Um algoritmo GRASP-ILS para um problema de planejamento de ordens de manutenções de longo prazo

João Luiz Marques de Andrade[1], Marcone Jamilson Freitas Souza[1,2], Sérgio Ricardo Souza[1], Gustavo Campos Menezes[1], Elisângela Martins de Sá[1]
[1]Programa de Pós Graduação em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG
Campus Nova Gameleira - CEP 30421-169 - Belo Horizonte - MG, Brasil
[2]Universidade Federal de Ouro Preto - UFOP
Campus Universitário Morro do Cruzeiro - CEP 35400-000 - Ouro Preto - MG, Brasil
andrad.joao15@gmail.com, marcone@iceb.ufop.br, sergio@cefetmg.br, gustavo@cefetmg.br, elisangelamartins@cefetmg.br

RESUMO

Este artigo estuda o problema de planejamento de ordens de manutenção de longo prazo. O problema busca definir quais ordens de manutenção são executadas, alocar as ordens de manutenção preventiva nas equipes de trabalho e definir quando as ordens devem ser executadas pelas equipes, buscando minimizar o número de equipes de manutenção utilizadas e a penalização de ordens de manutenção não executadas. Para solucionar este problema, pertencente à classe NP-difícil, com instâncias baseadas em casos reais, foi desenvolvido um algoritmo que combina as metaheurísticas GRASP e ILS. Os resultados computacionais mostram a superioridade do algoritmo proposto em relação às abordagens de solução da literatura, encontrando soluções melhores em um tempo expressivamente menor.

PALAVRAS CHAVE. Planejamento de manutenção de longo prazo, Escalonamento, Metaheurística.

Tópicos (Metaheurística, Otimização Combinatória)

ABSTRACT

This paper addresses the long-term maintenance programming problem. The problem seeks to define which maintenances are executed, allocate the preventive maintenances to the work teams, and determine when the maintenances should be performed by the teams, minimizing the number of maintenance teams used and the penalty of maintenances not performed. An algorithm that combines GRASP and ILS metaheuristics was developed to solve this problem belonging to the NP-hard class with instances based on real cases. The computational results show the superiority of the proposed algorithm over the solution approaches in the literature, finding better solutions in an expressively shorter time.

KEYWORDS. Long-term maintenance programming. Scheduling. Metaheuristics.

Paper topics (Metaheuristics, Combinatorial Optimization)

1. Introdução

A manutenção é uma atividade operacional que pretende garantir consistência operacional, eficiência e produtividade de um sistema, como, por exemplo, em indústrias e em instalações de transportes, dentre outras. Em geral, a manutenção é definida como o trabalho realizado para manter um sistema em bom estado e, principalmente, em funcionamento. O principal objetivo da manutenção é evitar ou mitigar as consequências de falhas e proporcionar um funcionamento confiável de um sistema.

De acordo com Wang [2002], a manutenção pode ser classificada em dois tipos básicos: *(i)* manutenção corretiva e *(ii)* manutenção preventiva. A manutenção corretiva é a ação executada como resultado de uma falha, a fim de restaurar um sistema em uma condição especificada. Por outro lado, a manutenção preventiva é a ação executada com finalidade de evitar a ocorrência de falhas. As boas políticas de manutenção preventiva podem reduzir a probabilidade de falhas e, consequentemente, aumentar a produtividade de um sistema.

Os benefícios importantes da manutenção e sua associação com estudos da área da pesquisa operacional formam um campo fértil para novas pesquisas. Pertencendo aos problemas de manutenção preventiva, este artigo estuda o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP). Este problema, que foi introduzido por Aquino et al. [2019], pretende definir quais ordens de manutenção são executadas, alocar as ordens de manutenção preventiva nas equipes de trabalho e definir quando as ordens devem ser executadas pelas equipes, minimizando o número de equipes de manutenção utilizadas e a soma das penalidades das ordens de manutenção não executadas. Cada manutenção dever ser executada em um equipamento por uma equipe especializada na respectiva atividade com uma janela de tempo e duração previamente definida.

Em Aquino et al. [2019], o PPOMPLP é reduzido ao problema de sequenciamento em máquinas paralelas não-relacionadas, quando são consideradas as ordens de manutenção como tarefas e as equipes de trabalho como máquinas paralelas, desse modo pertencendo à classe NP-difícil. Além disso, o PPOMPLP considera instâncias de larga escala baseadas em casos reais. Essas considerações motivam o desenvolvimento de novos e eficientes métodos de solução. Baseado nisso, o presente trabalho propõe um algoritmo que combina as metaheurísticas *Iterated Local Search* (ILS) e *Greedy Randomized Adaptive Search Procedure* (GRASP). Os resultados mostram que o algoritmo proposto encontra soluções melhores com tempo de processamento significativamente menor em relação ao trabalho de Aquino et al. [2019].

O restante do artigo está estruturado da seguinte forma: a Seção 2 fornece uma revisão de literatura. A Seção 3 define o problema e apresenta um exemplo simples. A Seção 4 apresenta a abordagem de solução desenvolvida. Os resultados e experimentos computacionais são analisados na Seção 5. Por fim, conclusões e propostas de pesquisas futuras são abordadas na Seção 6.

2. Revisão Literária

Os problemas de otimização associados à manutenção recebem uma considerável atenção dos pesquisadores de pesquisa operacional, formando uma literatura que se estende por várias linhas de pesquisas. Nesta seção, são revisados trabalhos recentes que apresentam relativa associação com o problema estudado por este artigo.

