Trabalho apresentado no XLI CNMAC, Unicamp - Campinas - SP, 2022.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics
Preprint

Um algoritmo GRASP eficiente para um problema de planejamento de ordens de manutenção de longo prazo

João L. M. de Andrade[1], Gustavo C. Menezes, Elisangela M. de Sá, Sérgio R. de Souza
PPGMMC/CEFET-MG, Belo Horizonte, MG
Marcone J. F. Souza
Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, MG

Resumo. Este artigo estuda o problema de planejamento de ordens de manutenção de longo prazo. O objetivo deste problema é definir quais ordens de manutenção são executadas ou não, 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 ordens de manutenção não executadas. Para solucionar este problema pertencente à classe NP-difícil com instâncias baseadas em casos reais, foi desenvolvido um algoritmo GRASP. Os resultados computacionais mostram sua superioridade em relação aos métodos de solução da literatura, encontrando soluções melhores em um tempo expressivamente menor.

Palavras-chave. Planejamento de longo prazo, Sequenciamento de manutenção preventiva, GRASP, Metaheurística, Otimização combinatória

1 Introdução

Os sistemas industriais estão, em geral, sujeitos à degradação devido à utilização. Esta degradação pode conduzir a uma falha do sistema e, por sua vez, resultar em problemas de segurança, danos aos equipamentos e indisponibilidade inesperada dos mesmos. Portanto, cada vez mais as organizações percebem que podem melhorar a sua eficiência através de um planejamento mais eficaz das manutenções preventivas. Os benefícios importantes da manutenção e sua associação com pesquisas na área da pesquisa operacional motivou pesquisadores no desenvolvimento de novos estudos.

Este artigo estuda o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP). 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 $\mathscr{T}_i \subseteq T$ 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, incorre-se em 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 PPOMPLP tem como objetivo minimizar o número de equipes de trabalho ativas e o somatório das penalidades das ordens de manutenção não executadas, satisfazendo às seguintes restrições: (i) cada ordem de manutenção tem que ser alocada simultaneamente a uma equipe de trabalho

[1] andrad.joao15@gmail.com

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.

Em [1], o PPOMPLP é reduzido ao problema de sequenciamento em máquinas paralelas não-relacionadas, que é um problema NP-difícil. Além disso, este artigo citado considera instâncias baseadas em dados reais. Essas considerações motivam a pesquisa de novos e eficientes métodos de solução. Assim, para tratar este problema, o presente artigo propõe um algoritmo baseado na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) capaz de gerar resultados superiores do que os métodos de solução de [1]. Os resultados mostram que o algoritmo proposto encontra soluções melhores com tempo de processamento significativamente menor.

2 Revisão da Literatura

As pesquisa em problemas de sequenciamento de manutenção se estende por diversas ramificações de linhas de pesquisas na área da pesquisa operacional. No entanto, existem diferentes problemas que foram estudados que tangenciam o problema investigado neste trabalho.

De acordo com [7], a manutenção pode ser classificada em dois tipos básicos: (a) manutenção corretiva e (b) manutenção preventiva. A manutenção corretiva significa fazer a manutenção de um componente após a ocorrência de uma falha. Por outro lado, a manutenção preventiva significa fazer a manutenção de um componente antes que este apresente uma falha; estas manutenções devem ser previamente agendadas. [8] combinam um modelo matemático de programação de manutenção preventiva com estratégias de manutenção corretiva. Os resultados desse trabalho mostram vantagens significativas quando considerada uma estratégia com manutenções corretivas e preventivas, ao invés de apenas manutenções corretivas. Em um problema diferente, mas ainda sobre o sequenciamento de manutenções preventivas, [5] abordam um problema de sequenciamento e alocação de equipes de reparadores em máquinas em deterioração. Neste problema, as equipes possuem habilidades específicas para certas máquinas. O trabalho de [4] trata o problema de sequenciamento de uma máquina com manutenção periódica dentro de janelas de tempo.

