Um estudo de algoritmos heurísticos construtivos e de posicionamento para um problema de planejamento de ordens de manutenção

Diego Gomes Coelho [*] **Luciano Perdigão Cota** [**]
Marcone Jamilson Freitas Souza [***]

[*] *Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto e Instituto Tecnológico Vale, MG, (e-mails: diego.coelho@aluno.itv.org)*
[**] *Instituto Tecnológico Vale, MG, (e-mail: luciano.p.cota@itv.org)*
[***] *Departamento de Computação, Universidade Federal de Ouro Preto, MG, (e-mail: marcone@ufop.edu.br)*

Abstract: In this work, we study the applications of constructive heuristic and job positioning algorithms for a long-term maintenance scheduling problem. This problem is complex, where it is necessary to schedule preventive maintenance orders for the available work teams for a time horizon. Up to date, constructive heuristic and job positioning algorithms have not been explored in the literature for this problem. This study proposes a method that can generate a set of constructive algorithms based on several construction rules and returns the best of them for each instance. We use large instances to evaluate the proposed method. In addition, we compare its results with those of meta-heuristic algorithms from the literature. The proposed method demanded much less computational time. Moreover, it found the best result in 49% of the instances, including the largest with more than 33,000 jobs.

Resumo: Neste trabalho é realizado um estudo da aplicação de algoritmos heurísticos construtivos e de posicionamento de tarefas para um problema de planejamento de ordens de manutenção de longo prazo. Trata-se de um problema complexo em que é necessário realizar a programação de ordens de manutenção preventiva para as equipes de trabalho disponíveis em um determinado horizonte de tempo. Até o presente momento, algoritmos heurísticos construtivos e de posicionamento de tarefas não foram explorados na literatura para este problema. Neste estudo é proposto um método capaz de gerar um conjunto de algoritmos construtivos baseado em diversas regras de construção e retornar a melhor delas para cada instância. Para avaliá-lo foram utilizadas instâncias de grande porte e os seus resultados foram comparados com os de algoritmos meta-heurísticos da literatura. O método proposto demandou um tempo computacional muito menor. Além disso, ele encontrou o melhor resultado em 49% das instâncias, incluindo a maior delas com mais de 33 mil tarefas.

Keywords: Long-term maintenance scheduling problem, Combinatorial optimization; constructive heuristics.

Palavras-chave: Sequenciamento de ordens de manutenção de longo prazo; Otimização combinatória; Heurísticas construtivas.

1. INTRODUÇÃO

O planejamento e controle da manutenção é muito importante para o setor industrial, pois tem por objetivo manter a disponibilidade dos ativos com o menor custo possível. Para atingir esse objetivo a manutenção preventiva é comumente utilizada. A manutenção preventiva compreende intervenções programadas em ativos que não estejam em falha, de acordo com critérios predeterminados, buscando a reduzir a probabilidade de falha nos ativos (Viana, 2006).

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.

Nos trabalhos de Aquino et al. (2019) e Aquino et al. (2018) são abordados o problema de alocação de ordens de manutenção preventiva de máquinas industriais às equipes de trabalho disponíveis ao longo de um planejamento de

[⋆] Os autores agradecem à Universidade Federal de Ouro Preto, ao Instituto Tecnológico Vale e às agências CNPq (processo 303266/2019-8), FAPEMIG (processo PPM-CEX 676/17) e CAPES (código de financiamento 001), pelo apoio ao desenvolvimento deste trabalho.

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 minerais. Para resolver o problema são propostos 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 and 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 and Cotta, 2012). Dentre estes, o MS e o BRMKA obtiveram os melhores resultados.

Woller and 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 and Ropke, 2010).

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.

Em Mena et al. (2021), os autores propõem um *framework* para a alocação de ordens de manutenção com o objetivo de minimizar o número de paradas dos equipamentos nos quais tarefas serão realizadas. O *framework* é baseado em um modelo de programação linear inteira mista.

No trabalho de Aquino et al. (2019), que trata da alocação de ordens manutenção do plano de 52 semanas em indústrias de mineração, verificou-se que os algoritmos meta-heurísticos propostos demandam um longo tempo de execução para encontrar uma boa solução. Além disso, observou-se que as heurísticas construtivas e o algoritmo responsável pelo posicionamento de tarefas nos algoritmos meta-heurísticos propostos não foram explorados. Para preencher esta lacuna, o presente trabalho tem como objetivo realizar um estudo de novos algoritmos heurísticos construtivos e de posicionamento de tarefas para este problema.

