para execução;
  - Coluna 2 - Máquina - Equipamento onde a tarefa será executada;
- Colunas-chave:
  - Coluna 3 - Janela Início $[E]$ - Início do intervalo de execução da tarefa;

– Coluna 4 - Janela Fim $[L]$ - Fim do intervalo de execução da tarefa;

– Coluna 5 - Duração - Tempo necessário para execução da tarefa;

– Coluna 6 - Peso $[\omega]$ - Peso por não execução da tarefa.

Neste trabalho são utilizadas somente as colunas-chave para a elaboração dos algoritmos construtivos, pois esses fatores possuem impacto direto no custo da função objetivo calculada pelo JPA.

Cada critério que venha a integrar o algoritmo construtivo, isto é, cada coluna-chave, pode ser classificado de forma crescente ou decrescente. Para facilitar o entendimento, foi estabelecido o uso de uma variável binária para indicar a classificação das colunas, sendo:

- 0 para indicar classificação crescente;
- 1 para indicar classificação decrescente.

Voltando ao exemplo da construção da solução $s''$ no exemplo anterior, pode-se descrever a regra de construção utilizada da seguinte maneira:

1. Critério Penalidade $\omega$ (Coluna 6): Classificação 1;
2. Critério Duração (Coluna 5): Classificação 1;
3. Critério Janela Fim $[L]$ (Coluna 4): Classificação 0.

Para facilitar o uso desta regra, pode-se também representá-la pelo seguinte esquema:

$$\text{R} = [(6, 1), (5, 1), (4, 0)]$$

para indicar que as colunas 6 e 5 são ordenadas de forma crescente e a coluna 4, de forma decrescente.

Dessa forma, para cada combinação das quatro colunas-chave pode-se gerar um novo algoritmo de construção. Assim, para cada algoritmo, cada critério (ou coluna-chave) deve ser disposto uma única vez e de maneira sequêncial, sendo a posição do critério determinante para o seu grau de importância. Portanto, com essas quatro colunas-chave, por meio de uma análise combinatória, pode-se formar $4! = 24$ algoritmos heurísticos construtivos.

Em cada regra de construção estabelecida, cada uma das colunas pode ser classificada em ordem crescente ou decrescente. Essa classificação é representada por uma variável binária, o que torna possível gerar duas variações por coluna-chave de cada regra de construção. Como as regras são formadas por quatro elementos, temos $2^4 = 16$ variações de ordenação em cada regra.

Logo, utilizando as quatro colunas-chave e as variações de ordenação, é possível gerar $24 \times 16 = 384$ regras construtivas diferentes.

No Algoritmo 2 apresenta-se o pseudocódigo do método responsável por gerar as 384 regras construtivas e avaliá-las. Ao final, o método retorna a melhor solução construída, o seu custo e qual das regras teve o melhor desempenho.

**Algoritmo 2:** *Best Constructive Algorithm Generator*

**Entrada:** Conjunto de colunas chave ($\mathscr{C}$), JPA(Conjunto de tarefas ($\mathscr{T}$), Conjunto de equipes ($\mathscr{W}$))

**Saída:** Solução $s$, *Custo*, Melhor regra $R$ do algoritmo construtivo

```
1  n ← número de colunas chave;
2  Custo ← 0;
3  Custo' ← ∞;
4  para i ← 1 até n! faça
5      Combinacao_i ← geraComb(i, 𝒞);
6      para j ← 1 até 2^n faça
7          Variacao_j ← geraVar(j);
8          R ← (Combinacao_i, Variacao_j);
9          s ← constroiSol(R);
10         Custo ← JPA(s);
11         se Custo < Custo' então
12             s' ← s;
13             Custo' ← Custo;
14             R' ← R;
15         fim
16     fim
17 fim
18 s ← s';
19 Custo ← Custo';
20 R ← R'
21 retorna s, Custo, R;
```

O algoritmo recebe como parâmetros as colunas-chave, as tarefas, as equipes e o JPA que será utilizado (JPA Direto ou JPA Invertido). A cada iteração do laço de repetição entre as linhas 4 e 17, gera-se uma combinação para o conjunto de colunas-chave. A seguir, para cada combinação, no laço de repetição entre as linhas 6 e 16, gera-se uma classificação para esta combinação, estabelecendo uma nova regra de construção ($R$). Posteriormente, na linha 9, uma solução $s$ é construída com a regra $R$ e, na linha 10, o JPA posiciona suas tarefas. Ao fim de cada iteração, a solução $s$ é avaliada. Se ela tiver o melhor custo até então, ela é armazenada. Ao final do algoritmo construtivo, retorna-se a melhor solução $s$, o seu custo, e a melhor regra utilizada para construí-la.

## 4.4. Algoritmo *Iterated Local Search*

Para tratar a versão determinística do problema de planejamento de tarefas de manutenção, foi aplicado o algoritmo *Iterated Local Search* (ILS) (LOURENÇO *et al.*, 2001).

O ILS é um método meta-heurístico que consiste em explorar o espaço de soluções por meio de aplicação de busca local e operações de perturbação nos ótimos locais encontrados. O Algoritmo 3 mostra o pseudocódigo do algoritmo proposto para tratar a versão determinística do problema de manutenção em estudo.

**Algoritmo 3:** Iterated Local Search (ILS)

**Entrada:** $RDMax$, $ILSMax$, $k_{\max}$, $t_{\max}$

```
1  t ← 0;
2  s ← ConstroiSolução();
3  s* ← RandomDescent(s, RDMax);
4  ILSIter ← 0;
5  enquanto (ILSIter ≤ ILSMax & t ≤ t_max) faça
6      ILSIter ← ILSIter + 1;
7      k ← 1;
8      enquanto (k ≤ k_max) faça
9          k ← k + 1;
10         s' ← Shake(s, k);
11         s' ← RandomDescent(s', RDMax)
12         se f(s') < f(s*) então
13             s* ← s';
14             k ← 1;
15             ILSIter ← 0;
16         fim
17     fim
18 fim
19 retorna s*
```

O algoritmo ILS é descrito a seguir através de suas etapas principais. Ele começa gerando uma solução inicial, em seguida realiza o procedimento de busca local, *RandomDescent* e armazena o resultado em $s^{\star}$. O método de busca local explora um subconjunto de soluções vizinhas e retorna a melhor solução encontrada durante o processo de busca.

O algoritmo entra em um laço que itera um número máximo de vezes (*ILSMax*) ou até o limite de tempo estabelecido, perturbando a solução atual por meio da função *Shake*(.) até um determinado nível máximo de perturbação ($k_{\max}$).

É importante frisar que o nível de perturbação começa pequeno e aumenta gradativamente à medida que o método não encontra soluções de melhoria. Isso acontece para evitar que o algoritmo fique preso em uma região de ótimo local. A ideia é que, à medida que o nível de

perturbação aumenta, o algoritmo se afasta da região atual de busca e vá em direção a outras regiões do espaço de soluções do problema.

No entanto, toda vez que o algoritmo encontra uma solução melhor, o nível de perturbação retorna ao seu valor inicial. Isso é feito para possibilitar uma exploração mais profunda da região em torno da nova solução encontrada, ao mesmo tempo em que mantém a diversidade na busca.

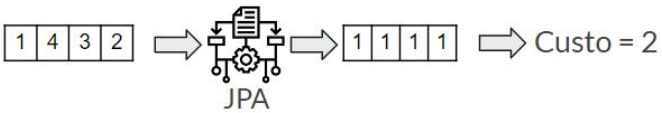
Depois de perturbar a solução, o algoritmo executa novamente o método *RandomDescent* para encontrar um ponto de melhoria na vizinhança da solução perturbada. Caso o seu custo seja menor do que o da solução vigente, ela é aceita e se torna a nova melhor solução; caso contrário, a solução vigente continua em vigor. O processo se repete até que o número máximo de iterações seja alcançado ou o tempo limite seja atingido.

O método ILS é de simples implementação e bem robusto, pois combina estratégias de intensificação e diversificação. A intensificação ocorre através de buscas locais realizadas após pequenas perturbações na solução vigente. A diversificação é realizada pela mudança na região de busca, situação que ocorre quando o nível de perturbação na solução vigente é aumentado gradativamente para o algoritmo não ficar preso em um ótimo local.

### 4.4.1. Busca local

Para fazer uma análise exploratória do espaço de soluções do problema, é definida uma estrutura de vizinhança simples que consiste na troca de posição de duas tarefas (*swap*) da solução vigente. Com esse tipo de movimento, é possível explorar todo o espaço de soluções do problema partindo-se de qualquer solução inicial.

Mantendo o exemplo das tabelas 4.1 e 4.2, temos a seguinte solução vigente: $s = \langle 1, \mathbf{2}, 3, \mathbf{4} \rangle$. Ao realizar o movimento de *swap* entre as atividades 2 e 4, chegamos à solução vizinha $s' = \langle 1, \mathbf{4}, 3, \mathbf{2} \rangle$. Na Figura 4.3 temos a avaliação da solução vizinha $s'$ pelo JPA. Observa-se que ela tem um custo menor do que a da solução inicial $s$. Logo, conclui-se que através da exploração de vizinhanças podemos chegar a soluções com menor custo, melhores do que a da solução inicial.




| s' | s'_avl | Equipe | Início | Duração |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 |
| 4 | 1 | 2 | 3 | 2 |
| 3 | 1 | 2 | 5 | 3 |
| 2 | 1 | 1 | 8 | 3 |

Figura 4.3: Avaliação de uma solução vizinha

Fonte: Compilação do autor.

A análise de toda a vizinhança pode ser um processo custoso à medida que a dimensão do problema aumenta. Ao analizarmos os exemplos até então apresentados, é possível perceber que o conjunto de tarefas da Tabela 4.5 é maior que o conjunto de tarefas da Tabela 4.1 e portanto, possui uma maior vizinhança.