Os problemas de sequenciamento de manutenção possuem diversas aplicações. A indústria de eletricidade é uma delas, como mostra a revisão literária de [3]. [9] estudam um problema integrado de programação de trens e planejamento de manutenções. No trabalho de [6], as manutenções devem ser planejadas dentro de janelas de tempo determinadas pela programação de aeronaves.

Em um problema diferente dos revisados, porém ainda pertencente à mesma linha de pesquisa, [1] fazem um estudo de caso do planejamento de manutenções preventivas de uma unidade de beneficiamento de minério de ferro no Brasil. O trabalho tinha como objetivo otimizar o planejamento de um conjunto de manutenções preventivas que devem ser executadas em um mapa 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. Os autores desenvolveram um modelo matemático, porém, para solucionar o problema em instâncias de larga escala, foram implementados diferentes algoritmos metaheurísticos. As soluções geradas pelos algoritmos foram capazes de produzir um planejamento melhor do que os empregados pela empresa.

O presente artigo estuda o problema pesquisado por [1]. Como contribuição, apresenta-se um novo e eficiente método de solução e, como consequência, novos limites superiores para o problema. O método de solução proposto neste artigo é um algoritmo GRASP, com uma modificação na versão clássica, que utiliza um novo algoritmo de posicionamento de ordens de manutenção.

DOI: 10.5540/03.2022.009.01.0314

3 Abordagem de solução

3.1 Representação da solução

Para representar a solução s do PPOMPLP, é utilizado um vetor $s = \langle s_1, \ldots, s_o \rangle$ com $|O|$ posições, no qual cada posição representa uma ordem de manutenção. Como em [1], este artigo optou por utilizar esta representação indireta da solução, pois simplifica a construção de soluções e 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). A sequência em que as manutenções são alocadas nas posições do vetor solução s é definida pelo algoritmo GRASP apresentado na seção 3.3.

3.2 Algoritmo de posicionamento de ordens de manutenção

Para gerar uma solução do PPOMPLP com a solução s é executado o algoritmo de posicionamento de ordens de manutenção. Esse algoritmo, tem como objetivos, (i) alocar as ordens de manutenção nas equipes de trabalho; (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

```
   Input: Dados do problema, solução s, lista Γ_i para manutenção i
1  begin
2      APOM ← 0
3      for cada equipe k ∈ T do
4          z_k ← false
5      for cada manutenção s_i para i ← 1 até o do
6          manutencaoAlocada ← false
7          for cada equipe de manutenção k ∈ Γ_{s_i} da posição 1 até a posição γ_i do
8              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              if intervalo encontrado then
10                 manutencaoAlocada ⟵ true
11                 Aloca a manutenção s_i na equipe k;
12                 Aloca a manutenção s_i no respectivo equipamento;
13                 if z_k = false then
14                     z_k ⟵ true
15                     APOM ⟵ APOM + 1;
16                 Retorne para linha 5 (próxima ordem de manutenção);
17         if manutencaoAlocada = false then
18             APOM ⟵ APOM + w_{s_i};
```

Além dos dados do problema e a solução s, o Algoritmo 1 tem, como entrada, 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 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 2). As equipes são inicializadas desativadas (linhas 3 e 4). Na linha 5, inicia-se um processo iterativo que, em cada iteração, tenta alocar apenas uma ordem de manutenção. A linha 6 inicializa uma variável booleana que controla se a manutenção é alocada (*true*) ou não (*false*). No próximo passo do algoritmo, inicia-se o processo de busca por um intervalo viável para alocar a ordem de manutenção s_i na equipe $\Gamma_{s_i}^1$. O processo começa tentando alocar a manutenção s_i no final da janela, ou o mais próximo. Caso não seja possível alocar a ordem de manutenção s_i, a busca deve retroceder, baseado na posição das demais ordens no equipamentos e nas equipes, até o início da janela. Caso não seja possível alocar a ordem de manutenção s_i na equipe $\Gamma_{s_i}^1$, deve-se repetir o mesmo procedimento com a equipe $\Gamma_{s_i}^2$. Todo esse procedimento deve ser repetido até que se esgotem as equipes capazes de executar a a ordem de manutenção s_i. Se não for possível alocar

DOI: 10.5540/03.2022.009.01.0314

a ordem de manutenção s_i em nenhuma equipe de manutenção, é somado o valor da penalidade em não executar a manutenção na função objetivo (linha 18), e o algoritmo segue para a próxima ordem de 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.