O restante deste trabalho está estruturado como segue. Na Seção 2 é realizada a descrição do problema. Na Seção 3 é detalhado o algoritmo de posicionamento proposto. Os algoritmos heurísticos construtivos propostos são apresentados na Seção 4. Na Seção 5 são reportados os experimentos computacionais. Por fim, na Seção 6 são apresentadas as conclusões deste estudo e indicadas sugestões para trabalhos futuros.

2. CARACTERIZAÇÃO DO PROBLEMA

A seguir, descreve-se a caracterização do problema e o modelo de programação linear inteira mista proposto por Aquino et al. (2019) para tratar o problema em estudo.

Seja $E = \{1, 2, \cdots, Q\}$ o conjunto de Q máquinas industriais em que as ordens de manutenção devam ser executadas. Seja também $\mathscr{T} = \{1, 2, \cdots, N\}$ o conjunto de N ordens de manutenção preventiva, e $\mathscr{W} = \{1, 2, \cdots, M\}$ o conjunto de M equipes de manutenção disponíveis para realização das ordens. Cada tarefa de manutenção preventiva $i \in \mathscr{T}$ deve ser realizada em uma única máquina E_i e somente uma equipe de trabalho é suficientemente capaz de executá-la, porém duas ou mais manutenções diferentes, podem ser associadas na mesma máquina. No entanto, somente uma manutenção preventiva pode ser executada por vez em uma máquina específica.

Além disso, cada ordem de manutenção i possui uma duração P_i e também uma janela de tempo para ser executada $[\, e_i, l_i \,]$, sendo e_i o inicio e l_i o final dessa janela. A penalidade por não execução é dada por w_i. Cada manutenção preventiva $i \in \mathscr{T}$ requer uma equipe de trabalho com uma habilidade específica, sendo que $\mathscr{W}_i \subseteq \mathscr{W}$ indica o conjunto de equipes com habilidades para executá-la.

Cada equipe de manutenção $k \in \mathscr{W}$ só pode executar uma manutenção preventiva por vez e está disponível até o instante h_k para execução das tarefas. O conjunto $\mathscr{T}_k \subseteq \mathscr{T}$ indica a habilidade de cada equipe k para execução das tarefas.

Em adição aos atributos das tarefas de manutenção e das equipes de trabalho, foram definidas as seguintes variáveis de decisão:

- x_{ij}^k: 1 se a manutenção i é executada imediatamente antes da 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 (1)-(17):

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

$$\sum_{k \in \mathscr{W}_i} y_{ik} \leq 1 \qquad \forall i \in \mathscr{T} \tag{2}$$

$$\sum_{i \in \mathscr{T}_k \cup \{0\} \setminus \{j\}} x_{ij}^k = y_{jk} \qquad \forall j \in \mathscr{T}, k \in \mathscr{W}_j \tag{3}$$

$$\sum_{j \in \mathscr{T}_k} x_{0j}^k = z_k \qquad \forall k \in \mathscr{W} \tag{4}$$

$$\sum_{i \in \mathscr{T}_k \cup \{0\} \setminus \{l\}} x_{il}^k = \sum_{j \in \mathscr{T}_k \cup \{0\} \setminus \{l\}} x_{lj}^k \forall k \in \mathscr{W}, l \in \mathscr{T}_k \tag{5}$$

$$c_{0k} = 0 \quad \forall k \in \mathscr{W} \tag{6}$$

$$c_{jk} \geq c_{ik} + P_j - M'_{ij}(1 - x^k_{ij}) \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \cup \{0\}, j \in \mathscr{T}_k \tag{7}$$

$$c_{ik} \geq (e_i + P_i)\, y_{ik} \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \tag{8}$$

$$c_{ik} \leq l_i \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \tag{9}$$

$$c_{jk'} \geq c_{ik} + P_j - M'_{ij}(1 - r_{ij}) \quad \forall k \in \mathscr{W}, k' \in \mathscr{W}, i \in \mathscr{T}_k, j \in \mathscr{T}_{k'}, k \neq k', i < j, E_i = E_j \tag{10}$$

$$c_{jk'} \leq c_{ik} - P_i + M''_{ij} r_{ij} \quad \forall k \in \mathscr{W}, k' \in \mathscr{W}, i \in \mathscr{T}_k, j \in \mathscr{T}_{k'}, k \neq k', i < j, E_i = E_j \tag{11}$$