Para este trabalho, a busca local aplicada foi pelo Método de Descida Randômico (*Random Descent Method*). Esse método consiste em selecionar aleatoriamente um vizinho para ser analisado, e caso o seu custo seja menor do que o da solução corrente, ele será aceito; caso contrário, a solução vigente continua em vigor e outro vizinho é selecionado aleatoriamente para ser avaliado. Esse processo continua até que o critério de parada determinado seja atingido. O Algoritmo 4 descreve seu pseudocódigo.

**Algoritmo 4:** *Random Descent*

**Entrada:** Solução $s$, *RDMax*

```
1  Iter ← 0;
2  enquanto (Iter < RDMax) faça
3  |  Iter ← Iter+1;
4  |  Selecione um vizinho s' de s aleatoriamente;
5  |  se f(s') < f(s) então
6  |  |  s ← s';
7  |  |  Iter ← 0;
8  |  fim
9  fim
10 retorna s
```

### 4.4.2. Perturbação

Ao final do processo de busca local, a solução retornada pode ser um ótimo local, isto é, uma solução em que todos os seus vizinhos apresentam um custo maior, fazendo com que o algoritmo fique preso naquela região.

Para evitar a parada do algoritmo nesses ótimos locais é aplicado o procedimento *Shake*, que consiste em gerar $k$ perturbações nesses ótimos locais. Cada perturbação consiste em aplicar o movimento *swap* entre duas tarefas.

**Algoritmo 5:** *Shake*

**Entrada:** Solução $s$, $k$

1 $s' \leftarrow s$;
2 $i \leftarrow 1$;
3 **enquanto** $(i \leq k)$ **faça**
4 $\quad s'' \leftarrow$ solução resultante de $s'$ pela troca aleatória de duas tarefas;
5 $\quad s' \leftarrow s''$;
6 $\quad i \leftarrow$i + 1;
7 **fim**
8 **retorna** $s'$

## 4.5. Algoritmo SIM-ILS

O algoritmo SIM-ILS proposto para tratar a versão estocástica do problema em estudo tem como base o algoritmo ILS (Algoritmo 3), e foi desenvolvido de acordo com as ideias de funcionamento dos algoritmos de Guimarans *et al.* (2018) e Keenan *et al.* (2021). Seu pseudocódigo é descrito pelo Algoritmo 6.

**Algoritmo 6:** *SIM-ILS*

**Entrada:** $n_{Fast}$, $n_{Deep}$, $RDMax$, $ILSMax$, $k_{\max}$, $t_{\max}$

```
1  t ← 0;
2  pool ← ∅;
3  s ← ConstructSolution();                          /* Conforme o Algoritmo 2 */
4  s* ← RandomDescent(s, RDMax);                     /* Conforme o Algoritmo 4 */
5  stochCost* ← Simulation(s*, n_Fast);              /* Conforme o Algoritmo 7 */
6  ILSIter ← 0;
7  enquanto (ILSIter ≤ ILSMax & t ≤ t_max) faça
8      ILSIter ← ILSIter + 1;
9      k ← 1;
10     enquanto (k ≤ k_max) faça
11         k ← k + 1;
12         s' ← Shake(s, k);                         /* Conforme o Algoritmo 5 */
13         s' ← RandomDescent(s', RDMax);            /* Conforme o Algoritmo 4 */
14         se f(s') < f(s*) então
15             stochCost ← Simulation(s', n_Fast);   /* Conforme o Algoritmo 7 */
16             se stochCost < stochCost* então
17                 s* ← s';
18                 stochCost* ← stochCost;
19                 insert(pool, s*);
20                 k ← 1;
21                 ILSIter ← 0;
22             fim
23         fim
24     fim
25 fim
26 para (s ∈ pool) faça
27     stochCost ← Simulation(s, n_Deep);            /* Conforme o Algoritmo 7 */
28     se stochCost < stochCost* então
29         s* ← s;
30         stochCost* ← stochCost;
31     fim
32 fim
33 retorna s*
```

O Algoritmo 6 inicializa as variáveis de controle de tempo e o *pool* de soluções nas linhas 1 e 2, respectivamente. Na linha 3, constrói-se uma solução por meio do Algoritmo 2. A seguir, na linha 4, aplica-se a busca local *RandomDescent*(.), definida pelo Algoritmo

4, à solução construída $s$. A solução refinada é armazenada em $s^\star$. Na linha 5, é calculado o custo estocástico da solução $s^\star$ utilizando a função *Simulation*(.) com $n_{Fast}$ iterações, descrita pelo Algoritmo 7. Em seguida, entra-se em um laço de repetição nas linhas 7-25 até que um dos critérios de parada sejam atendidos. Nas linhas 10-24 inicia-se um laço interno de repetição. Na linha 12, aplica-se o procedimento *Shake*(.) para perturbar a solução $s$, e na linha 13 realiza-se o refinamento da solução perturbada $s'$ através da busca local *RandomDescent*(.). A avaliação do custo determinístico da solução refinada $s'$ é feita na linha 14 por meio do Algoritmo 1. Caso $s'$ seja melhor que $s^\star$, ela é submetida à função *Simulation*(.) na linha 15 para encontrar seu custo estocástico. Caso $s'$ tenha um custo estocástico menor do que $s^\star$, a melhor solução é atualizada e inserida em um pool das melhores soluções.

Nas linhas 26-32, cada solução do *pool* é avaliada pela função *Simulation*(.), desta vez com $n_{Deep}$ iterações. Tal como anteriormente, cada solução é avaliada com relação ao seu custo estocástico e, caso ele seja menor do que o custo estocástico de $s^\star$ gerada durante toda a busca, ela é armazenada como a melhor solução. Ao final, o algoritmo SIM-ILS retorna a melhor solução para a versão estocástica do problema.

### 4.5.1. Procedimento de simulação

A função *Simulation*(.) realiza a avaliação estocástica de uma solução pelo Método de Monte Carlo. Seu pseudocódigo é apresentado pelo Algoritmo 7.

**Algoritmo 7:** *Simulation*

**Entrada:** $n_{Fast}$, $s$, JPA(.), *Conjunto de tarefas* ($\mathscr{T}$), *Conjunto de equipes* ($\mathscr{W}$), *Rand*(.), Tipo de distribuição (*distrib*), Desvio percentual (*desv*)

```
1  T_stoch ← ∅;
2  stochCost ← 0;
3  j ← 0;
4  enquanto (j ≤ n_Fast) faça
5  |   T_stoch ← T;
6  |   para cada tarefa i ∈ T_stoch faça
7  |   |   P_i ← P_i × Rand(distrib, desv);
8  |   fim
9  |   custo ← JPA(s, T_stoch, W);
10 |   stochCost ← stochCost + custo;
11 fim
12 stochCost ← stochCost / n_Fast;
13 retorna stochCost
```

A função *Simulation*(.) consiste em executar, pelas linhas 4-11, $n_{Fast}$ iterações de cálculo do custo da função objetivo com os valores de duração das tarefas simuladas através de uma distribuição normal. Para cada tarefa, linhas 6-8, calcula-se a sua nova duração $P_i$ utilizando-se

a função *Rand*(.), descrita pelo Algoritmo 8. O custo dessa solução estocástica é calculado na linha 9. Na linha 12, calcula-se o valor médio das $n_{Fast}$ soluções estocásticas geradas. Ao final, retorna-se esse valor médio como o custo estocástico da solução $s^{\star}$.

O Algoritmo 8 descreve como é gerada a aleatoriedade da função *Rand*(.) utilizando a distribuição normal. Essa distribuição utiliza dois parâmetros: 1) $\mu$, que é a média da distribuição, isto é, onde ela está centralizada e 2) $\sigma$, que é seu desvio padrão. Ao avaliarmos a curva de uma distribuição normal, verifica-se que 99,74% dos dados têm uma probabilidade de ocorrência dentro do intervalo de mais ou menos $3\sigma$.

**Algoritmo 8:** *Rand*

**Entrada:** Tipo de distribuição (*distrib*), Desvio percentual (*desv*)

1 $\mu \leftarrow 0$;
2 $\sigma \leftarrow desv/3$;
3 $r \leftarrow CreateRand(distrib, \mu, \sigma)$;
4 $r \leftarrow r + 1$;
5 **retorna** $r$

Na linha 1 do Algoritmo 8 é determinado que a distribuição seja centrada em 0. Na linha 2, o valor do desvio percentual *desv* é dividido por 3 para garantir que 99,74% dos dados gerados estejam dentro do intervalo desejado. Na linha 3 é gerado um valor aleatório pela função *CreateRand*(.) da linguagem de programação. Por fim, na linha 4, soma-se 1 ao valor gerado para criar o fator de multiplicação a ser aplicado à duração das tarefas.

# 5. Experimentos Computacionais

Este capítulo apresenta os resultados dos experimentos computacionais realizados com os algoritmos propostos no capítulo anterior. Inicialmente, são apresentados os resultados dos algoritmos construtivos, que foram avaliados em termos de qualidade das soluções encontradas e tempo de processamento. Em seguida, são apresentados os resultados do algoritmo meta-heurístico, que foi comparado com os resultados da literatura. Por fim, são apresentados os resultados do algoritmo simheurístico, que foram utilizados para tratar das características estocásticas do problema. Os resultados obtidos são discutidos e comparados com os resultados do algoritmo meta-heurístico proposto, com o objetivo de avaliar o desempenho dos algoritmos propostos e sua aplicabilidade prática.

## 5.1. Resultados dos algoritmos heurísticos construtivos e das duas versões do JPA

As duas versões do JPA (direto e indireto) e os algoritmos heurísticos construtivos foram desenvolvidos na linguagem de programação C++. Os algoritmos foram executados na plataforma Google Colab[1] em sua versão gratuita, que disponibiliza um processador Intel Xeon E5-2699 v4 @ 2.20GHz.