Uma parte desta literatura foca em problemas que consideram simultaneamente as manutenções corretivas e preventivas. Em Yu e Strömberg [2021] os autores combinam um modelo matemático de programação de manutenção preventiva com estratégias de manutenção corretiva em sistemas eólicos. Nesse trabalho, as manutenções preventivas devem ser alocadas em uma janela de tempo, de maneira semelhante ao considerado no presente artigo. Os resultados encontrados

mostram vantagens significativas quando é realizada uma estratégia com manutenções corretivas e preventivas, em vez de apenas manutenções corretivas. Os trabalhos de Qamhan et al. [2020] e Shen e Zhu [2018] pertencem à classe de problemas de sequenciamento de uma máquina com manutenção periódica. Em Qamhan et al. [2020], o problema tratado considera a integração do planejamento da produção e janelas de tempos para execução de manutenções preventivas periódicas e busca minimizar os atrasos. Para as instâncias de menor escala, os autores desenvolvem um modelo matemático, e, para resolver instâncias reais, desenvolvem uma metaheurística baseada em colônia de formigas.

Os problemas de sequenciamento de manutenção possuem aplicações em diversas áreas. A indústria de eletricidade é uma delas, como mostra a revisão literária de Froger et al. [2016]. Woller e Kulich [2021] estuda um problema de sequenciamento de manutenção em linhas de transmissão. Neste problema, as manutenções não podem ser executadas simultaneamente com outras devido à proximidade física das linhas de transmissão. Assim como no presente trabalho, manutenções no mesmo equipamento não podem ser executadas ao mesmo tempo. Para solucionar o problema, os autores utilizaram a metaheurística *Adaptive Large Neighborhood Search*. Zhang et al. [2019], a partir de um caso real na China, estudam um problema que integra a programação das operações de trens e o planejamento das manutenções nas ferrovias de alta velocidade. O objetivo consiste em minimizar o tempo total de viagem dos trens. As manutenções devem ser alocadas em uma janela de tempo no período da noite e, simultaneamente, devem ser definidas quais linhas devem ser desativadas durante o processo de manutenção. Além disso, as manutenções requerem equipes com habilidades específicas para serem executadas. Os autores usam técnicas de linearização para formular um modelo de programação linear inteira mista e solucionar o problema. Sedghi et al. [2021] apresentam uma revisão da literatura de problemas de planejamento e sequenciamento de manutenções em vias ferroviárias. Shaukat et al. [2020] estudam um problema em que manutenções preventivas devem ser planejadas dentro de janelas de tempo determinadas pela programação de parada de aeronaves. Cada manutenção tem um prazo em que deveria ser executada de modo ideal; portanto, o objetivo do problema é minimizar o desvio desta programação, sem exceder a capacidade de recursos disponíveis (equipes de trabalho) nos aeroportos. No artigo citado, é proposto um modelo de programação inteira mista e uma heurística matemática.

Em um problema diferente dos revisados e novo na literatura, porém ainda associado ao planejamento de manutenção, Aquino et al. [2019] fazem um estudo de caso do planejamento de manutenções preventivas de uma unidade de beneficiamento de minério de ferro no Brasil. O objetivo é otimizar o planejamento de um conjunto de manutenções preventivas que devem ser executadas em um horizonte de 52 semanas. Os autores buscavam aumentar o número de manutenções preventivas efetivamente executadas e minimizar o número de equipes de manutenção utilizadas e, como consequência, reduzir as manutenções corretivas. O artigo citado apresenta um modelo de programação inteira linear, porém, para solucionar o problema em instâncias de larga escala, foram implementados algoritmos metaheurístico baseados nos métodos *Variable Neighborhood Search*, *Biased Random-Key Genetic Algorithm* e *Biased Random-Key Memetic Algorithm*. As soluções geradas pelos algoritmos desenvolvidos foram capazes de produzir um planejamento melhor do que os empregados pela empresa.

Este artigo estuda o problema proposto por Aquino et al. [2019]. Como contribuição, este trabalho apresenta um novo e mais eficiente método de solução e, como consequência, novos limites superiores para o problema. O método de solução proposto neste artigo é uma abordagem que combina os algoritmos *Iterated Local Search* (ILS) e *Greedy Randomized Adaptive Search Procedure* (GRASP), e utiliza um novo algoritmo de posicionamento de ordens de manutenção.

3. Definição do problema

O PPOMPLP pode ser formalmente definido como a seguir. Seja um conjunto $O = \{1, \ldots, o\}$ de ordens de manutenção que devem ser executadas e um conjunto $T = \{1, \ldots, t\}$ de equipes de trabalho disponíveis para executar as ordens. Cada ordem de manutenção $i \in O$ pode ser realizada por um conjunto $\mathcal{T}_i \subseteq T$ de equipes que possuem a habilidade necessária para realizar a manutenção. Além disso, cada ordem de manutenção $i \in O$ requer um tempo de execução p_i e está associada a um equipamento E_i e a uma janela de tempo $[\alpha_i, \beta_i]$. Esta janela de tempo representa o intervalo de tempo disponível para a ordem de manutenção ser executada. Caso a ordem de manutenção $i \in O$ não seja realizada, é incorrida uma penalidade w_i. Cada equipe de trabalho $k \in T$ tem apenas uma habilidade e a disponibilidade de trabalhar no intervalo $[0, h_k]$ horas. O problema tem as seguintes características: *(i)* cada ordem de manutenção deve ser alocada simultaneamente a uma equipe de trabalho e a um equipamento; *(ii)* cada equipe de trabalho pode executar exatamente uma ordem de manutenção por vez; *(iii)* cada equipamento não pode ter mais de uma ordem de manutenção sendo executada simultaneamente; *(iv)* cada equipe tem um limite de horas que pode trabalhar; *(v)* cada ordem de manutenção deve ser executada exatamente dentro de uma janela de tempo; e *(vi)* cada ordem de manutenção requer uma habilidade específica para ser realizada. O objetivo do PPOMPLP é minimizar o número de equipes de trabalho ativas e o somatório das penalidades das ordens de manutenção não executadas.