$$c_{ik} \leq h_k \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \tag{12}$$

$$x^k_{ij} \in \{0,1\} \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \cup \{0\}, j \in \mathscr{T}_k \cup \{0\} \tag{13}$$

$$y_{ik} \in \{0,1\} k \in \mathscr{W}, \quad \forall i \in \mathscr{T}_k \cup \{0\} \tag{14}$$

$$z_k \in \{0,1\} \quad \forall k \in \mathscr{W} \tag{15}$$

$$c_{ik} \geq 0 \quad \forall k \in \mathscr{W}, i \in \mathscr{T}_k \tag{16}$$

$$r_{ij} \in \{0,1\} \quad \forall i \in \mathscr{T}, j \in \mathscr{T}, i < j, E_i = E_j \tag{17}$$

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 (1) busca reduzir o número de equipes de manutenção necessárias para executar o maior número possível de ordens de manutenção preventiva, através da minimização das penalidades. As restrições (2) garantem que cada tarefa seja executada por no máximo uma equipe. As restrições (3) asseguram que se a equipe de trabalho k executar a tarefa j, essa tarefa deve constar no agendamento de trabalho da da equipe k. Para o conjunto de restrições da equação (4), se pelo menos uma tarefa for programada para equipe de trabalho k, essa equipe será utilizada. As restrições (5) asseguram o sequenciamento das tarefas de manutenção executadas por uma equipe de trabalho. As restrições (6) garantem que as equipes terminam a tarefa fictícia 0 no instante 0. As restrições (7) garantem que uma tarefa j termina somente após a conclusão da tarefa imediatamente predecessora i mais a sua duração. As restrições (8) e (9) garantem que toda manutenção seja executada na sua respectiva janela de tempo. As restrições (10) e (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 (12) asseguram que o instante de conclusão da tarefa i não exceda o instante limite de disponibilidade da equipe k. Finalmente, as restrições (13) a (17) definem o domínio das variáveis de decisão.

3. ALGORITMO DE POSICIONAMENTO PROPOSTO

Nesta seção apresenta-se o algoritmo de posicionamento de tarefas (JPA, do inglês *Job Positioning Algorithm*) proposto para o problema. O JPA tem por finalidade fazer o posicionamento das tarefas de manutenção para execução, e calcular o custo final desse planejamento. O pseudocódigo do JPA é apresentado pelo Algoritmo 1.

Algoritmo 1: JPA - *Job Positioning Algorithm*

```
Entrada: Solução s, Tarefas, Equipes
Saída: Custo
1  Custo ← 0;
2  para tarefas i até n faça
3      alocou ← falso;
4      Procurar janela de tempo compatível entre tarefa
         e máquina predeterminada;
5      se intervalo compatível então
6          para Equipes de k até m faça
7              Verificar habilidade da equipe;
8              se habilidade compatível então
9                  Procurar janela de tempo compatível
                     entre tarefa, máquina e equipe;
10                 se intervalo compatível então
11                     Alocar tarefa i na equipe k;
12                     Alocar tarefa i na máquina
                         pré-determinada;
13                     alocou ← verdadeiro;
14                     se equipe não utilizada então
15                         Custo ← Custo + 1;
16                     fim
17                 fim
18             fim
19         fim
20     fim
21     se alocou = falso então
22         Custo ← Custo + w_i;
23     fim
24 fim
25 retorna Custo;
```

Inicialmente, o JPA identifica os intervalos de tempo disponíveis na máquina onde a tarefa será executada e na equipe candidata a executar a tarefa. A seguir, verifica-se a compatibilidade desses intervalos com os parâmetros da tarefa, que são: a janela de execução (início e fim) e a duração da tarefa. Caso os intervalos sejam compatíveis com a tarefa, esse intervalo é bloqueado, tanto na máquina, quanto na equipe, assim, a tarefa é alocada para execução. Caso os intervalos não sejam compatíveis, será verificada a compatibilidade com as outras equipes. Caso não seja identificada nenhuma equipe para realizar a tarefa, a penalidade de não execução será acrescida no valor da função objetivo.