As instâncias aqui utilizadas foram aquelas de Aquino *et al.* (2019), e estão disponibilizadas em Aquino e Souza (2016). Os experimentos foram realizados utilizando-se 37 instâncias de tamanhos variados com as seguintes características: *i*) 150 a 33484 tarefas; *ii*) 57 a 263 equipes de trabalho e *iii*) 91 a 1286 máquinas.

Para cada instância, foram testadas todas as 384 regras construtivas, cada qual usando as duas versões do JPA, direto e invertido. Ao final da execução, retorna-se qual regra obteve o melhor desempenho e qual foi o seu custo.

A Tabela 5.1 reporta os resultados dessas regras que compõem algoritmo construtivo. As colunas $A$, $E$ e $M$ correspondem, respectivamente, à quantidade de tarefas, equipes de manutenção e máquinas em cada instância. Em seguida, as colunas $Obj$, $\#A$, $\#E$, $T(s)$ e $R$ correspondem, respectivamente, ao valor da função objetivo gerada pela melhor regra do algoritmo construtivo, a quantidade de tarefas que foram programadas, o número de equipes utilizadas, o tempo de execução do algoritmo, em segundos, e a melhor regra utilizada pelo algoritmo construtivo.

Ao avaliar os resultados apresentados na Tabela 5.1, verifica-se que o algoritmo proposto (Algoritmo 2) constrói uma boa solução, a melhor dentre as 384 soluções construídas para cada JPA, em tempo computacional reduzido. Para instâncias com até 1200 tarefas, o algoritmo precisa de menos de 15 segundos para construir uma boa solução; já na maior instância, a que retrata o caso real, o algoritmo precisa de cerca de 50 minutos para encontrar uma boa

[1] A plataforma pode ser acessada no endereço `https://colab.research.google.com/`

Tabela 5.1: Resultados dos algoritmos heurísticos construtivos propostos

| Instâncias | | | Algoritmo Construtivo com JPA Direto | | | | | Algoritmo Construtivo com JPA Invertido | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | E | M | Obj | #A | #E | T(s) | R | Obj | #A | #E | T(s) | R |
| 150 | 148 | 120 | 1851 | 143 | 31 | 0,8 | [4,0 - 5,1 - 6,0 - 3,0] | **1848** | 143 | 28 | 0,8 | [4,1 - 6,1 - 5,0 - 3,0] |
| 150 | 75 | 129 | **30** | 150 | 30 | 0,7 | [6,0 - 5,0 - 4,0 - 3,0] | **30** | 150 | 30 | 0,7 | [6,0 - 5,0 - 4,0 - 3,0] |
| 150 | 102 | 91 | 3281 | 149 | 35 | 0,7 | [3,0 - 4,0 - 6,1 - 5,0] | **3273** | 149 | 27 | 0,7 | [4,1 - 6,0 - 5,0 - 3,0] |
| 150 | 57 | 126 | 25 | 150 | 25 | 0,8 | [5,0 - 6,0 - 4,0 - 3,0] | **24** | 150 | 24 | 0,8 | [6,1 - 5,0 - 4,0 - 3,0] |
| 150 | 92 | 93 | 303 | 139 | 52 | 0,7 | [3,0 - 6,1 - 5,0 - 4,0] | **49** | 150 | 49 | 0,7 | [6,1 - 5,0 - 4,0 - 3,0] |
| 150 | 71 | 101 | 681 | 146 | 45 | 0,7 | [5,1 - 4,0 - 6,0 - 3,0] | **106** | 149 | 34 | 0,7 | [6,1 - 5,0 - 4,0 - 3,0] |
| 300 | 158 | 179 | 5694 | 275 | 54 | 1,8 | [6,1 - 5,0 - 4,1 - 3,0] | **2023** | 292 | 43 | 1,8 | [4,1 - 6,1 - 5,0 - 3,0] |
| 300 | 221 | 239 | 2460 | 286 | 56 | 2,1 | [6,0 - 5,0 - 4,0 - 3,0] | **2470** | 285 | 54 | 1,9 | [4,1 - 6,0 - 5,0 - 3,0] |
| 300 | 112 | 177 | 3720 | 292 | 57 | 1,8 | [3,0 - 4,1 - 5,1 - 6,0] | **3360** | 298 | 42 | 1,7 | [4,1 - 3,1 - 6,1 - 5,0] |
| 300 | 75 | 181 | 182 | 298 | 38 | 1,8 | [4,0 - 5,0 - 6,0 - 3,0] | **35** | 300 | 35 | 1,8 | [4,1 - 6,1 - 5,0 - 3,0] |
| 300 | 121 | 162 | 189 | 297 | 79 | 1,8 | [3,0 - 4,1 - 5,1 - 6,0] | **65** | 300 | 65 | 1,8 | [4,1 - 6,0 - 5,0 - 3,0] |
| 300 | 119 | 176 | 1181 | 288 | 59 | 1,7 | [6,1 - 5,0 - 3,0 - 4,0] | **308** | 297 | 41 | 1,8 | [4,1 - 6,0 - 5,0 - 3,0] |
| 600 | 165 | 329 | 9703 | 558 | 73 | 4,3 | [3,0 - 5,1 - 6,0 - 4,1] | **4879** | 580 | 55 | 4,3 | [4,1 - 5,1 - 6,0 - 3,0] |
| 600 | 256 | 388 | 3556 | 569 | 85 | 4,4 | [3,0 - 4,0 - 5,0 - 6,1] | **3448** | 570 | 77 | 4,5 | [4,1 - 3,0 - 6,0 - 5,0] |
| 600 | 120 | 288 | 10019 | 571 | 65 | 4,0 | [5,1 - 6,1 - 3,1 - 4,0] | **9075** | 579 | 60 | 4,0 | [5,1 - 6,0 - 3,0 - 4,1] |
| 600 | 77 | 215 | 613 | 594 | 45 | 4,0 | [6,0 - 5,0 - 3,1 - 4,0] | **37** | 600 | 37 | 4,1 | [4,1 - 3,0 - 5,1 - 6,0] |
| 600 | 126 | 279 | 732 | 588 | 87 | 4,0 | [3,0 - 6,1 - 5,0 - 4,1] | **410** | 598 | 67 | 3,9 | [4,1 - 6,1 - 5,0 - 3,0] |
| 1200 | 186 | 519 | 18072 | 1103 | 88 | 12,0 | [5,1 - 3,0 - 4,1 - 6,1] | **13067** | 1137 | 71 | 12,2 | [4,1 - 5,1 - 3,0 - 6,1] |
| 1200 | 263 | 666 | 10883 | 1102 | 116 | 11,5 | [5,1 - 6,0 - 3,1 - 4,1] | **9893** | 1110 | 96 | 11,6 | [4,1 - 3,0 - 5,1 - 6,0] |
| 1200 | 122 | 470 | 25543 | 1143 | 73 | 10,6 | [5,1 - 3,1 - 4,0 - 6,0] | **23815** | 1162 | 62 | 10,7 | [6,1 - 5,0 - 3,1 - 4,0] |
| 1200 | 88 | 252 | 1965 | 1177 | 51 | 10,9 | [5,1 - 6,1 - 4,1 - 3,0] | **299** | 1199 | 39 | 11,3 | [4,1 - 5,0 - 6,1 - 3,0] |
| 1200 | 130 | 420 | 2758 | 1156 | 92 | 10,3 | [5,1 - 3,0 - 6,0 - 4,1] | **555** | 1195 | 72 | 10,3 | [4,1 - 6,1 - 5,0 - 3,1] |
| 1200 | 122 | 403 | 13702 | 1110 | 83 | 10,4 | [6,1 - 5,0 - 4,0 - 3,1] | **7402** | 1170 | 69 | 10,5 | [5,1 - 3,1 - 6,0 - 4,1] |
| 2400 | 188 | 738 | 37646 | 2177 | 93 | 34,5 | [6,1 - 5,0 - 3,0 - 4,1] | **31963** | 2230 | 83 | 35,7 | [4,1 - 5,1 - 6,0 - 3,0] |
| 2400 | 130 | 603 | 51367 | 2231 | 80 | 29,3 | [5,1 - 6,0 - 4,0 - 3,0] | **44941** | 2303 | 70 | 28,5 | [5,1 - 4,1 - 6,0 - 3,1] |
| 2400 | 90 | 278 | 6193 | 2345 | 67 | 31,0 | [4,1 - 6,1 - 5,0 - 3,0] | **1510** | 2391 | 54 | 31,8 | [4,1 - 3,1 - 6,1 - 5,0] |
| 2400 | 132 | 701 | 3459 | 2330 | 102 | 27,5 | [5,1 - 6,0 - 3,0 - 4,0] | **1058** | 2388 | 83 | 27,9 | [4,1 - 3,1 - 6,1 - 5,0] |
| 2400 | 126 | 561 | 17595 | 2268 | 92 | 27,3 | [6,1 - 5,0 - 3,0 - 4,1] | **9276** | 2346 | 84 | 27,8 | [5,1 - 3,0 - 6,0 - 4,1] |
| 4800 | 197 | 1128 | 70037 | 4372 | 114 | 107,0 | [5,1 - 3,1 - 4,1 - 6,0] | **62760** | 4459 | 98 | 110,1 | [4,1 - 3,0 - 6,1 - 5,0] |
| 4800 | 78 | 1286 | 55313 | 4379 | 177 | 85,1 | [6,1 - 5,0 - 3,1 - 4,1] | **45143** | 4460 | 144 | 88,0 | [4,1 - 3,1 - 6,1 - 5,0] |
| 4800 | 130 | 720 | 117255 | 4413 | 85 | 82,6 | [5,1 - 4,0 - 6,1 - 3,0] | **102571** | 4553 | 75 | 84,7 | [5,1 - 4,1 - 6,0 - 3,1] |
| 4800 | 91 | 294 | 19221 | 4646 | 69 | 98,6 | [4,1 - 6,1 - 5,0 - 3,0] | **2545** | 4781 | 57 | 99,6 | [4,1 - 3,1 - 6,1 - 5,0] |
| 4800 | 135 | 990 | 16628 | 4642 | 110 | 82,5 | [3,0 - 4,1 - 6,1 - 5,0] | **3194** | 4776 | 90 | 83,8 | [4,1 - 5,1 - 6,0 - 3,1] |
| 4800 | 129 | 713 | 30170 | 4594 | 96 | 81,9 | [6,1 - 5,0 - 4,0 - 3,1] | **16390** | 4709 | 82 | 83,6 | [6,1 - 5,0 - 4,1 - 3,1] |
| 9600 | 132 | 816 | 72860 | 9184 | 104 | 259,7 | [5,1 - 6,1 - 3,0 - 4,1] | **39849** | 9407 | 88 | 263,4 | [5,1 - 6,0 - 4,1 - 3,0] |
| 19200 | 132 | 908 | 156077 | 18377 | 105 | 947,8 | [6,1 - 5,0 - 4,0 - 3,1] | **89924** | 18812 | 94 | 970,9 | [5,1 - 6,0 - 4,1 - 3,1] |
| 33484 | 145 | 1032 | 234613 | 32231 | 111 | 2885,8 | [5,1 - 4,0 - 6,1 - 3,0] | **141902** | 32870 | 92 | 3006,6 | [5,1 - 6,1 - 3,1 - 4,0] |