A Figura 1 ilustra uma solução factível para uma instância-teste, composta por sete ordens de manutenção, três equipes de trabalho e três equipamentos. Os dados de entrada para a instância-teste são apresentados nas Tabelas 1 e 2. A Tabela 1 mostra, para cada ordem de manutenção, o número #O da ordem de manutenção, a habilidade Hab_i necessária para executar a manutenção e demais parâmetros, de acordo com a notação apresentada. A Tabela 2 apresenta, para cada equipe, o número da equipe de trabalho (coluna #T), a habilidade Hab_k que a equipe de trabalho possui e a disponibilidade h_k de horas que a equipe de trabalho pode prestar.

Tabela 1: Dados de entrada das ordens de manutenção.

#O	Hab_i.	E_i	α_i	β_i	p_i	w_i
1	2	213	3	9	4	72
2	2	199	2	7	3	132
3	1	199	1	5	2	20
4	1	342	2	6	3	30
5	2	213	0	10	5	212
6	1	213	0	3	2	168
7	2	199	4	8	3	40

Tabela 2: Dados de entrada das equipes de trabalho.

#T	Hab_k	h_k
1	1	6
2	2	10
3	2	8

A Figura 1 pode ser dividida em três partes. As partes (a) e (b) ilustram o sequenciamento das ordens de manutenção nas equipes de trabalho com e sem janelas de tempo, respectivamente. Em ambas as partes, observa-se que a equipe de trabalho 1 executa, nesta ordem, as ordens de manutenções 6 e 4, enquanto que a equipe 2 executa, nesta ordem, as ordens 2 e 5, e a equipe 3 executa a ordem 7. Nota-se, também, que as ordens 6 e 4 estão alocadas na equipe de trabalho 1, porque esta equipe possui a habilidade necessária para executar as duas ordens. As demais ordens estÃ£o alocadas nas equipes de trabalho 2 e 3, pois requerem a habilidade 2. As barras em cor cinza associadas às equipes de trabalho representam o limite de horas que cada equipe pode trabalhar. Apenas na parte (a) desta figura, a barra na parte superior da cada ordem de manutenção corresponde Ã janela de tempo. A parte (c) mostra o sequenciamento das ordens de manutenção em relação aos equipamentos. Nesta parte verifica-se que as ordens 2 e 7, nesta ordem, estão alocadas no equipamento 199, as ordens 6 e 5, nesta ordem, estão alocadas no equipamento 213, e a ordem 4 está alocada no equipamento 342. Note que, evidentemente, os tempos de início e

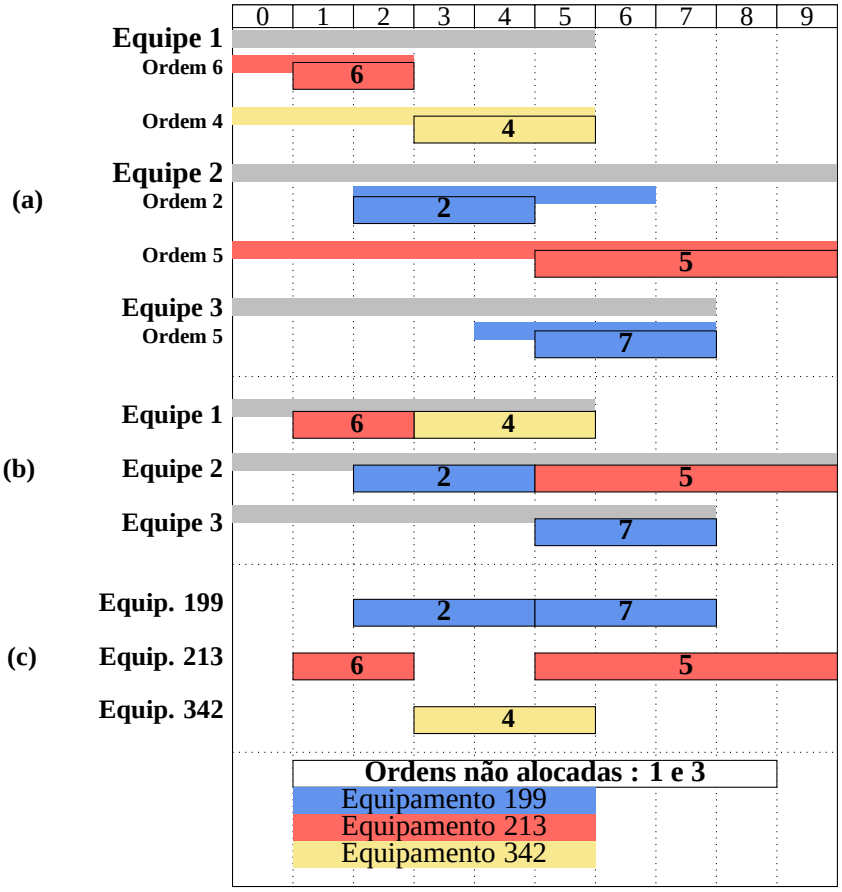

Figura 1: Exemplo de solução factível para o PPOMPLP. Sequenciamento das ordens de manutenção nas equipes (a) com e (b) sem janelas de tempo e (c) sequenciamento das ordens de manutenção nos equipamentos.

conclusão das ordens de manutenção são os mesmos nas equipes de trabalho e equipamentos. Esta divisão da figura pretende instruir as principais características e a complexidade do problema. Na parte superior da figura está a linha do tempo, no qual cada número representa uma hora. Na parte inferior da figura, as três cores diferentes identificam em qual equipamento será realizada a ordem de manutenção. Note que na solução apresentada na Figura 1, as ordens de manutenção 1 e 3 não são alocadas em nenhuma equipe de trabalho e nenhum equipamento.

