

MODELOS HEURÍSTICOS E META-HEURÍSTICOS PARA A RESOLUÇÃO DE UM PROBLEMA DE SEQUENCIAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA DE LONGO PRAZO

Arthur Almeida Santos

Universidade Federal de Ouro Preto - UFOP
Rua Trinta e Seis, Nº 115, Loanda, 35931-008, João Monlevade - MG
arthur.jf.mg@hotmail.com

Alexandre Xavier Martins

Universidade Federal de Ouro Preto - UFOP
Rua Trinta e Seis, Nº 115, Loanda, 35931-008, João Monlevade - MG
xmartins@ufop.edu.br

Marcone Jamilson Freitas Souza

Universidade Federal de Ouro Preto - UFOP
Campus Universitário, Morro do Cruzeiro, 35400-000, Ouro Preto - MG
marcone@ufop.edu.br

RESUMO

O sucesso de uma empresa requer o bom funcionamento e a confiabilidade de seus sistemas com máquinas e equipamentos em bom estado. Para isto, é essencial um bom plano de manutenção preventiva, que tende a ficar mais complexo conforme aumenta o número de equipamentos e o horizonte de planejamento. Portanto, o presente trabalho tem como objetivo obter uma solução viável que gere redução de manutenções não realizadas e de custo com equipes em um Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP) por meio de meta-heurísticas em instâncias de grande complexidade. O trabalho se inicia com o desenvolvimento da heurística construtiva e de alocação, seguido do desenvolvimento de heurísticas de busca local e de meta-heurísticas, com a comparação destas com os valores disponíveis na literatura. Para a calibragem e validação das meta-heurísticas foram resolvidas instâncias fictícias pequenas. Após a calibragem, as meta-heurísticas foram aplicadas na resolução de instâncias maiores e a real. Como resultado, o *Iterated Local Search* (ILS) foi a meta-heurística de melhor performance e o resultado obtido para a instância real foi 40,5% melhor que o apresentado na literatura.

PALAVRAS CHAVE. Planejamento de manutenção de longo prazo, GRASP, *Simulated Annealing*, *Iterated Local Search*, Meta-heurísticas.

ABSTRACT

For the success of a company, the good functioning and reliability of its systems are essential. To keep this functioning properly, machines and equipment must always be in good condition, and for this, a good preventive maintenance plan is essential. The maintenance plan tends to become

more complex as the number of equipment and the planning horizon increases. Therefore, the present paper aims to obtain a feasible solution that produces a reduction of not executed maintenances and costs with teams in a Long Term Preventive Maintenance Order Planning Problem (PPOM-PLP) through meta-heuristics. The work begins with the development of constructive heuristics and allocation heuristics. Subsequently, the development of local search heuristics and, finally, the development of meta-heuristics and comparisons with the values available in the literature. For the calibration and validation of the meta-heuristics, small fictitious instances were used. After the calibration, the meta-heuristics were applied to the resolution of larger and real instances. As a result, the Iterated Local Search (ILS) was the better performing meta-heuristic, and the result obtained for the real instance was 40.5% better than the one presented in the literature.

KEYWORDS. Long-term maintenance scheduling, GRASP, Simulated Annealing, Iterated Local Search, Meta-heuristics, Heuristics.

1. Introdução

As atividades de manutenção preventiva são essenciais para a disponibilidade e confiabilidade dos sistemas dentro da indústria. Os mais diversos equipamentos, desde esteiras transportadoras a empilhadeiras, possuem um cronograma de manutenções que devem ser realizadas com certa frequência, que geralmente é sugerida pelo fabricante. As atividades de manutenção preventiva podem ser inspeção, limpeza, lubrificação, ajuste, alinhamento ou substituição de componentes desgastados [Ebrahimipour et al., 2015].

A manutenção preventiva é especialmente importante para evitar que ocorram falhas na operação que possam causar danos consideráveis ao sistema, como quebra de máquina, ou ao ambiente, como poluição, explosões, perda de informação [Levitin et al., 2021]. Estas atividades de manutenção preventiva necessitam de uma equipe e ferramentas adequadas para sua execução. Isso faz com que um número limitado de atividades possam ser executadas dentro de determinado período de tempo.

Para que o maior número de atividades, ou para que as mais importantes sejam executadas, é necessário que se realize a programação das ordens de manutenção preventiva. Um problema de sequenciamento de ordens de manutenção em uma única máquina é NP-difícil [Qi et al., 1999]. O presente trabalho irá abordar o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP) em máquinas paralelas não relacionadas. Como este tipo de problema é mais complexo que um problema de máquina única, o problema também é definido como NP-difícil.