Fonte: Compilação do autor.

solução. Ainda na avaliação dos resultados dessa tabela, verifica-se que a versão com JPA invertido obteve o melhor desempenho por ter gerado os menores custos em todas as instâncias avaliadas. Pode-se atribuir o melhor desempenho do JPA Invertido devido à alocação das tarefas no final do intervalo disponível, esse fato contribui para que tarefas com janelas de execução mais restritivas, que estejam mais adiante na solução avaliada, encontrem intervalos disponíveis para sua alocação.

A seguir, na Tabela 5.2, são comparados os resultados do melhor algoritmo construtivo proposto com os melhores resultados dos algoritmos meta-heurísticos de Aquino *et al.* (2019): o MSVNS, o BRKGA e o BRKMA. Nesta tabela, o tempo de execução dos algoritmos meta-heurísticos foi dividido pelo fator de conversão 2,35, pois de acordo com PassMark (2022), o computador utilizado nessas execuções é 2,35 mais lento que o utilizado neste trabalho. Nesta tabela, a coluna RPD indica o desvio percentual relativo dos algoritmos em relação à melhor solução encontrada por todos os algoritmos meta-heurísticos para cada instância, calculada pela Eq. (5.1):

$$RPD_i = \frac{Obj_i^{Alg} - Obj_i^{Best}}{Obj_i^{Best}} \tag{5.1}$$

em que $Obj_i^{Alg}$ é o valor de função objetivo encontrado pelo algoritmo $Alg$ na instância $i$ e $Obj_i^{Best}$ é o melhor valor para a função objetivo encontrado pelos três algoritmos meta-heurísticos de Aquino *et al.* (2019). Valores de RPD negativos indicam que o algoritmo $Alg$

encontrou valores melhores do que os da literatura.

Tabela 5.2: Resultados da comparação entre o algoritmo construtivo com JPA invertido e os algoritmos meta-heurísticos de Aquino *et al.* (2019)

| Instâncias | | | Construtivo com JPA Invertido | | | MSVNS | | | BRKGA | | | BRKMA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | E | M | Obj | T(s) | RPD (%) | Obj | T(s) | RPD (%) | Obj | T(s) | RPD (%) | Obj | T(s) | RPD (%) |
| 150* | 148* | 120* | 1848 | 0,8 | - | - | - | - | - | - | - | - | - | - |
| 150 | 75 | 129 | **30** | 0,7 | 0,00 | **30** | 63,8 | 0,00 | **30** | 63,8 | 0,00 | **30** | 63,8 | 0,00 |
| 150 | 102 | 91 | **3273** | 0,7 | 0,00 | **3273** | 63,8 | 0,00 | **3273** | 63,8 | 0,00 | **3273** | 63,8 | 0,00 |
| 150 | 57 | 126 | **24** | 0,8 | 0,00 | **24** | 63,8 | 0,00 | **24** | 63,8 | 0,00 | **24** | 63,8 | 0,00 |
| 150 | 92 | 93 | **49** | 0,7 | 0,00 | **49** | 63,8 | 0,00 | **49** | 63,8 | 0,00 | **49** | 63,8 | 0,00 |
| 150 | 71 | 101 | **106** | 0,7 | 0,00 | **106** | 63,8 | 0,00 | **106** | 63,8 | 0,00 | **106** | 63,8 | 0,00 |
| 300* | 158* | 179* | 2023 | 1,8 | - | - | - | - | - | - | - | - | - | - |
| 300 | 221 | 239 | 2470 | 1,9 | 0,73 | **2452** | 127,6 | 0,00 | **2452** | 127,6 | 0,00 | **2452** | 127,6 | 0,00 |
| 300 | 112 | 177 | **3360** | 1,7 | 0,00 | **3360** | 127,6 | 0,00 | 3361 | 127,6 | 0,03 | 3362 | 127,6 | 0,06 |
| 300 | 75 | 181 | **35** | 1,8 | 0,00 | **35** | 127,6 | 0,00 | 37 | 127,6 | 5,71 | 37 | 127,6 | 5,71 |
| 300 | 121 | 162 | **65** | 1,8 | -76,87 | 281 | 127,6 | 0,00 | 282 | 127,6 | 0,36 | 282 | 127,6 | 0,36 |
| 300 | 119 | 176 | **308** | 1,8 | 0,00 | **308** | 127,6 | 0,00 | **308** | 127,6 | 0,00 | **308** | 127,6 | 0,00 |
| 600* | 165* | 329* | 4879 | 4,3 | - | - | - | - | - | - | - | - | - | - |
| 600 | 256 | 388 | 3448 | 4,5 | 1,89 | **3384** | 255,1 | 0,00 | **3384** | 255,1 | 0,00 | 3388 | 255,1 | 0,12 |
| 600 | 120 | 288 | 9075 | 4,0 | 5,79 | **8578** | 255,1 | 0,00 | 8580 | 255,1 | 0,02 | 8584 | 255,1 | 0,07 |
| 600 | 77 | 215 | **37** | 4,1 | -11,90 | 42 | 255,1 | 0,00 | 43 | 255,1 | 2,38 | 43 | 255,1 | 2,38 |
| 600 | 126 | 279 | **410** | 3,9 | -0,97 | 414 | 255,1 | 0,00 | 415 | 255,1 | 0,24 | 416 | 255,1 | 0,48 |
| 1200* | 186* | 519* | 13067 | 12,2 | - | - | - | - | - | - | - | - | - | - |
| 1200 | 263 | 666 | 9893 | 11,6 | 3,84 | **9527** | 510,3 | 0,00 | 9587 | 510,3 | 0,63 | 9575 | 510,3 | 0,50 |
| 1200 | 122 | 470 | 23815 | 10,7 | 8,60 | **21930** | 510,3 | 0,00 | 22075 | 510,3 | 0,66 | 22318 | 510,3 | 1,77 |
| 1200 | 88 | 252 | **299** | 11,3 | -3,24 | 309 | 510,3 | 0,00 | 765 | 510,3 | 147,57 | 390 | 510,3 | 26,21 |
| 1200 | 130 | 420 | 555 | 10,3 | 5,51 | **526** | 510,3 | 0,00 | 532 | 510,3 | 1,14 | 679 | 510,3 | 29,09 |
| 1200 | 122 | 403 | 7402 | 10,5 | 8,20 | **6841** | 510,3 | 0,00 | **6841** | 510,3 | 0,00 | 6842 | 510,3 | 0,01 |
| 2400 | 188 | 738 | 31963 | 35,7 | 19,69 | 26777 | 1020,6 | 0,27 | **26705** | 1020,6 | 0,00 | 27179 | 1020,6 | 1,77 |
| 2400 | 130 | 603 | 44941 | 28,5 | 17,97 | **38094** | 1020,6 | 0,00 | 39614 | 1020,6 | 3,99 | 38764 | 1020,6 | 1,76 |
| 2400 | 90 | 278 | **1510** | 31,8 | -4,73 | 2000 | 1020,6 | 26,18 | 2026 | 1020,6 | 27,82 | 1585 | 1020,6 | 0,00 |
| 2400 | 132 | 701 | 1058 | 27,9 | 29,66 | **816** | 1020,6 | 0,00 | 1768 | 1020,6 | 116,67 | 1608 | 1020,6 | 97,06 |
| 2400 | 126 | 561 | 9276 | 27,8 | 12,31 | **8259** | 1020,6 | 0,00 | 8839 | 1020,6 | 7,02 | 8775 | 1020,6 | 6,25 |
| 4800 | 197 | 1128 | 62760 | 110,1 | 5,34 | **59580** | 2041,1 | 0,00 | 62395 | 2041,1 | 4,72 | 61138 | 2041,1 | 2,61 |
| 4800 | 78 | 1286 | 45143 | 88,0 | 7,68 | **41925** | 2041,1 | 0,00 | 42643 | 2041,1 | 1,71 | 42781 | 2041,1 | 2,04 |
| 4800 | 130 | 720 | 102571 | 84,7 | 11,10 | 94780 | 2041,1 | 2,66 | 95629 | 2041,1 | 3,58 | **92320** | 2041,1 | 0,00 |
| 4800 | 91 | 294 | **2545** | 99,6 | -98,72 | 207390 | 2041,1 | 4,56 | 199117 | 2041,1 | 0,39 | 198340 | 2041,1 | 0,00 |
| 4800 | 135 | 990 | **3194** | 83,8 | -46,84 | 6194 | 2041,1 | 3,10 | 6008 | 2041,1 | 0,00 | 6088 | 2041,1 | 1,33 |
| 4800 | 129 | 713 | **16390** | 83,6 | -2,12 | 16745 | 2041,1 | 0,00 | 19062 | 2041,1 | 13,84 | 17749 | 2041,1 | 6,00 |
| 9600 | 132 | 816 | 39849 | 263,4 | 11,38 | 50272 | 4082,2 | 40,51 | **35778** | 4082,2 | 0,00 | 38004 | 4082,2 | 6,22 |
| 19200 | 132 | 908 | **89924** | 970,9 | -12,50 | 198838 | 8164,4 | 93,47 | 102773 | 8164,4 | 0,00 | 103863 | 8164,4 | 1,06 |
| 33484 | 145 | 1032 | **141902** | 3006,6 | -35,51 | 616479 | 14238,4 | 180,16 | 223752 | 14238,4 | 1,68 | 220048 | 14238,4 | 0,00 |

*Solução infactível retornada pelos algoritmos MSVNS, BRKGA e BRKMA.