4. Abordagem de solução

A solução do PPOMPLP é representada de forma indireta através de um vetor $s = \langle s_1, \ldots, s_o \rangle$ com $|O|$ posições. Semelhante a Aquino et al. [2019], este artigo optou utilizar a representação indireta da solução, pois simplifica a construção de soluções, bem como a exploração do espaço de soluções viáveis utilizando operadores de vizinhança mais simples em relação à representação direta (lista de ordens de manutenção para cada equipe de manutenção e equipamento). Cada posição deste vetor possui uma ordem de manutenção. A posição em que cada ordem de manutenção s_i apresenta no vetor solução s é definida pelo algoritmo GRASP mostrado na Seção 4.2.1.

4.1. Algoritmo de posicionamento de ordens de manutenção

Para gerar um solução factível para o PPOMPLP com o vetor solução s é executado o algoritmo de posicionamento de ordens de manutenção (Algoritmo 1). Esse algoritmo é responsável por (i) alocar as ordens de manutenção nas equipes de trabalho, e (ii) definir o tempo de conclusão das manutenções alocadas nas equipes. O pseudocódigo deste algoritmo é apresentado no Algoritmo 1.

Algoritmo 1 Algoritmo de posicionamento de ordens de manutenção

Entradas: Dados do problema, solução s, lista Γ_i para manutenção i

1: $APOM \longleftarrow 0$
2: **for** cada equipe $k \in T$ **do**
3: $\quad z_k \leftarrow false$
4: **end for**
5: **for** cada manutenção s_i para $i \leftarrow 1$ até o **do**
6: $\quad manutencaoAlocada \leftarrow false$
7: $\quad$ **for** cada equipe de manutenção $k \in \Gamma_{s_i}$ da posição 1 até a posição γ_i **do**
8: $\quad\quad$ Encontre um intervalo igual ao tempo de execução da manutenção s_i dentro da janela da mesma ordem, em que nenhuma outra manutenção é executada ao mesmo tempo na equipe k e no respectivo equipamento. A busca inicia no final da janela da manutenção e finaliza no início da janela;
9: $\quad\quad$ **if** intervalo encontrado **then**
10: $\quad\quad\quad manutencaoAlocada \longleftarrow true$
11: $\quad\quad\quad$ Aloca a manutenção s_i na equipe k
12: $\quad\quad\quad$ Aloca a manutenção s_i no respectivo equipamento;
13: $\quad\quad\quad$ **if** z_k = false **then**
14: $\quad\quad\quad\quad z_k \longleftarrow true$
15: $\quad\quad\quad\quad APOM \longleftarrow APOM + 1$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad\quad$ Retorne para linha 5 (próxima ordem de manutenção)
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: $\quad$ **if** $manutencaoAlocada = false$ **then**
21: $\quad\quad APOM \longleftarrow APOM + w_{s_i}$
22: $\quad$ **end if**
23: **end for**

O Algoritmo 1 tem como entrada os dados do problema, e também, para cada manutenção i, uma lista de prioridades $\Gamma_i = (\Gamma_i^1, ..., \Gamma_i^{\gamma_i})$ de γ_i posições, com cada posição representando uma equipe. Cada lista Γ_i possui apenas as equipes de manutenção capazes de executar a manutenção i. A lista de prioridades Γ_i é baseada na ordenação decrescente do valor de h_k, para $k \in \mathscr{T}_i$; assim, as equipes com maior valor de h_k são as primeiras da lista.

O algoritmo, primeiramente, inicializa o valor da função objetivo $(APOM)$ igual a zero (linha 1). Além disso, todas as equipes de manutenção são inicializadas desativadas (linhas 2 a 4). Na linha 5, inicia-se um processo iterativo que, em cada iteração, tenta alocar apenas uma ordem de manutenção. Na linha 6, é inicializada uma variável booleana que controla se a manutenção é alocada (*true*) ou não (*false*). O próximo passo do algoritmo consiste em um processo de busca por um intervalo viável para alocar a ordem de manutenção s_i na equipe $\Gamma_{s_i}^1$. Esse processo começa tentando alocar a manutenção s_i na equipe $\Gamma_{s_i}^1$ no final da janela da respectiva ordem, ou o mais próximo disso, dependendo das manutenções já alocadas. Como próximo passo, deve-se verificar se manutenção s_i está sendo executada simultaneamente com outra ordem de manutenção no respectivo equipamento. Caso a manutenção s_i não estiver sendo executada ao mesmo tempo com nenhuma outra manutenção, ela pode ser alocada. Caso a manutenção s_i estiver sendo executada simultaneamente com outra manutenção, a mesma deve ser alocada antes do início da manutenção que faz conflito no equipamento. Como passo seguinte, o mesmo processo de verificação de conflito no equipamento deve ser executado na equipe $\Gamma_{s_i}^1$. Este processo de alocar a manutenção s_i