Apesar da programação da manutenção preventiva ser um tema amplamente abordado, estudos como de Lee e Cha [2016] e Wang e Miao [2021] trabalham com modelos para previsão das falhas, e não com modelos focados na programação das ordens de manutenção preventiva. Outros trabalhos como Pacheco et al. [2018] e Chen et al. [2015] tratam de modelos para a resolução de problemas de programação das ordens de manutenção, mas trabalham com instâncias pequenas comparadas a instâncias reais. Portanto, este trabalho tem como objetivo obter uma solução viável que gere redução de manutenções não realizadas e de custo com equipes no PPOMPLP por meio de meta-heurísticas em instâncias de grande complexidade.

O local alvo do estudo é uma unidade de beneficiamento de minério de ferro, localizada no Estado de Minas Gerais, na qual será realizada a programação das ordens do período de um ano (52 semanas). Para a resolução do problema em questão, propõe-se inicialmente um método heurístico construtivo para realizar o sequenciamento das ordens de manutenção. Com a sequência já definida, uma segunda heurística é utilizada para realizar a alocação das ordens às respectivas equipes ao longo do período de programação. Estas primeiras heurísticas já terão seus resultados comparados aos apresentados na literatura [Aquino, 2018], que trabalharam com este mesmo problema, com diferentes abordagens de resolução. Como métodos meta-heurísticos serão utilizados *Greedy Randomized Adaptive Search Procedure* (GRASP), *Simulated Annealing* (SA) e *Iterated Local Search* (ILS). Serão analisadas instâncias fictícias no mesmo contexto da situação problema e uma instância real.

A empresa abordada pelo trabalho é uma multinacional. O problema real do tipo PPOMPLP vem de uma das plantas de beneficiamento de minério de ferro, que realiza o planejamento de todas as ordens de manutenção para o período de um ano. Este plano é chamado de mapa de 52 semanas, no qual cada ordem de manutenção é relativa a uma atividade de manutenção específica, que deve ser executada por uma equipe de trabalho, utilizando uma ferramenta.

Quando a empresa realiza o plano de manutenção, cada área da manutenção (ex.: mecânica, elétrica) realiza seu próprio plano, baseado apenas na disponibilidade do equipamento necessário para a manutenção. Este plano é inserido no sistema ERP (do inglês *Enterprise Resource*

Planning, um sistema de gestão empresarial) utilizado pela empresa; porém, esse sistema não contém informação das restrições de capacidade presentes no sistema real, como por exemplo a disponibilidade de mão de obra. Então, a programação das manutenções é ajustada mensalmente, por cada equipe, conforme a disponibilidade das respectivas mãos de obra e equipamentos necessários. Este método de planejamento de longo prazo faz com que apenas cerca de 50% das ordens de manutenção sejam atendidas pela mão de obra interna. O restante das ordens de manutenção são terceirizadas, o que resulta em mais custo para a empresa. Neste contexto, há uma oportunidade de melhoria utilizando outro método para a realização deste planejamento de longo prazo.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta o referencial teórico, contextualizando o problema de programação de manutenções, heurísticas e meta-heurísticas. A Seção 3 apresenta a metodologia seguida no estudo. As Seções 4 e 5 apresentam as heurísticas construtivas e a heurística de alocação. Na Seção 6 apresentam-se as meta-heurísticas utilizadas e na Seção 7 os resultados obtidos. Por fim, na Seção 8, são apresentadas as conclusões e as sugestões para trabalhos futuros.

2. Referencial teórico

A manutenção preventiva traz para os equipamentos vantagens como: a redução da frequência de falhas, aumento da vida útil e aumento da qualidade dos produtos produzidos. Porém, como ponto negativo, a manutenção preventiva necessita da interrupção da atividade produtiva, ou da utilização do equipamento alvo da manutenção [Swanson, 2001]. Apesar da atividade de manutenção preventiva trazer este prejuízo para a programação da produção, os benefícios são de maior grandeza [Pacheco et al., 2018] e, portanto, é importante que a empresa tenha um método eficaz de programação da manutenção preventiva.

Esta terminologia de Pinedo [2016], foi utilizada por Aquino [2018] para representar uma simplificação do PPOMPLP como um problema $P_m \mid r_j M_j \mid \gamma$. Um ponto de discordância é que o campo γ , Aquino [2018] considera que o objetivo de otimização seria minimizar a mão de obra necessária para executar o maior número de tarefas, e não havendo notação correspondente, o campo permanece γ . Porém, o problema em questão possui dois objetivos de otimização: minimizar o custo relacionado às ordens de manutenção não atendidas e minimizar o custo de mão de obra. O custo de mão de obra tem ordem de grandeza muito menor comparado ao custo de cada manutenção não realizada, portanto o primeiro pode ser considerado o principal objetivo de otimização.