Para ilustrar o funcionamento do JPA, considere um exemplo com quatro tarefas, uma máquina e duas equipes de trabalho. A Tabela 1 descreve as quatro tarefas de manutenção para uma máquina e a sequência destas tarefas. Já a Tabela 2 apresenta as características das duas equipes disponíveis para execução das tarefas.

Tabela 1. Tarefas de Manutenção

Máquina: Britadeira					
#Tarefa	Habilidade	Janela Início [e]	Janela Fim [l]	Duração	Peso [ω]
1	Mecânica	1	6	2	10
2	Mecânica	3	10	3	20
3	Elétrica	4	9	3	10
4	Elétrica	1	4	2	20

Tabela 2. Equipes de Manutenção

Equipe	Habilidade $\mathcal{T}_k$	Disponibilidade Máxima $[h]$
1	Mecânica	10
2	Elétrica	10

Na Tabela 3 reporta-se o resultado da aplicação do JPA para este exemplo. Na tabela verifica-se a representação das janelas de tempo da máquina e das equipes.

Tabela 3. Posicionamento de tarefas

	1	2	3	4	5	6	7	8	9	10
Britadeira	1	1	2	2	2	3	3	3		
Equipe 1	1	1	2	2	2					
Equipe 2						3	3	3		

No resultado foi possível posicionar três das quatro tarefas de manutenção e foram utilizadas as duas equipes disponíveis. Dessa maneira é possível fazer o cálculo da função objetivo, sendo a soma o uso das duas equipes com a penalidade da tarefa não executada, $Custo = 2 + 20 = 22$.

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 2: [1, 2, 3, 4]. Para verificar essa influência, pode-se classificar de maneira crescente a coluna "Janela Fim [l]" da Tabela 2, 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.

Tabela 4. Posicionamento de tarefas

	1	2	3	4	5	6	7	8	9	10
Britadeira	4	4	1	1	3	3	3	2	2	2
Equipe 1			1	1				2	2	2
Equipe 2	4	4			3	3	3			

Com essa nova sequência de tarefas, foi possível realizar o posicionamento de todas elas, gerando um melhor valor de função objetivo. Assim, a função objetivo irá somar apenas o custo de utilização das duas equipes, $Custo = 2$. Portanto, fica demonstrado que a ordem em que as tarefas são submetidas para o algoritmo JPA tem influencia no valor da função da objetivo.

A partir do JPA proposto, apresentado no Algoritmo 1, foi proposto também uma nova variante deste JPA, cuja a única diferença se dá na opção de posicionamento da tarefa dentro do intervalo disponível. As versões são aqui denominadas como JPA direto, para o JPA original, e JPA invertido para a nova variante.

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

O JPA direto faz o posicionamento da tarefa do exemplo conforme ilustrado na Figura 1, inserindo a tarefa no início do intervalo disponível.

Usando os mesmos dados do exemplo anterior, o JPA invertido faz o posicionamento da tarefa conforme a Figura 2, inserindo cada tarefa no final do intervalo disponível.

Figura 1. Posicionamento de tarefa

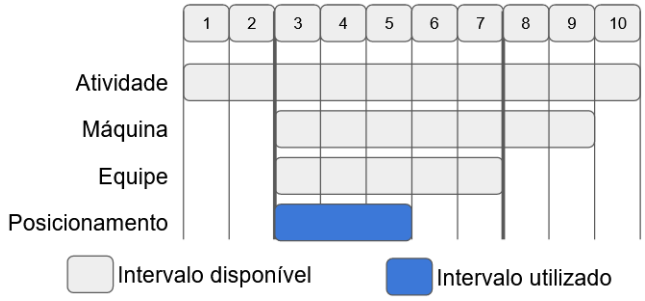
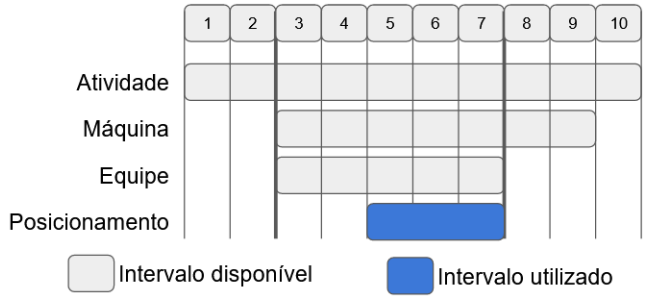


Figura 2. Posicionamento de tarefa



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

4. HEURÍSTICAS CONSTRUTIVAS PROPOSTAS

Neste trabalho é utilizada a representação de solução do problema proposta por Aquino et al. (2019). A representação da solução é 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 a frente está uma dada tarefa na sequência, maior a sua prioridade.