Fonte: Compilação do autor.

Pelos resultados apresentados na Tabela 5.2, verifica-se que o melhor algoritmo construtivo demanda um tempo muito menor que o dos algoritmos meta-heurísticos. Em instâncias com até 1200 tarefas, o construtivo demanda até 100 vezes menos tempo que os algoritmos meta-heurísticos. Já na maior instância, o construtivo precisa de cerca de 50 minutos para construir uma boa solução, enquanto os algoritmos meta-heurísticos necessitam de cerca de 4 horas.

Sobre a qualidade das soluções geradas, o melhor algoritmo construtivo conseguiu encontrar a melhor solução em 18 das 37 instâncias testadas. Além disso, ele encontrou uma melhor solução para a maior instância, que representa o caso real da indústria em estudo.

Esses resultados validam a importância das estratégias implementadas tanto no algoritmo de posicionamento de tarefas, como na execução das regras estabelecidas nos algoritmos heurísticos construtivos desenvolvidos.

## 5.2. Resultados dos Algoritmos ILS e SIM-ILS

Os algoritmos propostos, JPA, ILS e SIM-ILS, foram implementados na linguagem de programação C++. Os experimentos foram divididos em duas etapas. Na primeira comparamos os resultados do JPA e do ILS com os dos algoritmos meta-heurísticos de Aquino *et al.* (2019):

o MSVNS, o BRKGA e o BRKMA, que são o estado da arte deste problema. Na segunda etapa, os resultados do algoritmo ILS, que obteve o melhor desempenho na primeira etapa, são comparados com os do SIM-ILS.

Os algoritmos JPA, ILS e SIM-ILS foram executados em um computador Intel Xeon E-2378G @ 2.80GHz X 16, 64GB RAM, com um sistema operacional Ubuntu 20.04.4 LTS. Apesar de essa máquina possibilitar multiprocessamento, os algoritmos desenvolvidos não usam esse recurso. Já os algoritmos da literatura, MSVNS, BRKGA e BRKMA, foram executados em um computador Intel Xeon E5-2660 v2 @ 2.20GHz x 40, com 384GB de memória RAM utilizando multiprocessamento.

Para testar os algoritmos utilizamos 33 das 37 instâncias grandes de Aquino *et al.* (2019), disponibilizadas em Aquino e Souza (2016). Essas instâncias têm as seguintes características: *i*) 150 a 33484 tarefas; *ii*) 57 a 263 equipes de trabalho e *iii*) 91 a 1286 máquinas. Quatro instâncias desse conjunto foram eliminadas da comparação após verificar inviabilidade nos resultados reportados.

### 5.2.1. Calibração de parâmetros

O algoritmo ILS possui três parâmetros a serem calibrados. Para estabelecer seus valores, foi utilizada a ferramenta Irace (*Iterated Racing for Automatic Algorithm Configuration*) (LÓPEZ-IBÁÑEZ *et al.*, 2016). O pacote Irace é um software que implementa vários procedimentos de configuração automática, baseados em uma faixa de valores e em instâncias fornecidas para o problema em questão. Ao final do processo de calibração, o Irace retorna os melhores valores encontrados para os parâmetros de algoritmo calibrado.

Para realizar o experimento de calibração, foram selecionadas três instâncias representativas de todo o conjunto, com as seguintes características: *i*) 80, 150 e 300 tarefas; *ii*) 14, 148 e 158 equipes de trabalho e *iii*) 5, 120 e 179 máquinas. Também foram estabelecidas faixas de valores para os parâmetros, conforme descrito a seguir na Tabela 5.3.

Tabela 5.3: Resultado da calibração de parâmetros pelo Irace.

| Parâmetro | Descrição | Faixa de Valores | Valor Retornado |
|---|---|---|---|
| *ILSMax* | Número máximo de iterações sem melhora no ILS | {30, 40, 50} | 50 |
| $k_{\max}$ | Nível máximo de perturbações na solução vigente | {10, 15, 20} | 15 |
| *RDMax* | Número máximo de iterações do método de busca local | {30, 40, 50 } | 40 |

Fonte: Compilação do autor.

### 5.2.2. Comparação entre JPA, ILS, MSVNS, BRKGA e BRKMA

Para cada instância, foram testados todos os 384 algoritmos construtivos, cada qual usando as duas versões do JPA, direto e invertido. Ao final da execução, retorna-se qual al-

goritmo construtivo obteve o melhor desempenho e qual foi o seu custo. O JPA invertido obteve o melhor desempenho por ter gerado os menores custos em todas as instâncias avaliadas. Portanto, somente seus resultados foram utilizados na comparação com os demais algoritmos. Além disso a solução fornecida pelo melhor algoritmo construtivo, foi usada como solução inicial do método ILS.

Na Tabela 5.4 são comparados os resultados dos algoritmos propostos, JPA e ILS, com os melhores resultados dos algoritmos meta-heurísticos de Aquino *et al.* (2019): o MSVNS, o BRKGA e o BRKMA. As colunas $A$, $E$ e $M$ correspondem, respectivamente, à quantidade de tarefas, equipes de manutenção e máquinas em cada instância. Em seguida, as colunas *Avg*, *Best*, *Gap* correspondem respectivamente, à média dos resultados encontrados, ao melhor resultado encontrado e ao desvio percentual relativo em relação à melhor solução encontrada por todos os algoritmos (*BKV*, do inglês *Best Known Value* ) para cada instância, calculada pela Eq. (5.2):

$$Gap = \frac{f_{Avg} - f_{BKV}}{f_{BKV}} \tag{5.2}$$

Para o JPA, os valores das colunas *Avg* e *Best* são iguais, pois esse algoritmo é determinístico. Para o ILS, a coluna *Avg* reporta a média de 30 execuções para cada instância. Para o MSVNS, o BRKGA e o BRKMA, essa coluna reporta os mesmos valores de Aquino *et al.* (2019), que executaram 5 vezes cada instância. As instâncias indicadas com asterisco (*) são aquelas cujos resultados reportados por Aquino *et al.* (2019) são inviáveis. Os melhores resultados para cada instância estão destacados em negrito.

Tabela 5.4: Resultado da comparação entre os algoritmos JPA, ILS, MSVNS, BRKGA e BRKMA.