Seguindo esta mesma terminologia de Pinedo [2016], o presente trabalho aborda um problema $P_m \mid r_j M_j \mid \sum_j w_j T_j$. O campo P_m se refere à problemas de máquinas em paralelo. r_j significa que a atividade j não pode começar seu processamento antes da data de lançamento. M_j é utilizado para problemas de máquinas em paralelo, nos quais nem todas as m máquinas são capazes de processar todas as atividades. E no último campo, $\sum_j w_j T_j$ se refere à média ponderada dos tempos de atraso. Apesar das manutenções não poderem ser alocadas com atraso, estas entram como custo para a função objetivo, de forma ponderada, de acordo com sua prioridade.

Para este tipo de problema podem ser utilizadas heurísticas. As heurísticas clássicas podem encontrar soluções viáveis rapidamente; porém, há disparidade em relação aos valores ótimos. Para encontrar soluções mais próximas da ótima, ou até mesmo a solução ótima global, são aplicadas as meta-heurísticas GRASP, SA, ILS.

O conceito de SA foi aplicado pela primeira vez para resolução de problemas de otimização por Kirkpatrick et al. [1983]. O método se baseia no processo de aquecimento e resfriamento de um sólido. Ao aquecer, o algoritmo produz soluções mais diversas para o problema, e ao resfriar, o algoritmo converge para um ótimo local.

O ILS é amplamente aplicado à resolução de problemas de programação de produção, que é um problema análogo ao problema de programação de ordens de manutenção preventiva [Xu et al., 2019].

Outra técnica de rápida implementação e bons resultados é o GRASP. O GRASP é uma meta-heurística *multi-start* que consiste de duas fases: a fase construtiva, na qual uma solução viável é produzida, e a segunda fase, que é um método de busca local, que irá encontrar uma solução ótima local [Feo e Resende, 1995].

3. Metodologia

Esta pesquisa se inicia pelo desenvolvimento das heurísticas construtivas, que irão gerar uma sequência de programação inicial. Posteriormente, uma heurística de alocação é utilizada para estruturar as atividades de cada equipe ao longo do período de programação. Serão então desenvolvidos os métodos de busca local *first improvement* e *random descent* para posterior utilização nas meta-heurísticas. Em uma etapa seguinte, é realizada a implementação das meta-heurísticas GRASP, SA e ILS para resolução do PPOMPLP.

A parametrização das meta-heurísticas utiliza o pacote IRACE [López-Ibáñez et al., 2016], realizando rodadas teste com uma amostra de instâncias fictícias. Para a validação das meta-heurísticas são resolvidas instâncias fictícias pequenas. Após a calibragem e validação, as meta-heurísticas são aplicadas na resolução de instâncias maiores e a real. Os resultados obtidos são comparados entre as meta-heurísticas aplicadas e com os resultados apresentados na literatura.

4. Heurísticas construtivas

Para descrição matemática do modelo foram utilizadas as seguintes notações. $\mathcal{T} = \{1, 2, \dots, n\}$ é o conjunto de n atividades de manutenção que devem ser realizadas pelo conjunto $\mathcal{W} = \{1, 2, \dots, m\}$ de m equipes de trabalho. Cada manutenção $i \in \mathcal{T}$ é associada ao conjunto $\mathcal{W}'_i \subseteq \mathcal{W}$ de equipes de trabalho capazes de realizá-la. Também associados a cada manutenção $i \in \mathcal{T}$ estão o tempo p_i necessário para executá-la, A_i o equipamento ou ferramenta que será utilizado, e janela de tempo $[e_i, l_i]$ que a manutenção pode ser realizada. A penalidade por não realizar a manutenção é w_i . O valor da penalidade foi definido como o tempo p_i multiplicado por um valor de prioridade da ordem de manutenção, definido pela empresa.