Para gerar as soluções são utilizados algoritmos heurísticos construtivos. 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 beneficio de sua inserção. Dessa forma, a cada passo do algoritmo, somente o elemento com maior beneficio é inserido na sequência.

No exemplo das tabelas 1 e 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]". 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 5, um exemplo de instância com um conjunto maior de tarefas de manutenção a serem executadas.

Tabela 5. Características das tarefas de manutenção

#Tarefa	Habilidade	Máquina	[e]	[l]	Duração	[ω]
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

Nesta tabela, é possível verificar que para os vários parâmetros das tarefas de manutenção, temos valores que se repetem, como o peso ([ω]), 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 5, podemos estabelecer 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 ([ω]);
(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 5, chega-se à seguinte solução:

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

Para facilitar o entendimento, podemos verificar na Tabela 6 os parâmetros das tarefas classificadas geradas na solução s''.

Tabela 6. Tarefas de manutenção ordenadas conforme regras estabelecidas acima

#Tarefa	Habilidade	Máquina	[e]	[l]	Duração	[ω]
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

Ao avaliarmos os parâmetros das tarefas de manutenção, na Tabela 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.

- 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 [ω] - Peso por não execução da tarefa.

Neste trabalho serão 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 um 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 - Classificação crescente;
- 1 - Classificação decrescente.

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

(1) Critério Penalidade ω (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 de maneira resumida:

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

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 sequencial, 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 $24 \times 16 = 384$ algoritmos construtivos diferentes.

No Algoritmo 2 apresenta-se o pseudocódigo do método responsável por gerar os 384 algoritmos construtivos e avaliá-los. Ao final, o método retorna a melhor solução construída, o seu custo e qual dos algoritmos obteve o melhor desempenho.

Algoritmo 2: *Best Constructive Algorithm Generator*

Entrada: ColunasChave, Tarefas, Equipes, JPA
Saída: Solução s, Custo, Melhor Algoritmo construtivo R

1 $n \leftarrow calculaQtde(ColunasChave)$;
2 $Custo \leftarrow 0$;
3 $Custo' \leftarrow$ Valor suficientemente grande;
4 **para** $i \leftarrow 1$ *até* $n!$ **faça**
5 $\quad Combinacao[i] \leftarrow geraComb(i, ColunasChave)$;
6 $\quad$ **para** $j \leftarrow 1$ *até* 2^n **faça**
7 $\quad\quad Variacao[j] \leftarrow geraVar(j)$;
8 $\quad\quad R \leftarrow (Combinacao[i], Variacao[j])$;
9 $\quad\quad s \leftarrow constroiSol(R)$;
10 $\quad\quad Custo \leftarrow JPA(s)$;
11 $\quad\quad$ **se** $Custo < Custo'$ **então**
12 $\quad\quad\quad s' \leftarrow s$;
13 $\quad\quad\quad Custo' \leftarrow Custo$;
14 $\quad\quad\quad R' \leftarrow R$;
15 $\quad\quad$ **fim**
16 $\quad$ **fim**
17 **fim**
18 $s \leftarrow s'$;
19 $Custo \leftarrow Custo'$;
20 $R \leftarrow R'$
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 regra utilizada para construí-la.

5. EXPERIMENTOS COMPUTACIONAIS

O JPA e os algoritmos heurísticos construtivos usando o JPA 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 and 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; *iii*) 91 a 1286 máquinas.

Para cada instância submetida, 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 algoritmo construtivo obteve o melhor desempenho e qual foi o seu custo.

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

A Tabela 7 reporta os resultados desses algoritmos construtivos. 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 gerado pelo melhor 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 para gerar o algoritmo construtivo.

Ao avaliar os resultados apresentados na Tabela 7, é possível verificar que o algoritmo proposto (Algoritmo 2) constrói uma boa solução, a melhor dentre as 384 soluções construídas para cada JPA, em reduzido tempo computacional. 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 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.

A seguir, na Tabela 8, 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. (18):

$$RPD_i = \frac{Obj_i^{Alg} - Obj_i^{Best}}{Obj_i^{Best}} \tag{18}$$

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.