| Instância | | | JPA invertido | | | ILS | | | MSVNS | | | BRKGA | | | BRKMA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | E | M | Avg | Best | Gap (%) | Avg | Best | Gap (%) | Avg | Best | Gap (%) | Avg | Best | Gap (%) | Avg | Best | Gap (%) |
| 150* | 148* | 120* | 1848 | 1848 | - | 1848 | 1848 | - | - | - | - | - | - | - | - | - | - |
| 150 | 75 | 129 | 30 | **30** | 0,00 | 30 | **30** | 0,00 | 30 | **30** | 0,00 | 30 | **30** | 0,00 | 30 | **30** | 0,00 |
| 150 | 102 | 91 | 3273 | **3273** | 0,00 | 3273 | **3273** | 0,00 | 3273 | **3273** | 0,00 | 3273 | **3273** | 0,00 | 3273 | **3273** | 0,00 |
| 150 | 57 | 126 | 24 | **24** | 0,00 | 24 | **24** | 0,00 | 24 | **24** | 0,00 | 24 | **24** | 0,00 | 24 | **24** | 0,00 |
| 150 | 92 | 93 | 49 | **49** | 0,00 | 49 | **49** | 0,00 | 49 | **49** | 0,00 | 49 | **49** | 0,00 | 49 | **49** | 0,00 |
| 150 | 71 | 101 | 106 | **106** | 0,00 | 106 | **106** | 0,00 | 106 | **106** | 0,00 | 106 | **106** | 0,00 | 106 | **106** | 0,00 |
| 300* | 158* | 179* | 2023 | 2023 | - | 1938 | 1931 | - | - | - | - | - | - | - | - | - | - |
| 300 | 221 | 239 | 2470 | 2470 | 0,78 | 2451 | **2451** | 0,01 | 2452 | 2452 | 0,04 | 2452 | 2452 | 0,04 | 2452 | 2452 | 0,04 |
| 300 | 112 | 177 | 3360 | 3360 | 0,03 | 3359 | **3359** | 0,00 | 3361 | 3360 | 0,06 | 3362 | 3361 | 0,09 | 3363 | 3362 | 0,12 |
| 300 | 75 | 181 | 35 | **35** | 0,00 | 35 | **35** | 0,00 | 36 | 35 | 2,86 | 38 | 37 | 8,57 | 38 | 37 | 8,57 |
| 300 | 121 | 162 | 65 | **65** | 0,00 | 65 | **65** | 0,00 | 281 | 281 | 332,31 | 283 | 282 | 335,38 | 283 | 282 | 335,38 |
| 300 | 119 | 176 | 308 | **308** | 0,00 | 308 | **308** | 0,00 | 308 | **308** | 0,00 | 309 | **308** | 0,32 | 309 | **308** | 0,32 |
| 600* | 165* | 329* | 4879 | 4879 | - | 4522 | 4494 | - | - | - | - | - | - | - | - | - | - |
| 600 | 256 | 388 | 3448 | 3448 | 2,01 | 3392 | **3380** | 0,35 | 3385 | 3384 | 0,15 | 3387 | 3384 | 0,21 | 3398 | 3388 | 0,53 |
| 600 | 120 | 288 | 9075 | 9075 | 5,81 | 8577 | **8577** | 0,00 | 8579 | 8578 | 0,02 | 8581 | 8580 | 0,05 | 8593 | 8584 | 0,19 |
| 600 | 77 | 215 | 37 | **37** | 0,00 | 37 | **37** | 0,00 | 42 | 42 | 13,51 | 106 | 43 | 186,49 | 84 | 43 | 127,03 |
| 600 | 126 | 279 | 410 | **410** | 0,00 | 410 | **410** | 0,00 | 414 | 414 | 0,98 | 416 | 415 | 1,46 | 417 | 416 | 1,71 |
| 1200* | 186* | 519* | 13067 | 13067 | - | 11281 | 10921 | - | - | - | - | - | - | - | - | - | - |
| 1200 | 263 | 666 | 9893 | 9893 | 3,95 | 9542 | **9517** | 0,26 | 9535 | 9527 | 0,19 | 9634 | 9587 | 1,23 | 9632 | 9575 | 1,21 |
| 1200 | 122 | 470 | 23815 | 23815 | 8,64 | 22100 | **21921** | 0,82 | 21932 | 21930 | 0,05 | 22268 | 22075 | 1,58 | 22434 | 22318 | 2,34 |
| 1200 | 88 | 252 | 299 | **299** | 0,00 | 299 | **299** | 0,00 | 324 | 309 | 8,36 | 1016 | 765 | 239,80 | 664 | 390 | 122,07 |
| 1200 | 130 | 420 | 555 | 555 | 8,19 | 513 | **513** | 0,00 | 528 | 526 | 2,92 | 665 | 532 | 29,63 | 779 | 679 | 51,85 |
| 1200 | 122 | 403 | 7402 | 7402 | 8,28 | 6839 | **6836** | 0,05 | 6842 | 6841 | 0,09 | 6852 | 6841 | 0,23 | 6850 | 6842 | 0,20 |
| 2400 | 188 | 738 | 31963 | 31963 | 19,69 | 27978 | 26932 | 4,77 | 26991 | 26777 | 1,07 | 27797 | **26705** | 4,09 | 27835 | 27179 | 4,23 |
| 2400 | 130 | 603 | 44941 | 44941 | 19,63 | 39127 | **37566** | 4,15 | 38293 | 38094 | 1,94 | 41268 | 39614 | 9,85 | 40402 | 38764 | 7,55 |
| 2400 | 90 | 278 | 1510 | 1510 | 164,91 | 584 | **570** | 2,39 | 2273 | 2000 | 298,77 | 2333 | 2026 | 309,30 | 1953 | 1585 | 242,63 |
| 2400 | 132 | 701 | 1058 | 1058 | 70,37 | 628 | **621** | 1,11 | 945 | 816 | 52,17 | 2319 | 1768 | 273,43 | 2244 | 1608 | 261,35 |
| 2400 | 126 | 561 | 9276 | 9276 | 12,55 | 8244 | **8242** | 0,03 | 8280 | 8259 | 0,46 | 9454 | 8839 | 14,71 | 9363 | 8775 | 13,60 |
| 4800 | 197 | 1128 | 62760 | 62760 | 10,29 | 57762 | **56902** | 1,51 | 60525 | 59580 | 6,37 | 62883 | 62395 | 10,51 | 62649 | 61138 | 10,10 |
| 4800 | 78 | 1286 | 45143 | 45143 | 10,23 | 41219 | **40953** | 0,65 | 42215 | 41925 | 3,08 | 43650 | 42643 | 6,59 | 43981 | 42781 | 7,39 |
| 4800 | 130 | 720 | 102571 | 102571 | 19,07 | 88412 | **86147** | 2,63 | 95815 | 94780 | 11,22 | 97950 | 95629 | 13,70 | 95958 | 92320 | 11,39 |
| 4800 | 91 | 294 | 197978 | 197978 | 1,56 | 195014 | **194944** | 0,04 | 208749 | 207390 | 7,08 | 200655 | 199117 | 2,93 | 199368 | 198340 | 2,27 |
| 4800 | 135 | 990 | 3194 | 3194 | 64,30 | 1958 | **1944** | 0,74 | 6803 | 6194 | 249,95 | 7279 | 6008 | 274,43 | 7831 | 6088 | 302,83 |
| 4800 | 129 | 713 | 16390 | 16390 | 8,04 | 15186 | **15171** | 0,10 | 17181 | 16745 | 13,25 | 19738 | 19062 | 30,10 | 18138 | 17749 | 19,56 |
| 9600 | 132 | 816 | 39849 | 39849 | 33,08 | 31075 | **29943** | 3,78 | 51712 | 50272 | 72,70 | 39650 | 35778 | 32,42 | 39445 | 38004 | 31,73 |
| 19200 | 132 | 908 | 89924 | 89924 | 29,04 | 73805 | **69688** | 5,91 | 208419 | 198838 | 199,07 | 108320 | 102773 | 55,44 | 108290 | 103863 | 55,39 |
| 33484 | 145 | 1032 | 141902 | 141902 | 14,87 | 127407 | **123538** | 3,13 | 621953 | 616479 | 403,45 | 237002 | 223752 | 91,85 | 222586 | 220048 | 80,18 |

*Solução infactível retornada pelos algoritmos MSVNS, BRKGA e BRKMA.

Fonte: Compilação do autor.

Pelos resultados apresentados na Tabela 5.4, percebe-se a superioridade do algoritmo ILS. Ao analisarmos a coluna *Best* dos métodos, verificamos que o ILS reportou o melhor resultado em 97% das instâncias, sendo seguido pelo JPA com 33%, pelo MSVNS e o BRKGA, os quais foram melhores em 21% e o BRKMA em 18%. Analisando a coluna *Gap*, identifica-se que o ILS possui o menor desvio em 85% das instâncias. Dessa forma, podemos concluir que o algoritmo ILS é o melhor, pois além de atingir o melhor custo em 97% das instâncias, também se mostrou o mais robusto, já que teve o menor desvio na maioria das instâncias.

A Figura 5.1 ilustra o gráfico boxplot da distribuição do gap dos métodos avaliados. Nesta figura temos o desenho de uma caixa que indica o intervalo interquartil, ou seja, o intervalo entre o primeiro e o terceiro quartil, com duas linhas horizontais, uma sólida indicando a mediana e outra pontilhada indicando a média do conjunto. As barras verticais em cada caixa indicam os valores máximos e mínimos de cada distribuição. Fora das caixas foram plotados os pontos com valores discrepantes, conhecidos como outliers, os quais estão além dos máximos de cada distribuição.

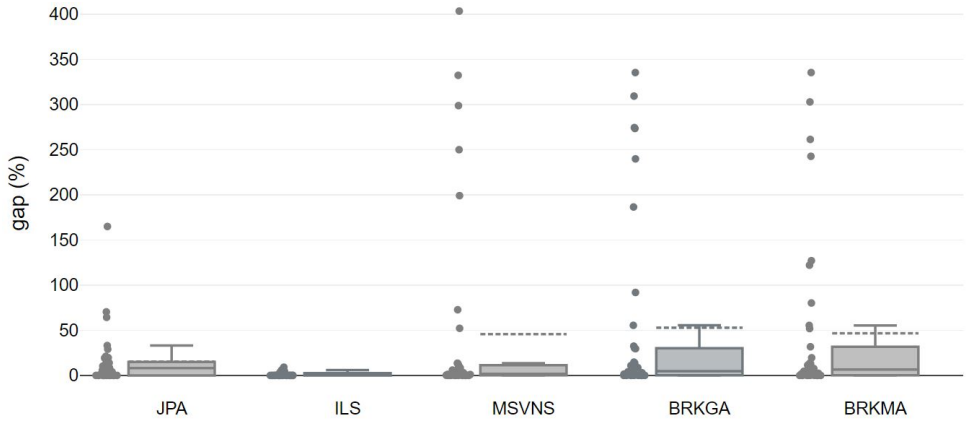



Figura 5.1: Diagrama boxplot dos resultados de gap dos algoritmos

Fonte: Compilação do autor.

Analisando o gráfico, confirma-se a superioridade do ILS para tratar o problema, já que a dispersão dos valores do gap foi muito menor que a dos outros métodos, além de não apresentar outliers. Vale também destacar o desempenho do JPA desenvolvido neste trabalho, que teve o segundo melhor desempenho. Por ser um método construtivo, é muito mais simples do que os demais métodos da literatura e, ainda assim, foi capaz de atingir melhores resultados.

### 5.2.3. Comparativo ILS × SIM-ILS

Conforme visto na subseção anterior, o algoritmo ILS obteve o melhor desempenho entre os métodos avaliados para o problema em estudo. Porém, todos os testes foram realizados com durações das tarefas determinísticas, o que nem sempre é realidade no ambiente industrial.

Por outro lado, o algoritmo SIM-ILS considera que a duração das tarefas assume valores estocásticos. Ele tem três parâmetros. O primeiro é o tipo de distribuição com a aleatoriedade que será aplicada na duração das tarefas (para mais ou para menos). A distribuição normal foi o tipo considerado e a aleatoriedade foi fixada em 20%. Por fim, os parâmetros $n_{Fast}$ e $n_{Deep}$ representam o número de avaliações da função objetivo com os valores simulados de duração das tarefas, os quais foram fixados em 100 e 1000, respectivamente. Os dois últimos valores são aqueles utilizados nos trabalhos de Guimarans *et al.* (2018) e Keenan *et al.* (2021). O *pool* de soluções foi limitado às 10 melhores encontradas.