Cada equipe de trabalho $k \in \mathcal{W}$ está disponível no período de $[0, h_k]$. O conjunto $\mathcal{T}'_k \subseteq \mathcal{T}$ indica as manutenções que podem ser realizadas pela equipe de trabalho $k \in \mathcal{W}$. A função objetivo (equação 1) busca minimizar a soma do custo de mão de obra com as manutenções não atendidas. Sendo $z_k = 1$, se a equipe de trabalho k é utilizada; 0, caso contrário. E $y_{ik} = 1$, se a manutenção i será executada pela equipe de trabalho k ; 0, caso contrário;

$$\min \sum_{k \in \mathcal{W}} z_k + \sum_{i \in \mathcal{T}} w_i (1 - \sum_{k \in \mathcal{W}'_i} y_{ik}) \quad (1)$$

Para o problema, foram propostas duas heurísticas construtivas diferentes: uma heurística chamada de “construção do sequenciamento simples das ordens” e a outra “construção do sequenciamento das ordens por ferramenta”. A primeira (Algoritmo 1), realiza o sequenciamento das ordens tendo o tempo limite da ordem (l_c) como o principal parâmetro e, em caso de empate entre 2 ordens, a penalidade (w_c) é o critério de desempate. Ordens com menor valor de l_c e maior valor de w_c têm prioridade de alocação. O algoritmo retorna a sequência de ordens N .

O segundo algoritmo construtivo realiza primeiro uma avaliação de qual é a ferramenta mais utilizada. O sequenciamento das ordens realizadas pela ferramenta mais utilizada é definido primeiro, seguindo os mesmos critérios do Algoritmo 1. Ordens com menor valor de l_c e maior valor

Algoritmo 1: Construção do sequenciamento simples das ordens

Entrada: Lista de ordens de manutenção
Saída: N

```

1 início
2    $N \leftarrow \emptyset$ ;
3    $C \leftarrow$  Inicializa o conjunto de ordens candidatas;
4   enquanto  $C \neq \emptyset$  faça
5      $melhor\_tempo \leftarrow \infty$ ;
6      $melhor\_penalidade \leftarrow -1$ ;
7     para  $c \in C$  faça
8       se  $(l_c < melhor\_tempo)$  ou  $(l_c = melhor\_tempo$  e
9          $w_c > melhor\_penalidade)$  então
10         $c^* \leftarrow c$ ;
11         $melhor\_tempo \leftarrow l_c$ ;
12         $melhor\_penalidade \leftarrow w_c$ ;
13      fim
14     $N \leftarrow N \cup \{c^*\}$ ;
15     $C \leftarrow C \setminus \{c^*\}$ ;
16  fim
17 fim
  
```

de w_c têm prioridade de alocação. Após sequenciar todas as ordens da ferramenta mais utilizada, é realizado o sequenciamento da segunda mais utilizada, até que o sequenciamento de todas seja concluído.

Para exemplificar, uma instância fictícia foi apresentada no Quadro 1. Para a instância em questão, a sequência resultado do Algoritmo 1 é $N = \{1,4,6,2,5,3\}$. Para o segundo algoritmo construtivo, como a ferramenta carro é a mais utilizada, a sequência se iniciaria por ela. Isto resultaria na seguinte sequência: $N = \{1,5,3,6,2,4\}$. As meta-heurísticas desenvolvidas utilizam o algoritmo construtivo com melhor resultado como solução inicial do problema porque há diferença de acordo com a instância.

Quadro 1: Ordens de manutenção.

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

Fonte: adaptado de [Aquino, 2018]

5. Heurística de alocação

Com a sequência das ordens definida, é necessário realizar a alocação destas. A alocação é realizada por equipes, iniciando pela equipe, ou grupo de equipes, mais ocupada. A Figura 1 complementa o exemplo apresentado no Quadro 1, mostrando em (a) quantas equipes estão disponíveis

para execução das ordens e quais suas especialidades. Em (b) é possível ver como é feito o cálculo de qual é a equipe mais ocupada. Primeiro soma-se a duração de todas as atividades que podem ser realizadas por grupo de equipes (neste caso o grupo é mecânica ou elétrica). Esta duração total é, então, dividida pelo número de equipes em um grupo. O grupo com maior ocupação tem prioridade na alocação das ordens.

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

(a)

	Duração total	Número de equipes	Ocupação
Mecânica	7	2	3,5
Elétrica	3	1	3

(b)

Figura 1: (a) Especialidade e ocupação das equipes. (b) Ocupação das equipes

Para a alocação das ordens foi implementado o Algoritmo 2. A alocação das ordens é realizada do começo da janela de tempo da ordem c (e_c). Caso não seja possível alocar a ordem na primeira posição, o horário de início da ordem é postergado até ser possível. Caso ultrapasse a janela de tempo da ordem, a ordem será alocada em outra equipe. Caso não seja possível alocar em nenhuma das outras equipes, a ordem não será alocada e entra como penalidade no resultado da função objetivo. Há dois motivos para que a ordem não possa ser alocada em sua primeira posição: caso já exista uma ordem alocada na mesma equipe no mesmo horário ou caso tenha uma ordem alocada usando a mesma ferramenta em outra equipe no mesmo horário. As variáveis *inicio* e *fim* utilizadas no Algoritmo 2 são os horários de início e fim de execução de uma ordem na programação. O algoritmo retorna a solução s , a penalidade e a lista de ordens não utilizadas.