na equipe $\Gamma^1_{s_i}$ e verificar no respectivo equipamento, ou alocar a manutenção s_i no equipamento e verificar na equipe $\Gamma^1_{s_i}$, deve ser repetido iterativamente, retrocedendo até o início da janela, até que seja possível alocar a manutenção s_i dentro da sua janela de execução. Caso não seja possível alocar a manutenção s_i na equipe $\Gamma^1_{s_i}$, deve-se repetir o mesmo procedimento com a equipe $\Gamma^2_{s_i}$. O mesmo procedimento executado na equipe $\Gamma^1_{s_i}$ deve ser repetido nas outras equipes pertencentes à lista Γ_{s_i}, até que se esgotem as equipes disponíveis. Se não for possível alocar a manutenção s_i em nenhuma equipe, é somado o valor da penalidade em não executar a respectiva manutenção na função objetivo (linha 18), e o algoritmo segue para a próxima manutenção da solução s. O objetivo desta estratégia é tentar alocar o maior número possível de manutenções em uma equipe, seguindo para a próxima equipe apenas quando não for mais possível encontrar intervalos viáveis. Se for encontrado um intervalo viável para manutenção s_i (ou seja, é possível alocar a manutenção), ela é alocada na equipe $k \in \Gamma_{s_i}$ (linha 11) e no respectivo equipamento (linha 12), e a variável $manutencaoAlocada$ recebe *true* (linha 10). Além disso, se a manutenção s_i for a primeira a ser alocada à equipe $k \in \Gamma_{s_i}$, o valor de $APOM$ deve ser incrementado em mais uma unidade (linha 15). A cada execução completa do algoritmo de posicionamento de ordens de manutenção tem-se uma solução factível para o PPOMPLP.

O presente artigo e o trabalho de Aquino et al. [2019] consideram a representação indireta da solução com um vetor s, e um algoritmo que utiliza esse vetor para obter uma solução para o problema. No entanto, o presente trabalho propõe novas estruturas para ambos. Neste artigo, o algoritmo de posicionamento de ordens de manutenção busca inicialmente alocar as manutenções no final da janela, diferentemente de Aquino et al. [2019], que tenta alocar primeiro as ordens no início da janela. Além disso, o algoritmo proposto neste trabalho cria uma lista de prioridade de equipes para cada ordem de manutenção, enquanto que o algoritmo de Aquino et al. [2019] não utiliza esta estratégia. Com relação ao vetor solução s, em Aquino et al. [2019] este vetor é previamente definido conforme os valores das penalidades das manutenções, enquanto que no presente artigo ele é determinado através do algoritmo GRASP.

4.2. O algoritmo metaheurístico híbrido GRASP-ILS

Esta seção é dedicada em descrever o algoritmo GRASP-ILS proposto. Este algoritmo consiste, basicamente, em primeiro executar um algoritmo GRASP para obter um vetor solução s de boa qualidade e, em seguida, aplicar o algoritmo ILS com o vetor s. O pseudocódigo do algoritmo metaheurístico proposto é apresentado no Algoritmo 2.

O algoritmo GRASP-ILS inicia-se executando o algoritmo de posicionamento de ordens de manutenção (Algoritmo 1) utilizando um vetor s_o (linha 1). A posição de cada manutenção neste vetor é definida aleatoriamente. Na linha 2, a variável referente a melhor solução f^* é inicializado com valor da função objetivo ($APOM(s_o)$) dada pelo algoritmo de posicionamento de ordens de manutenção com s_o, enquanto que na linha 3 a solução s^* é inicializada recebendo s_o. O contador de iterações it é inicializado em zero na linha 4. Entre as linhas 5 e 15 é executado o algoritmo GRASP detalhado na Seção 4.2.1. O contador de iterações it novamente recebe zero na linha 16. As linhas 17 a 31 estão associadas ao processo iterativo do algoritmo ILS, que é explicado na Seção 4.2.3.

4.2.1. *Greedy randomized adaptive search procedure* (GRASP)

O GRASP é um algoritmo iterativo, introduzido por Feo e Resende [1989], que consiste em duas fases: uma fase de construção, que gera uma solução inicial para o problema, e uma fase de busca local, que tem objetivo de refinar uma solução. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado. A eficiência da fase de busca local depende da qualidade da solução construída. O procedimento de construção tem então um

Algoritmo 2 GRASP-ILS

Entradas: solução aleatória s_o, parâmetros $GRASPmax$, $ILSmax$, $Rmax$, K

1: $AlgoritmoPosicionamentoOrdensManutencao(s_o)$
2: $f^* \longleftarrow APOM(s_o)$
3: $s^* \longleftarrow s_o$
4: $it \longleftarrow 0$
5: **while** $(it \leq GRASPmax)$ **do** ▷ veja seção 4.2.1
6: $s \longleftarrow FaseConstrucao(K)$
7: **if** $(APOM(s) < f^*)$ **then**
8: $s^* \longleftarrow s$
9: $f^* \longleftarrow APOM(s)$
10: $it \longleftarrow 0$
11: **else**
12: $it \longleftarrow it + 1$
13: **end if**
14: **end while**
15: $FaseBuscaLocal(s^*, Rmax)$
16: $it \longleftarrow 0$
17: $nivel \longleftarrow 1$
18: $MelhorIt \longleftarrow it$
19: **while** $(it - MelhorIt < ILSmax)$ **do** ▷ veja seção 4.2.3
20: $it \longleftarrow it + 1$
21: $s^{'} \longleftarrow Perturbacao(s^*, nivel)$
22: $s^{''} \longleftarrow BuscaLocal(s^{'}, Rmax)$
23: **if** $(APOM(s^{''}) < f^*)$ **then**
24: $s^* \longleftarrow s^{''}$
25: $f^* \longleftarrow APOM(s^{''})$
26: $MelhorIt \longleftarrow it$
27: $nivel \longleftarrow 1$
28: **else**
29: $nivel \longleftarrow nivel + 1$
30: **end if**
31: **end while**

papel importante na busca local, uma vez que as soluções construídas constituem bons pontos de partida para a busca local.

O algoritmo GRASP implementado neste artigo difere da versão proposta por Feo e Resende [1989] por executar apenas a fase de construção em cada iteração. A fase de busca local é executada apenas uma vez, após a conclusão do processo iterativo na linha 15. Através desta estratégia, é possível explorar melhor o algoritmo de posicionamento de manutenções e obter um tempo relativamente reduzido de execução do algoritmo metaheurístico. Esta estratégia é justificada devido ao alto custo computacional da fase de busca local. Executar esta fase em todas as iterações elevaria consideravelmente o tempo computacional do algoritmo GRASP, principalmente nas instâncias de maior escala. Além disso, o algoritmo de posicionamento de ordens de manutenção é capaz de gerar boas soluções em um tempo relativamente curto.