Por fim, o valor do parâmetro $t_{\max}$, presente nos dois métodos, é o mesmo utilizado por Aquino *et al.* (2019), sendo fixado como o número de ordens de manutenção da instância, em segundos.

O resultado da comparação de desempenho entre os algoritmos ILS e SIM-ILS é reportado na Tabela 5.5. Para calcular a coluna Gap (%), foi levado em consideração somente os resultados desses dois métodos e a coluna $T_{Avg}(s)$ reporta a média de tempo para as 30 execuções de cada instância.

Tabela 5.5: Resultado da comparação entre os algoritmos ILS e SIM-ILS.

| Instância | | | ILS | | | | SIM-ILS | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | E | M | Avg | Best | Gap (%) | $T_{Avg}(s)$ | Avg | Best | Gap (%) | $T_{Avg}(s)$ |
| 150 | 148 | 120 | 1848 | 1848 | 0,00 | 39 | 1848 | 1848 | 0,00 | 88 |
| 150 | 75 | 129 | 30 | 30 | 0,00 | 31 | 30 | 30 | 0,00 | 70 |
| 150 | 102 | 91 | 3273 | 3273 | 0,00 | 27 | 3273 | 3273 | 0,00 | 41 |
| 150 | 57 | 126 | 24 | 24 | 0,00 | 26 | 24 | 24 | 0,00 | 61 |
| 150 | 92 | 93 | 49 | 49 | 0,00 | 21 | 49 | 49 | 0,00 | 50 |
| 150 | 71 | 101 | 106 | 106 | 0,00 | 24 | 106 | 106 | 0,00 | 61 |
| 300 | 158 | 179 | 1938 | 1931 | 0,35 | 300 | 2000 | 1931 | 3,59 | 300 |
| 300 | 221 | 239 | 2451 | 2451 | 0,01 | 299 | 2457 | 2452 | 0,23 | 288 |
| 300 | 112 | 177 | 3359 | 3359 | 0,00 | 298 | 3359 | 3359 | 0,00 | 284 |
| 300 | 75 | 181 | 35 | 35 | 0,00 | 261 | 35 | 35 | 0,00 | 205 |
| 300 | 121 | 162 | 65 | 65 | 0,00 | 250 | 65 | 65 | 0,00 | 270 |
| 300 | 119 | 176 | 308 | 308 | 0,00 | 266 | 308 | 308 | 0,00 | 254 |
| 600 | 165 | 329 | 4522 | 4494 | 0,62 | 600 | 4645 | 4502 | 3,36 | 600 |
| 600 | 256 | 388 | 3392 | 3380 | 0,35 | 600 | 3407 | 3383 | 0,81 | 600 |
| 600 | 120 | 288 | 8577 | 8577 | 0,00 | 600 | 8584 | 8577 | 0,08 | 600 |
| 600 | 77 | 215 | 37 | 37 | 0,00 | 600 | 37 | 37 | 0,00 | 600 |
| 600 | 126 | 279 | 410 | 410 | 0,00 | 600 | 410 | 410 | 0,00 | 600 |
| 1200 | 186 | 519 | 11281 | 10921 | 3,30 | 1201 | 11375 | 10992 | 4,15 | 1201 |
| 1200 | 263 | 666 | 9542 | 9517 | 0,26 | 1201 | 9564 | 9524 | 0,50 | 1201 |
| 1200 | 122 | 470 | 22100 | 21921 | 0,82 | 1201 | 22554 | 21942 | 2,89 | 1201 |
| 1200 | 88 | 252 | 299 | 299 | 0,00 | 1201 | 299 | 299 | 0,00 | 1201 |
| 1200 | 130 | 420 | 513 | 513 | 0,00 | 1201 | 514 | 513 | 0,22 | 1201 |
| 1200 | 122 | 403 | 6839 | 6836 | 0,05 | 1201 | 6867 | 6839 | 0,45 | 1201 |
| 2400 | 188 | 738 | 27978 | 26932 | 3,89 | 2404 | 28281 | 26989 | 5,01 | 2404 |
| 2400 | 130 | 603 | 39127 | 37566 | 4,15 | 2403 | 39717 | 38768 | 5,73 | 2403 |
| 2400 | 90 | 278 | 584 | 570 | 2,39 | 2403 | 686 | 571 | 20,36 | 2403 |
| 2400 | 132 | 701 | 628 | 621 | 1,11 | 2403 | 651 | 622 | 4,82 | 2403 |
| 2400 | 126 | 561 | 8244 | 8242 | 0,03 | 2403 | 8381 | 8243 | 1,68 | 2404 |
| 4800 | 197 | 1128 | 57762 | 56902 | 1,51 | 4816 | 57853 | 56970 | 1,67 | 4817 |
| 4800 | 78 | 1286 | 41219 | 40953 | 0,65 | 4814 | 41411 | 41000 | 1,12 | 4813 |
| 4800 | 130 | 720 | 88412 | 86147 | 2,63 | 4812 | 88613 | 86767 | 2,86 | 4812 |
| 4800 | 91 | 294 | 195014 | 194944 | 0,04 | 4811 | 195187 | 194944 | 0,12 | 4811 |
| 4800 | 135 | 990 | 1958 | 1944 | 0,74 | 4809 | 1982 | 1945 | 1,97 | 4808 |
| 4800 | 129 | 713 | 15186 | 15171 | 0,10 | 4810 | 15265 | 15170 | 0,63 | 4808 |
| 9600 | 132 | 816 | 31075 | 29943 | 3,79 | 9633 | 31147 | 29940 | 4,03 | 9640 |
| 19200 | 132 | 908 | 73805 | 69688 | 6,43 | 19287 | 74127 | 69347 | 6,89 | 19359 |
| 33484 | 145 | 1032 | 127407 | 123538 | 5,18 | 33778 | 126292 | 121134 | 4,26 | 33973 |

Fonte: Compilação do autor.

Ao avaliarmos a coluna $T_{Avg}(s)$ podemos verificar que, para instâncias com até 150 ordens de manutenção, o parâmetro de tempo não foi atingido como critério de parada para os dois métodos. Outro ponto a observar é que o tempo médio de execução para instâncias de 300 até 9600 ordens de manutenção é bem semelhante entre os dois métodos; porém, o SIM-ILS possui etapas adicionais de avaliação do custo estocástico das soluções. Essa similaridade entre eles pode ser devido a uma rápida convergência do SIM-ILS para a solução reportada, tendo assim poucas soluções no *pool* para serem avaliadas. Já para as duas maiores instâncias, verifica-se que o SIM-ILS consumiu mais tempo de processamento, o que é esperado devido às avaliações das soluções do *pool*, que é uma etapa adicional não limitada pelo parâmetro $t_{\max}$.

Na coluna *gap* de ambos os métodos, verificamos que o ILS foi mais eficiente em 62% das instâncias, o SIM-ILS em 3% e houve empate em 35% delas. Não é possível fazer uma comparação direta das soluções geradas pelos dois métodos, já que apenas no SIM-ILS são tratadas as características estocásticas do problema. No entanto, é esperado que o *gap* apresentado pelo SIM-ILS seja maior que o do ILS, conforme confirmado pela própria Figura 5.2. Somente na maior instância o SIM-ILS teve desempenho melhor do que o ILS; provavelmente devido às próprias características estocásticas do método heurístico.

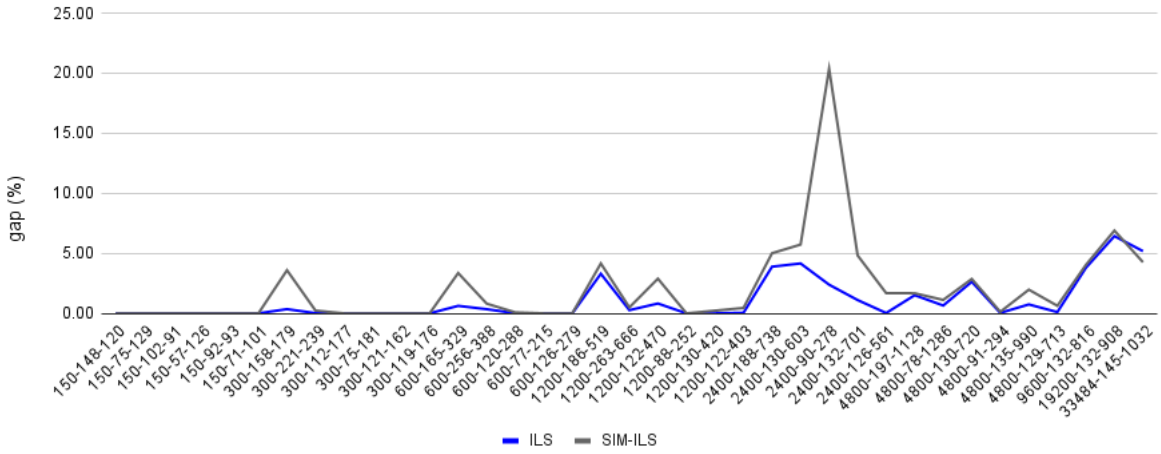



Figura 5.2: Gráfico de linhas dos resultados de *gap* dos algoritmos

Fonte: Compilação do autor.

# 6. Conclusões e trabalhos futuros

## 6.1. Conclusões

Este trabalho teve seu foco no problema de alocação de tarefas de manutenção preventiva em um planejamento de 52 semanas. Para tratá-lo, foram utilizados algoritmos heurísticos construtivos e de posicionamento de tarefas em conjunto com um algoritmo meta-heurístico ILS. O objetivo foi encontrar soluções melhores do que as apresentadas na literatura, levando em consideração a complexidade e tempo de execução das tarefas, bem como a imprevisibilidade do ambiente industrial.