A Figura 2 exemplifica o funcionamento da alocação de ordens. O primeiro gráfico de Gantt representa a alocação das ordens do Quadro 1 seguindo o método “construção do sequenciamento simples das ordens” (Algoritmo 1). A sequência $N = \{1,4,6,2,5,3\}$, deve começar pelas equipes de mecânica. Isto pode ser traduzido em uma sequência menor: $N = \{1,6,2,3\}$. As ordens são alocadas na equipe Mecânica A respeitando-se as respectivas janelas de tempo. Após a alocação de todas ordens das equipes de mecânica, a alocação é realizada para as equipes de elétrica, seguindo o restante da sequência $N = \{4,5\}$. O mesmo procedimento de alocação é realizado para a “construção do sequenciamento das ordens por ferramenta”(segundo Gantt), sendo apenas a sequência diferente.

6. Meta-heurísticas

Os parâmetros utilizados nas meta-heurísticas foram obtidos por meio de calibração utilizando o pacote IRACE. O único parâmetro fixo e comum a todas as meta-heurísticas é o tempo limite de execução, que é de n segundos, sendo n o número de ordens de manutenção da instância. Cada algoritmo foi executado com cinco repetições para cada instância. O algoritmo GRASP foi desenvolvido com base no de Feo e Resende [1995], utilizando como método de busca local o *random descent*. O *random descent* desenvolvido realiza trocas aleatórias entre duas ordens (i, j) da sequência dada N . Só é avaliada a troca das ordens i e j se elas estiverem dentro de um mesmo período de programação ($e_j > l_i > l_j$ ou $e_i > l_j > l_i$). Caso a troca produza uma solução melhor, ela é aceita, modificando-se N .

Para cada nova sequência gerada dentro da meta-heurística é necessário refazer a alocação das ordens, para que seja calculado o novo custo da função objetivo. A heurística de alocação, por

Algoritmo 2: Aloca ordens

Entrada: Lista de ordens de manutenção, lista de equipes, sequência(N)
Saída: s, penalidade, ordens não utilizadas;

```

1  início
2       $s \leftarrow \emptyset$ ;
3      Organiza equipes, começando pelas equipes mais ocupadas;
4      para  $k \in W$  faça
5           $C \leftarrow$  Inicializa conjunto de ordens candidatas, seguindo sequência N, apenas
              com ordens que a equipe é capaz de realizar e com ordens  $\notin s$ ;
6          enquanto  $C \neq \emptyset$  e  $fim_c \leq h_k$  faça
7               $c \leftarrow C[0]$ ;
8               $inicio_c \leftarrow e_c$ ;
9               $fim_c \leftarrow inicio_c + p_c$ ;
10             enquanto Houver conflito de horários entre  $c$  e  $s[i]$  e  $fim_c \leq l_c$  e
                   $fim_c \leq h_k$  faça
11                 para  $i \in s$  faça
12                     se Ordem  $s[i]$  alocada na mesma equipe e mesmo horário então
13                          $inicio_c \leftarrow s[i][fim_c]$ ;
14                          $fim_c \leftarrow inicio_c + p_c$ ;
15                     fim
16                     se Ordem  $s[i]$  com a mesma ferramenta alocada em outra equipe
                          e mesmo horário então
17                          $inicio_c \leftarrow s[i][fim_c]$ ;
18                          $fim_c \leftarrow inicio_c + p_c$ ;
19                     fim
20                 fim
21             fim
22             se  $fim_c > l_c$  ou  $fim_c > h_k$  então
23                  $C \leftarrow C \setminus c$ ;
24             senão
25                 Insere  $c$  na solução  $s$ , com horário de início e término definidos por
                      $inicio_c$  e  $fim_c$  respectivamente;
26                  $C \leftarrow C \setminus c$ ;
27             fim
28         fim
29     fim
30     Calcula penalidade e ordens não utilizadas;
31 fim

```

ser a mais custosa em termos de processamento, tornaria inviável que todo o período de programação fosse realocado a cada troca. Por isso, a cada nova sequência gerada dentro dos métodos meta-heurísticos, é refeita apenas a alocação dentro de períodos em que hajam ordens não alocadas, mantendo a alocação anterior para os demais períodos.