O procedimento iterativo entre as linhas 5 e 14 do Algoritmo 2 é repetido até que um número $GRASPmax$ de iterações sem melhora de solução seja atingido. Na linha 6 é implementada a fase da construção, que consiste em alocar todas as manutenções disponíveis em uma posição no vetor solução s e, em seguida, executar o algoritmo de posicionamento de ordens de manutenção. As manutenções são adicionadas sequencialmente (primeiro na posição 1, depois na 2, e assim em diante) na solução s através de um processo iterativo. Este processo de adição de manutenções utiliza uma lista de candidatos (LC) e uma lista restrita de candidatos (LRC). A LC é uma lista com todas as ordens de manutenções, ordenada de forma decrescente de acordo com o valor da penalidade. A LRC possui K posições e é composta pelas K manutenções de maio-

res valores de penalidades da LC. Após a construção da LRC, inicializa-se um processo iterativo. Este processo tem os seguintes passos: (i) selecione aleatoriamente uma manutenção da LRC; (ii) adicione a manutenção selecionada na solução s; (iii) remova a manutenção selecionada da LRC; e (iv) insira na LRC a manutenção de maior penalidade ainda não inserida. Esse procedimento é repetido até que todas as ordens de manutenção disponíveis sejam alocadas. Com o vetor solução s determinado, inicia-se a segunda parte da fase de construção, que consiste em executar o algoritmo de posicionamento de ordens de manutenção para obter uma solução factível para o PPOMPLP.

4.2.2. Busca local

Como as instâncias consideradas para o PPOMPLP são baseadas em casos reais e de larga escala, analisar todos os vizinhos nas instâncias de maior dimensão é totalmente impraticável. Para contornar essa dificuldade, a fase de busca local implementa o método da descida randômica. Ele consiste em analisar um vizinho qualquer, selecionado aleatoriamente, e o aceitar caso ele tenha um valor de função objetivo estritamente menor que a melhor solução obtida até então. Caso contrário, a melhor solução permanece inalterada e outro vizinho é gerado aleatoriamente. Este procedimento é interrompido após um número de iterações $Rmax$ sem melhora no valor da melhor solução obtida até então.

Para modificar a solução e explorar o espaço de solução, é utilizado apenas o movimento de troca. Ele consiste em trocar a ordem de manutenção s_j da posição j do vetor solução s, com a ordem de manutenção s_r da posição r. Para gerar um vizinho no método da descida randômica, é necessário selecionar aleatoriamente duas posições do vetor s e trocar as ordens de manutenção de posição.

4.2.3. ILS

ILS é uma metaheurística introduzida por Baxter [1981] [Lourenço et al., 2019], que envolve a aplicação iterativa de uma busca local e uma perturbação, como mecanismo de diversificação. A cada iteração, uma nova solução é gerada, através de uma pequena modificação, e refinada através de uma busca local. Para aplicar um algoritmo ILS, quatro componentes devem ser especificados: *(i)* solução inicial para o problema; *(ii)* busca local; *(iii)* perturbação; e *(iv)* critério de aceitação.

No algoritmo proposto neste artigo, a solução inicial é dada pelo processo iterativo do algoritmo GRASP (linhas 5 e 14). Na linha 15 é executada a busca local na solução inicial. Na linha 17, a variável $nivel$, que refere-se ao nível de pertubação, recebe 1, enquanto que na linha 18 a variável $MelhorIt$, que refere-se a iteração da melhor solução, recebe zero. O processo iterativo do algoritmo ILS está situado entre as linhas 19 e 31, e é finalizado após $ILSmax$ iterações sem melhora de solução. Após o contador de iterações ser incrementado em uma unidade (linha 20), inicia-se o procedimento de perturbação (linha 21). A pertubação é caracterizada por aplicar $nivel+1$ vezes o movimento de troca na solução s^*. Por exemplo, se $nivel = 1$, então o movimento de troca será aplicado duas vezes. Note que o número de modificações é maior ou igual a dois. As posições do vetor s^* em que são feitas as trocas de manutenções são selecionadas aleatoriamente. Vale lembrar que apenas um movimento de troca consiste em trocar duas manutenções de posição. Na linha 22 é executada a busca local com o método de descida randômica na solução $s^{'}$, resultante da pertubação. Caso a solução $s^{''}$, provida pela busca local, tenha um valor de função de objetivo estritamente menor que a melhor solução obtida até então (linha 23), ela deve ser aceita e passa a ser a melhor solução encontrada (linhas 24 e 25). Além disso, o nível de pertubação retorna para 1 (linha 27). Caso contrário, o nível de perturbação é incrementado em uma unidade. Após a conclusão do processo iterativo do algoritmo ILS, o algoritmo GRASP-ILS é finalizado.

5. Experimentos e resultados computacionais

Para analisar o desempenho do algoritmo GRASP-ILS proposto, foram realizados experimentos computacionais com 39 instâncias da literatura, propostas em Aquino et al. [2019]. As melhores soluções conhecidas, bem como seus tempos de execução, para cada uma destas instâncias, são providas por Aquino et al. [2019]. Como o número de manutenções nas instâncias varia de 150 a 33484, elas são divididas em três conjuntos, como mostra a Tabela 3. Nesta tabela também são apresentados os valores dos parâmetros utilizado para o algoritmo GRASP-ILS para cada conjunto de instâncias. A primeira coluna ($conj.$) desta tabela identifica os conjuntos. A segunda coluna ($\#ord.$) refere-se ao intervalo do número de manutenções de ordens de manutenção de cada conjunto de instância. As demais colunas apresentam os valores para os parâmetros do algoritmo. Estes valores foram obtidos de forma empírica após vários testes.