Pelos resultados apresentados na Tabela 8, podemos verificar que o melhor algoritmo construtivo proposto demanda um tempo muito menor que os algoritmos meta-heurísticos para gerar uma boa solução. 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. Este resultado habilita o uso do algoritmo heurístico construtivo em aplicações industriais.

Sobre a qualidade das soluções geradas, o melhor construtivo proposto 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.

Tabela 7. 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]

Tabela 8. 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	5,24	1756	63,8	0,00	1756	63,8	0,00	1756	63,8	0,00
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	13,91	1776	127,6	0,00	1932	127,6	8,78	1932	127,6	8,78
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	10,66	4409	255,1	0,00	4545	255,1	3,08	4609	255,1	4,54
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	21,17	10784	510,3	0,00	11035	510,3	2,33	11077	510,3	2,72
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

Portanto, verifica-se a importância do estudo de heurísticas construtivas e de algoritmo de posicionamento de tarefas no problema em questão.

6. CONCLUSÕES

O presente trabalho abordou o problema de programação de ordens de manutenção preventiva com o objetivo de executar o maior número de tarefas com a menor quantidade de equipes. Na literatura, alguns estudos propuseram algoritmos meta-heurísticos baseados em busca local e busca populacional para tratá-lo. Contudo, foi observado que, apesar da relevância das heurísticas construtivas para gerar rapidamente soluções iniciais e da existência de diferentes possibilidades de posicionar as tarefas de uma solução produzida, esses aspectos não foram explorados.

Para preencher esta lacuna, foi desenvolvido um método capaz de gerar dinamicamente o melhor algoritmo heurístico construtivo baseado em 384 regras de construção. Para avaliá-lo, foram realizados experimentos computacionais com instâncias de grande porte da literatura. Além disso, os resultados do método proposto foram comparados com os de algoritmos meta-heurísticos estado da arte para o problema.

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 o método construtivo proposto gerou algumas soluções melhores que as existentes, inclusive, na maior instância do conjunto, que foi extraída da indústria.

Como trabalhos futuros, propõem-se avaliar o uso do método proposto para gerar a solução inicial e posicionar as tarefas de algoritmos meta-heurísticos. Sugere-se, também, avaliar o uso de *simheuristic* para tratar características estocásticas ainda não estudadas neste problema.

REFERÊNCIAS

Aquino, R.D., Chagas, J.B.C., and Souza, M.J.F. (2018). A variable neighborhood search algorithm for the long-term preventive maintenance scheduling problem. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, 303–310. INSTICC, SciTePress. doi:10.5220/0006689703030310.

Aquino, R.D. and Souza, M.J.F. (2016). Instances for the LTPMSP problem. `http://www.decom.ufop.br/prof/marcone/projects/LTPMSP/instances2016.zip`. Acessed on April 1, 2022.

Aquino, R.D., Chagas, J.B.C., and Souza, M.J.F. (2019). 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*, 11(3), 159–182.

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

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

Martinez, C., Loiseau, I., Resende, M.G., and Rodriguez, S. (2011). Brkga algorithm for the capacitated arc routing problem. *Electronic Notes in Theoretical Computer Science*, 281, 69–83.

Mena, R., Viveros, P., Zio, E., and Campos, S. (2021). An optimization framework for opportunistic planning of preventive maintenance activities. *Reliability Engineering & System Safety*, 215, 107801.

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

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

PassMark (2022). Cpu benchmarks. URL `https://www.cpubenchmark.net/compare/Intel-Xeon-E5-2660-v2-vs-Intel-Xeon-E5-2699-v4/2184vs2753`.

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, 399–419. Springer.

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

Simões, J.M., Gomes, C.F., and Yasin, M.M. (2011). A literature review of maintenance performance measurement: A conceptual framework and directions for future research. *Journal of Quality in Maintenance Engineering*, 17(2). Disponível em `https://www.emerald.com/insight/content/doi/10.1108/13552511111134565/full/html`.

Viana, H. (2006). *PCM, planejamento e controle da manutenção*, volume 1. Qualitymark.

Viveros, P., Miqueles, L., Mena, R., and Kristjanpoller, F. (2021). Opportunistic strategy for maintenance interventions planning: A case study in a wastewater treatment plant. *Applied Sciences*, 11(22), 10853.

Woller, D. and Kulich, M. (2021). The ALNS metaheuristic for the maintenance scheduling problem. In *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO 2021*, volume 18, 156–164. SciTePress, online.