O algoritmo SA foi desenvolvido com base no trabalho de Kirkpatrick et al. [1983]. O

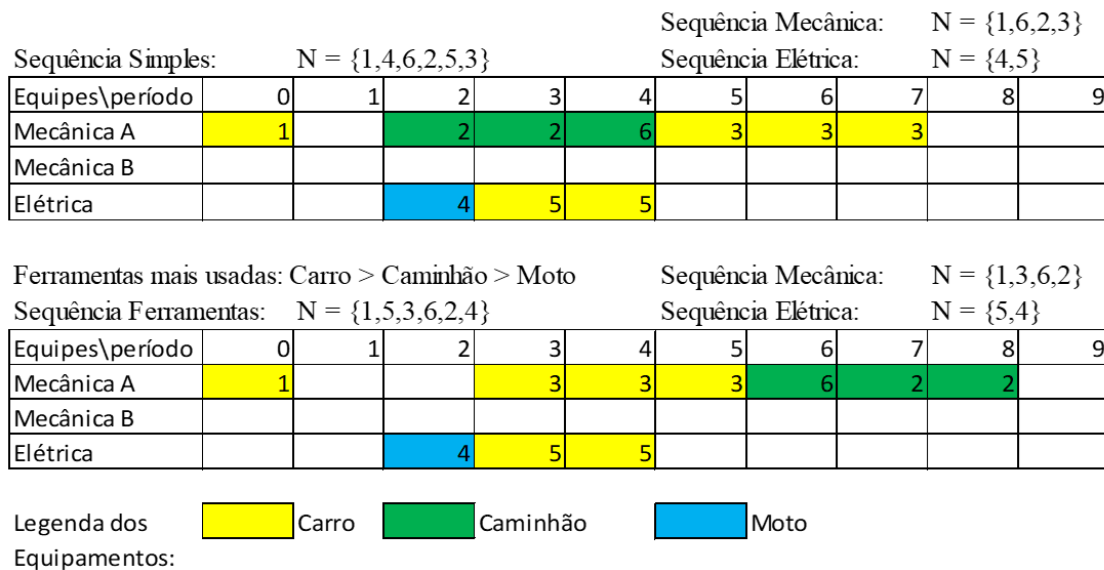


Figura 2: Exemplo de alocação de ordens

vizinho aleatório é gerado trocando duas ordens aleatórias (i, j) da sequência de programação. Para limitar o número de trocas, as ordens i e j só são trocadas se estiverem dentro do mesmo período de programação, semelhante à lógica utilizada no *random descent*. Ordens em períodos de tempos diferentes têm pouca influência na alocação da outra.

O algoritmo ILS foi adaptado do proposto por Lourenço et al. [2019]. Como métodos de busca local foram utilizados o *random descent*, já explicado anteriormente, e o *first improvement*, sendo sorteados, com igual probabilidade, qual será utilizado em cada iteração. O método *first improvement* realiza a troca das ordens i e j , começando por i sendo o primeiro elemento de N e j o segundo elemento. A cada iteração é acrescida uma unidade à i ou j até que todas as trocas possíveis sejam avaliadas. Ao longo das iterações, se uma troca representa melhora no resultado da função objetivo, a troca é realizada, atualizando N , e reiniciando o *first improvement*. As trocas do ILS também só são avaliadas se estiverem dentro de um mesmo período de programação.

7. Resultados

Ao iniciar os testes com as meta-heurísticas, foi observado que mesmo rodando os algoritmos por tempos longos, alguns resultados da literatura não eram alcançáveis. Após uma análise detalhada dos resultados obtidos por Aquino [2018], foi possível apontar algumas incoerências. Em algumas instâncias, existem ordens não alocáveis, que podem ser basicamente de três tipos:

- O tempo mínimo de início (e_c) da ordem c é maior que a disponibilidade (h_k) das k equipes capazes de realizar a ordem;
- O tempo de processamento (p_c) é maior que a janela de tempo $[e_c, l_c]$ da ordem c ;
- Dado um grupo de ordens G_c , que possua a mesma janela de tempo $[e_c, l_c]$ e que utilize uma mesma ferramenta A_c , a soma dos tempos de processamento (p_c) das ordens G_c , ultrapassa a janela de tempo $[e_c, l_c]$.

Em instâncias que possuem essas ordens não alocáveis não é possível obter valores da função objetivo menores que a soma das penalidades destas. No caso do terceiro tipo, onde há

um grupo de ordens que ultrapassa a janela de tempo, as ordens com menor penalidade devem ser as não alocadas. Portanto, foram identificadas as instâncias em que Aquino [2018] apresenta tais incoerências e estas foram removidas com a finalidade de comparar o resultado das demais com o resultado do presente trabalho.