Tabela 3: Valores dos parâmetros utilizados para o algoritmo GRASP-ILS para cada conjunto de instâncias.

Conj.	$\#ord.$	$GRASPmax$	$ILSmax$	$Rmax$	K
P	150 a 600	100	50	800	15
M	1200 a 4800	50	30	600	15
G	9600 a 33484	25	15	200	15

Os resultados computacionais do algoritmo GRASP-ILS, juntamente com os melhores resultados encontrados por Aquino et al. [2019] são apresentados na Tabela 4. Nesta tabela, a coluna # identifica a instância. As colunas $|O|$, $|E|$ e $|T|$ representam o número de ordens de manutenção, equipamentos e equipes de manutenção, respectivamente. As colunas t_{BKS} e f_{BKS} estão associadas às soluções dadas pelos algoritmos metaheurísticos de Aquino et al. [2019] e representam o tempo de execução em segundos do algoritmo e o valor da função objetivo da melhor solução encontrada, respectivamente. A coluna t_{media} representa o tempo médio, em segundos, em 10 execuções, do algoritmo GRASP-ILS em cada instância. As colunas f_{media}, f_{desvio} e f_{melhor} estão associadas ao valor da função objetivo gerado pelo algoritmo GRASP-ILS e representam o resultado médio, o desvio padrão entre os resultados obtidos e a melhor solução encontrado pelo algoritmo, respectivamente. Por fim, as colunas $|ONA|$ e $\%TA$ representam o número de ordens não alocadas e equipes de manutenção ativadas, respectivamente, da melhor solução encontrada pelo algoritmo proposto.

Os valores em negrito na última linha (**Médias**) da Tabela 4 destacam os melhores resultados do algoritmo proposto no presente artigo em relação aos algoritmos heurísticos abordados por Aquino et al. [2019]. Essas médias demonstram que o algoritmo GRASP-ILS encontrou melhores resultados em um tempo expressivamente menor. Mesmo sendo computadores com capacidade computacional diferentes, é considerável a diferença entre os tempos. Através dos valores em negrito nas colunas f_{BKS} e f_{melhor}, verifica-se que o algoritmo GRASP-ILS encontrou limites superiores melhores em aproximadamente 80% das instâncias. Nas 20% instâncias restantes, o valor de função objetivo encontrado foi igual. Vale destacar que nas instâncias do conjunto G, e principalmente na instância real (instância 39), o algoritmo GRASP-ILS melhorou consideravelmente os limites superiores encontrados por Aquino et al. [2019]. Os novos limites superiores encontrados na maior parte das instâncias significam que as novas soluções apresentam um menor percentual de equipes ativadas e menor número de ordens de manutenção não executadas. Note que nas instâncias 7, 9, 14, 16, 17, 18, 22, 23, e 24, a diferença entre os valores de f_{BKS} e f_{melhor} é relativamente menor do que as demais, e isto significa que, nessas novas soluções, as ordens de manutenção foram alocadas de um modo diferente que permite um menor número de equipes de manutenção ativadas. Com relação as duas últimas colunas da Tabela 4, é possível notar que nas instâncias de menor escala, o número de ordens de manutenção não executadas é menor do que nas de maior escala. Por outro lado, nas instâncias de maior escala, o percentual de equipes ativadas é menor do que nas instâncias de menor escala.

Tabela 4: Resultados computacionais obtidos com o algoritmo GRASP-ILS proposto.

Instância				Aquino et al. [2019]*		GRASP-ILS**					
#	$\|O\|$	$\|E\|$	$\|T\|$	t_{BKS}	f_{BKS}	t_{media}	f_{media}	f_{desvio}	f_{melhor}	$\|ONA\|$	$\%TA$
1	150	148	120	150	**1756**	6,4	1756,0	0,0	**1756**	6	23%
2	150	75	129	150	**30**	5,4	30,0	0,0	**30**	0	23%
3	150	102	91	150	**3273**	5,6	3273,2	0,4	**3273**	1	30%
4	150	57	126	150	**24**	9,6	24,0	0,0	**24**	0	19%
5	150	92	93	150	**49**	4,2	49,0	0,0	**49**	0	53%
6	150	71	101	150	**106**	4,6	106,00	0,0	**106**	1	34%
7	300	158	179	300	1776	21,6	1771,00	0,0	**1771**	6	24%
8	300	221	239	300	2452	16,4	2427,00	0,0	**2427**	15	21%
9	300	112	177	300	3360	14,2	3359,00	0,0	**3359**	2	23%
10	300	75	181	300	**35**	22,0	35,00	0,0	**35**	0	19%
11	300	121	162	300	281	11,2	65,00	0,0	**65**	0	40%
12	300	119	176	300	**308**	12,6	308,00	0,0	**308**	3	23%
13	600	165	329	600	4409	78,4	4290,00	0,0	**4290**	18	16%
14	600	256	388	600	3384	59,6	3379,00	0,0	**3379**	30	19%
15	600	120	288	600	8578	68,2	5648,00	0,0	**5648**	16	18%
16	600	77	215	600	42	83,4	38,00	0,0	**38**	0	18%
17	600	126	279	600	414	37,8	410,00	0,0	**410**	2	24%
18	600	120	292	600	5663	48,0	5659,00	0,0	**5659**	7	15%
19	1200	186	519	1200	10784	125,6	10314,00	5,6	**10311**	51	10%
20	1200	263	666	1200	9527	117,8	8439,40	6,2	**8434**	79	12%
21	1200	122	470	1200	21930	108,6	18357,00	157,5	**18242**	26	6%
22	1200	88	252	1200	309	67,4	299,20	0,4	**299**	1	15%
23	1200	130	420	1200	526	101,2	539,60	35,1	**514**	3	17%
24	1200	122	403	1200	6841	36,4	6830,00	0,0	**6830**	18	13%
25	2400	188	738	2400	26705	646,0	24835,20	56,5	**24793**	137	10%
26	2400	283	869	2400	24553	282,0	22223,40	35,1	**22189**	177	13%
27	2400	130	603	2400	38094	208,0	27973,80	1611,6	**26821**	67	10%
28	2400	90	278	2400	1585	359,4	565,40	0,5	**565**	2	16%
29	2400	132	701	2400	816	199,0	629,00	5,5	**625**	4	12%
30	2400	126	561	2400	8259	143,6	8233,00	0,0	**8233**	32	11%
31	4800	197	1128	4800	59580	845,4	55842,80	248,9	**55666**	326	8%
32	4800	78	1286	4800	41925	509,2	37769,80	303,2	**37441**	300	11%
33	4800	130	720	4800	92320	856,0	58664,60	1201,0	**57927**	158	9%
34	4800	91	294	4800	198340	482,6	194910,60	39,3	**194867**	1143	11%
35	4800	135	990	4800	6008	525,6	2258,20	157,4	**2075**	14	9%
36	4800	129	713	4800	16745	446,8	12444,00	574,7	**12024**	69	9%
37	9600	132	816	9600	35778	1112,2	21894,40	953,7	**20659**	140	9%
38	19200	132	908	19200	102773	2578,8	43946,00	1365,7	**41870**	287	8%
39	33483	145	1032	33483	220048	7441,8	73151,40	2788,0	**70017**	509	7%
Médias				3050,85	24599,64	**453,91**			**16744,33**		