Inicialmente, foi desenvolvido um novo algoritmo construtivo, nomeado JPA, capaz de testar automaticamente 384 regras de construção e retornar a melhor regra para cada instância. Para avaliá-lo, foram realizados experimentos computacionais usando as duas versões do JPA (direto e invertido), em instâncias de médio e grande porte da literatura. Foi constatado que a versão invertida do JPA obteve o melhor desempenho em todas as instâncias avaliadas, pois foi a que gerou os menores custos. Além disso, observou-se que o método proposto foi capaz de construir soluções de boa qualidade em tempo computacional muito menor que aqueles dos algoritmos meta-heurísticos da literatura. Verificou-se, também, que ele gerou algumas soluções melhores do que as existentes, inclusive para a maior instância do conjunto, que foi extraída da indústria.

Na sequência, foi desenvolvido o algoritmo meta-heurístico ILS para aprimorar as soluções geradas pelo método heurístico construtivo. Nos experimentos computacionais realizados, a solução fornecida pelo melhor algoritmo construtivo, isto é, o algoritmo com a melhor regra para cada instância usando o JPA invertido, foi utilizado como ponto de partida do método ILS. Observou-se que o algoritmo ILS apresentou desempenho superior em relação aos outros algoritmos meta-heurísticos da literatura, sendo capaz de atingir o melhor resultado em 97% das instâncias.

Destaca-se, no entanto, que o método ILS é determinístico, assim como os outros métodos da literatura, e não leva em consideração as incertezas que podem ocorrer durante a execução de uma tarefa, o que pode levar a um planejamento insuficiente e a um tempo de execução total mais longo do que o previsto.

Dessa forma, apesar de terem sido geradas soluções de alta qualidade com o método ILS, pode ser que elas não sejam aplicáveis em um ambiente real devido às incertezas existentes na operação de manutenção. Para contornar essa situação, foi proposto o método SIM-ILS, um algoritmo simheurístico capaz de levar em consideração esses aspectos estocásticos. Nesse método, as soluções são submetidas a uma avaliação estocástica com simulações de Monte Carlo. As simulações são feitas na duração das tarefas com o objetivo de representar o ambiente real da indústria.

Os resultados alcançados pelo método simheurístico apresentaram um custo levemente

superior ao método meta-heurístico ILS determinístico. Porém, deve-se ressaltar que as soluções por ele geradas captam as incertezas da operação de manutenção, o que não ocorre com a sua versão determinística. Sendo assim, essas soluções são mais adequadas para aplicação em ambientes reais da indústria.

## 6.2. Propostas de continuidade

Considerando os resultados obtidos neste trabalho, é possível identificar algumas oportunidades para aprimorar ainda mais o planejamento e controle da manutenção preventiva em indústrias.

Uma das propostas para dar continuidade a este trabalho é testar outros métodos de solução para o problema. Embora o algoritmo ILS tenha apresentado um desempenho superior em relação aos outros algoritmos testados neste estudo, há uma série de outras meta-heurísticas que ainda podem ser exploradas para este problema.

Outra proposta é tratar como flexíveis as janelas de realização das ordens de manutenção. Neste caso, as tarefas executadas fora da janela seriam penalizadas. Esta variante do problema abordado pode reduzir a necessidade de contratação de mão de obra terceirizada para executar tarefas não alocadas dentro das suas janelas.

Dessa forma, essas propostas podem contribuir para reduzir o custo total de manutenção, beneficiando as empresas e os consumidores finais.

## 6.3. Publicação

A seguir é listada a produção gerada com os resultados intermediários obtidos durante a execução deste trabalho.

1. **Título:** Um estudo de algoritmos heurísticos construtivos e de posicionamento para um problema de planejamento de ordens de manutenção.

   **Autores:** Diego Gomes Coelho, Luciano Perdigão Cota e Marcone Jamilson Freitas Souza.

   **Evento:** XXIV Congresso Brasileiro de Automática (CBA 2022).

   **Local:** Fortaleza, Brasil.

   **Disponível em:** `https://www.sba.org.br/cba2022/wp-content/uploads/artigos_cba2022/paper_933.pdf`

# Referências Bibliográficas


ALMAKHLAFI, A., KNOWLES, J. "Iterated Local Search for the Generator Maintenance Scheduling Problem", pp. 708–742, 2015.

AQUINO, R. D., SOUZA, M. J. F. "Instances for the LTPMSP Problem". 2016. `http://www.decom.ufop.br/prof/marcone/projects/LTPMSP/instances2016.zip`. Acessado em 1 de Abril de 2022.

AQUINO, R. D., CHAGAS, J. B. C., SOUZA, M. J. F. "A Variable Neighborhood Search Algorithm for the Long-term Preventive Maintenance Scheduling Problem". Em: *Proceedings of the 20th Int. Conf. on Enterprise Information Systems - Volume 1: ICEIS,*, pp. 303–310. INSTICC, SciTePress, 2018. doi: 10.5220/0006689703030310.

AQUINO, R. D., CHAGAS, J. B. C., SOUZA, M. J. F. "Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo: Um Estudo de Caso Industrial de Larga Escala", *Pesquisa Operacional para o Desenvolvimento*, v. 11, n. 3, pp. 159–182, 2019.

GUIMARANS, D., DOMINGUEZ, O., PANADERO, J., et al.. "A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times", *Simulation Modelling Practice and Theory*, v. 89, pp. 1–14, 2018. ISSN: 1569-190X. doi: https://doi.org/10.1016/j.simpat.2018.09.004.

JUAN, A., LI, Y., AMMOURIOVA, M., et al.. "Simheuristics: An Introductory Tutorial". pp. 1325–1339, 12 2022. doi: 10.1109/WSC57314.2022.10015318.

KEENAN, P., PANADERO, J., JUAN, A. A., et al.. "A strategic oscillation simheuristic for the Time Capacitated Arc Routing Problem with stochastic demands", *Computers Operations Research*, v. 133, pp. 105377, 2021. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2021.105377.

KIRKPATRICK, S., GELATT JR, C. D., VECCHI, M. P. "Optimization by simulated annealing", *science*, v. 220, n. 4598, pp. 671–680, 1983.

KIZYS, R., DOERING, J., JUAN, A. A., et al.. "A simheuristic algorithm for the portfolio optimization problem with random returns and noisy covariances", *Computers Operations Research*, v. 139, pp. 105631, 2022. ISSN: 0305-0548. doi: https://doi.org/10.1016/j.cor.2021.105631.

LOURENÇO, H., MARTIN, O., STÜTZLE, T. "A beginner's introduction to Iterated Local Search", pp. 1–11, 06 2001.

LÓPEZ-IBÁÑEZ, M., DUBOIS-LACOSTE, J., PÉREZ CÁCERES, L., et al.. "The irace package: Iterated racing for automatic algorithm configuration", *Operations Research Perspectives*, v. 3, pp. 43–58, 2016.

MARTÍ, R., RESENDE, M. G., RIBEIRO, C. C. "Multi-start methods for combinatorial optimization", *European Journal of Operational Research*, v. 226, n. 1, pp. 1–8, 2013.

MARTINEZ, C., LOISEAU, I., RESENDE, M. G., et al.. "BRKGA algorithm for the capacitated arc routing problem", *Electronic Notes in Theoretical Computer Science*, v. 281, pp. 69–83, 2011.

MENA, R., VIVEROS, P., ZIO, E., et al.. "An optimization framework for opportunistic planning of preventive maintenance activities", *Reliability Engineering & System Safety*, v. 215, pp. 107801, 2021.

MLADENOVIĆ, N., HANSEN, P. "Variable neighborhood search", *Computers & operations research*, v. 24, n. 11, pp. 1097–1100, 1997.

NERI, F., COTTA, C. "Memetic algorithms and memetic computing optimization: A literature review", *Swarm and Evolutionary Computation*, v. 2, pp. 1–14, 2012.

PASSMARK. "Cpu benchmarks". Março 2022. Disponível em: <https://www.cpubenchmark.net/compare/Intel-Xeon-E5-2660-v2-vs-Intel-Xeon-E5-2699-v4/2184vs2753>.

PISINGER, D., ROPKE, S. "Large neighborhood search". Em: *Handbook of metaheuristics*, Springer, pp. 399–419, 2010.

SANTOS, M. S., PINTO, T. V., ÊNIO LOPES JÚNIOR, et al.. "Simheuristic-based decision support system for efficiency improvement of an iron ore crusher circuit", *Engineering Applications of Artificial Intelligence*, v. 94, pp. 103789, 2020. ISSN: 0952-1976. doi: https://doi.org/10.1016/j.engappai.2020.103789.

SARAIVA, J. T., PEREIRA, M. L., MENDES, V. T., et al.. "A simulated annealing based approach to solve the generator maintenance scheduling problem", *Electric Power Systems Research*, v. 81, n. 7, pp. 1283–1291, 2011.

SHARMA, A., YADAVA, G., DESHMUKH, S. “A literature review and future perspectives on maintenance optimization”, *Journal of Quality in Maintenance Engineering*, v. 17, n. 1, 2011.

SIMÕES, J. M., GOMES, C. F., YASIN, M. M. “A literature review of maintenance performance measurement: A conceptual framework and directions for future research”, *Journal of Quality in Maintenance Engineering*, v. 17, n. 2, 2011.

VIANA, H. *PCM, planejamento e controle da manutenção*, v. 1. Qualitymark, 2006.

VIVEROS, P., MIQUELES, L., MENA, R., et al.. “Opportunistic Strategy for Maintenance Interventions Planning: A Case Study in a Wastewater Treatment Plant”, *Applied Sciences*, v. 11, n. 22, pp. 10853, 2021.

WOLLER, D., KULICH, M. “The ALNS Metaheuristic for the Maintenance Scheduling Problem”. Em: *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO 2021*, v. 18, pp. 156–164, online, 6-8 July 2021. SciTePress.