Para análise dos resultados, as instâncias fictícias foram divididas em dois grupos: grupo 1, com as instâncias pequenas e grupo 2 com as instâncias maiores. A comparação entre as meta-heurísticas está apresentada na Tabela 1. A coluna “Média” mostra o valor da média dos valores obtidos para cada instância. A coluna “Melhor” apresenta o menor valor obtido dentre todas instâncias. Nas colunas “contagem melhores” indica-se em quantas instâncias cada meta-heurística obteve o melhor resultado. O ILS obteve resultados melhores em 98,9% das instâncias do grupo 1, sendo o GRASP melhor em apenas uma instância e SA atingindo resultados no máximo iguais aos demais. Entre SA e GRASP, os resultados foram muito próximos na média, com SA tendo melhor desempenho nas instâncias do grupo 2. Nas instâncias do grupo 2, o ILS foi a meta-heurística com melhor desempenho, obtendo o melhor resultado em 68,7% das instâncias. O SA obteve o melhor resultado em 53,1% e o GRASP em 40,6%. Esses números somados são maiores que 100% porque pode haver empate entre as meta-heurísticas em uma ou mais instâncias.

Tabela 1: Comparativo entre meta-heurísticas

	Número de ordens	Número de instâncias	GRASP			SA			ILS		
			Média	Melhor	Contagem melhores	Média	Melhor	Contagem melhores	Média	Melhor	Contagem melhores
GRUPO 1	20	20	299	219	20	299	219	20	299	219	20
	30	20	299	219	20	299	219	18	299	219	20
	40	20	332	219	19	339	219	14	330	219	20
	60	20	504	219	18	502	219	12	482	219	19
	80	10	637	219	5	601	219	6	591	219	10
GRUPO 2	150	5	697	24	5	696	24	5	696	24	5
	300	5	1.247	35	2	1.247	36	3	1.246	35	4
	600	5	3.729	40	1	3.704	47	2	3.656	41	4
	1.200	5	8.897	300	1	8.625	300	3	8.331	303	3
	2.400	5	16.401	567	1	15.705	567	2	15.692	567	4
	4.800	5	34.648	1.422	3	34.769	1.426	1	34.645	1.426	1
	9.600	1	31.664	30.451	0	30.199	30.009	0	30.011	30.002	1
	19.200	1	74.257	70.963	0	69.200	68.711	1	69.672	68.720	0
REAL	33.484	1	134.610	128.856	1	133.291	129.082	0	134.507	130.906	0

O ILS obtém resultados piores em algumas das instâncias maiores por utilizar como método de busca local o *first improvement*, que tem grande custo computacional, e, assim, não se encerra dentro do tempo de processamento estipulado.

Ao identificar o ILS como o melhor algoritmo de forma geral, este será comparado com a literatura. No trabalho de Aquino [2018] algoritmos diferentes também obtiveram diferentes performances dependendo do tamanho da instância. Portanto, para fins de comparação, foi escolhido o *Biased Random-Key Memetic Algorithm* (BRKMA), porque é o algoritmo que obteve melhores resultados nas instâncias maiores e na real, sendo a real o principal objetivo do trabalho.

A Tabela 2 apresenta os resultados da comparação com a literatura, agrupando as instâncias pelo número de ordens. A coluna “Média ILS” apresenta a média dos melhores valores obtidos pelo ILS em cada instância. A coluna “Média Aquino” mostra a média dos melhores valores obtidos pelo BRKMA de Aquino em cada instância.

Para o grupo 1, em 95,6% das instâncias o ILS foi capaz de alcançar resultado igual aos do BRKMA de Aquino [2018], sendo grande parte destes o resultado ótimo. Porém, o ILS obtém resultados piores em 4,4%. Por outro lado, o gap entre o resultado obtido neste trabalho e de Aquino

Tabela 2: Comparativo com a literatura entre grupos de instâncias

	Número de ordens	Número de instâncias	Média ILS	Média Aquino	Gap	Média Gaps
GRUPO 1	20	20	299	299	0,0%	1,4%
	30	20	299	299	0,0%	
	40	20	324	313	3,4%	
	60	20	475	465	2,3%	
	80	10	587	579	1,2%	
GRUPO 2	150	5	696	696	0,0%	-9,8%
	300	5	1.246	1.288	-3,3%	
	600	5	3.631	3.619	0,3%	
	1.200	5	8.191	7.961	2,9%	
	2.400	5	15.191	15.089	0,7%	
	4.800	5	33.925	71.456	-52,5%	
	9.600	1	30.002	38.004	-21,1%	
	19.200	1	68.720	103.863	-33,8%	
REAL	33.484	1	130.906	220.048	-40,5%	

[2018] variou entre 0% e 3,4%, sendo 1,4% piores na média.