* Executado em um Intel Xeon CPU E5-2660v2 2.20GHz x 40 e com 384GB de memória RAM;
** Executado em um Intel Core i3 3.00GHz e com 16GB de memória RAM;

6. Conclusões

Este trabalho estudou o problema de planejamento de ordens de manutenção de longo prazo. Para solucioná-lo com eficiência e obter boas soluções, foi desenvolvido um algoritmo baseado na combinação das metaheurísticas GRASP e ILS. Na fase construtiva do algoritmo GRASP foi desenvolvido um algoritmo de posicionamento de ordens de manutenção. O algoritmo GRASP-ILS foi aplicado em instâncias da literatura e os seus resultados comparados com as melhores soluções providas em Aquino et al. [2019]. Os resultados computacionais mostraram que o algoritmo proposto no presente artigo foi capaz de obter limites superiores melhores em 80% das instâncias, em um tempo computacional consideravelmente menor. Os novos e bons limites superiores obtidos estão associados com nova estratégia para obter o vetor solução s, a combinação das duas metaheurísticas, e principalmente, com o algoritmo de posicionamento de ordens de manutenção. Este algoritmo é parte fundamental do algoritmo desenvolvido, e sua estratégia de tentar alocar as manutenções primeiramente no final da janela origina novas e diferentes soluções para o problema.

Para trabalhos futuros sugere-se testar novas estratégias com algoritmos heurísticos, que possibilitem acionar as buscas locais periodicamente sem prejuízo excessivo no tempo de processa-

mento. Sugere-se, também, trabalhar o problema com uma abordagem de métodos exatos, propondo uma nova formulação matemática.

Agradecimentos Os autores agradecem à CAPES (código de financiamento 001), ao CNPq, à FAPEMIG e ao PPGMMC/CEFET-MG pelo apoio ao desenvolvimento deste trabalho.

Referências

Aquino, R. D., Souza, M. J. F., e das Chagas, J. B. C. (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.

Baxter, J. (1981). Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32(9):815–819. URL https://doi.org/10.1057/jors.1981.159.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71. ISSN 0167-6377.

Froger, A., Gendreau, M., Mendoza, J. E., Éric Pinson, e Rousseau, L.-M. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, 251(3):695–706. ISSN 0377-2217.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2019). *Iterated Local Search: Framework and Applications*, p. 129–168. Springer International Publishing. URL https://doi.org/10.1007/978-3-319-91086-4_5.

Qamhan, A. A., Ahmed, A., Al-Harkan, I. M., Badwelan, A., Al-Samhan, A. M., e Hidri, L. (2020). An exact method and ant colony optimization for single machine scheduling problem with time window periodic maintenance. *IEEE Access*, 8:44836–44845.

Sedghi, M., Kauppila, O., Bergquist, B., Vanhatalo, E., e Kulahci, M. (2021). A taxonomy of railway track maintenance planning and scheduling: A review and research trends. *Reliability Engineering & System Safety*, 215:107827. ISSN 0951-8320.

Shaukat, S., Katscher, M., Wu, C.-L., Delgado, F., e Larrain, H. (2020). Aircraft line maintenance scheduling and optimisation. *Journal of Air Transport Management*, 89:101914. ISSN 0969-6997.

Shen, J. e Zhu, K. (2018). An uncertain single machine scheduling problem with periodic maintenance. *Knowledge-Based Systems*, 144:32–41. ISSN 0950-7051.

Wang, H. (2002). A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3):469–489. ISSN 0377-2217.

Woller, D. e Kulich, M. (2021). The ALNS metaheuristic for the maintenance scheduling problem. In *ICINCO*.

Yu, Q. e Strömberg, A.-B. (2021). Mathematical optimization models for long-term maintenance scheduling of wind power systems.

Zhang, C., Gao, Y., Yang, L., Kumar, U., e Gao, Z. (2019). Integrated optimization of train scheduling and maintenance planning on high-speed railway corridors. *Omega*, 87(C):86–104.