Dentre as instâncias do grupo 2, foi possível obter resultados melhores em 56,2%, iguais em 18,8% e piores em 25%. Conforme a Tabela 2 evidencia, ao fazer o comparativo de forma agrupada, o gap varia entre -52,5% (percentual negativo indica que é melhor que da literatura) e 2,9%. Os resultados obtidos pelo presente trabalho foram 9,8% melhores na média das diferenças do grupo 2.

Destaca-se que os melhores resultados foram em grande parte obtidos nas maiores instâncias do grupo 2. Para a instância real, que é o principal objetivo do trabalho, a heurística construtiva já produzia um resultado 37% melhor e, após a aplicação do ILS, a solução final houve uma melhora de 40,5% em relação ao trabalho de Aquino [2018].

8. Conclusões e trabalhos futuros

Neste trabalho foram implementadas heurísticas construtivas e meta-heurísticas para gerar soluções viáveis em um PPOMPLP, visando à redução de manutenções não realizadas e de custo com equipes. Os resultados obtidos foram comparados com a literatura disponível e entre si. Para as instâncias pequenas, os resultados são na sua maioria iguais aos presentes na literatura. Para as instâncias maiores, houve ganho de 9,8% na média e ganho de 40,5% para a instância real. Dentre os algoritmos meta-heurísticos, o ILS foi o que apresentou melhores resultados para a grande maioria das instâncias, sendo GRASP e SA melhores apenas para algumas instâncias maiores do grupo 2.

Uma sugestão para estudos futuros é a realização do processamento das meta-heurísticas por mais tempo de processamento, para verificar se resultados melhores são alcançados, principalmente com a completude do método *first improvement* para a instância real. Outra sugestão seria trabalhar com a programação das ordens de manutenção não atendidas, inserindo-as na programação com atraso, ao invés de simplesmente não alocá-las. Isto gera um novo problema com objetivo de otimização diferente.

Agradecimentos

Os autores gostariam de agradecer à CAPES, ao CNPq, à FAPEMIG e à UFOP pelo apoio ao desenvolvimento deste projeto.

Referências

- Aquino, R. D. (2018). *Abordagem Exata e Heurísticas para o Problema de Planejamento de Ordens de Manutenção de Longo Prazo: Um Estudo de Caso Industrial de Larga Escala*. Dissertação de mestrado, Programa de pós-graduação em Ciência da Computação, Universidade Federal de Ouro Preto, Ouro Preto.
- Chen, X., XiAo, L., ZhAng, X., XiAo, W., e Li, J. (2015). An integrated model of production scheduling and maintenance planning under imperfect preventive maintenance. *Eksplotacja i Niezawodność*, 17(1):70–79.
- Ebrahimipour, V., Najjarbashi, A., e Sheikhalishahi, M. (2015). Multi-objective modeling for preventive maintenance scheduling in a multiple production line. *Journal of Intelligent Manufacturing*, 26(1):111–122.
- Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.
- Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.
- Lee, H. e Cha, J. H. (2016). New stochastic models for preventive maintenance and maintenance optimization. *European Journal of Operational Research*, 255(1):80–90.
- Levitin, G., Xing, L., e Dai, Y. (2021). Optimal operation and maintenance scheduling in m-out-of-n standby systems with reusable elements. *Reliability Engineering & System Safety*, 211:107582.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.
- Lourenço, H. R., Martin, O. C., e Stützle, T. (2019). Iterated local search: Framework and applications. In Gendreau, M. e Potvin, J.-Y., editors, *Handbook of metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, p. 129–168. Springer.
- Pacheco, J., Porras, S., Casado, S., e Baroque, B. (2018). Variable neighborhood search with memory for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. *Knowledge-Based Systems*, 145:236–249.
- Pinedo, M. L. (2016). *Scheduling: theory, algorithms, and systems*. Springer, Cham, 5 edition.
- Qi, X., Chen, T., e Tu, F. (1999). Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, 50(10):1071–1078.
- Swanson, L. (2001). Linking maintenance strategies to performance. *International Journal of Production Economics*, 70(3):237–244.
- Wang, J. e Miao, Y. (2021). Optimal preventive maintenance policy of the balanced system under the semi-markov model. *Reliability Engineering & System Safety*, 213:107690.
- Xu, J., Liu, S.-C., Zhao, C., Wu, J., Lin, W.-C., e Yu, P.-W. (2019). An iterated local search and tabu search for two-parallel machine scheduling problem to minimize the maximum total completion time. *Journal of Information and Optimization Sciences*, 40(3):751–766.