Caso a ordem de manutenção s_i possa ser alocada, ou seja, um intervalo foi encontrado, ela é alocada na equipe k (linha 11) e no respectivo equipamento (linha 12), e a variável booleana recebe *true* (linha 10). Além disso, caso a ordem de manutenção s_i seja a primeira a ser alocada à equipe k, o valor da função objetivo deve ser incrementado em mais uma unidade (linha 15). Com a conclusão do algoritmo de posicionamento de ordens, tem-se uma solução factível para o PPOMPLP.

O algoritmo de posicionamento de ordens de manutenção aplica uma estratégia diferente da utilizada por [1]. No presente trabalho, o algoritmo busca inicialmente alocar as ordens de manutenção no final da janela, ao contrário de [1], que aloca primeiro as ordens de manutenção 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 [1] não utiliza esta estratégia. Outra diferença está no vetor solução s. Enquanto em [1] este vetor é previamente definido de acordo com as penalidades das ordens de manutenção, no presente artigo ele é determinado por um algoritmo GRASP.

3.3 GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) é um algoritmo *multi-start* que foi introduzido por [2]. Cada iteração da metaheurística GRASP proposta por [2] é constituída por uma fase de construção e uma fase de busca local. Na fase de construção, determina-se uma solução inicial para o problema. A fase de busca local tem o objetivo de refinar uma solução. O melhor ótimo local encontrado em todas as iterações é salvo como melhor solução encontrada.

Diferentemente de [2], o algoritmo proposto no presente trabalho não executa a fase de busca local após a fase construtiva, mas, sim, apenas uma vez, e com a melhor solução dada em um processo iterativo que não encontrou melhoria na solução após um número de iterações. Com esta 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 adotada pois a fase de busca local é muito custosa computacionalmente e executá-la em todas as iterações elevaria muito 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 pequeno. O Algoritmo 2 mostra o pseudocódigo do algoritmo GRASP proposto neste trabalho.

Algoritmo 2: *GRASP*

```
   Entradas: ITMax.
1  f* ⟵ ∞
2  s* ⟵ ∅
3  it ⟵ 0
4  while (it ≤ ITmax) do
5      Construcao(s, APOM(s))
6      if (APOM(s) < f*) then
7          s* ⟵ s
8          f* ⟵ APOM(s)
9          it ⟵ 0
10     else
11         it ⟶ t + 1
12 BuscaLocal(s*)
```

DOI: 10.5540/03.2022.009.01.0314

A primeira etapa da fase de construção consiste em alocar as manutenções disponíveis em uma posição no vetor solução s. As manutenções são adicionadas sequencialmente (primeiro na posição 1, depois na 2, e assim em diante) no vetor s através de um processo iterativo. Este processo de adição de manutenções utiliza uma lista de candidatos (LC), que é uma lista com todas as ordens de manutenções, ordenada de forma decrescente de acordo com o valor da penalidade, e uma lista restrita de candidatos (LRC), que possui K posições. A LRC é composta pelas K manutenções de maiores valores de penalidades da LC. Após a construção da LRC, inicializa-se o 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 definido, inicia-se a segunda parte da fase de construção, que executa o algoritmo de posicionamento de ordens de manutenção para obter uma solução para o problema.

As instâncias consideradas neste trabalho são baseadas em casos reais, no qual o número de ordens de manutenção varia entre 150 e 33484. Logo, analisar todos os vizinhos nas instâncias maiores é totalmente inviável. Para contornar isso, 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 ($itmax$) 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 Experimentos e resultados computacionais

Para analisar o desempenho do algoritmo GRASP proposto, foram realizados experimentos computacionais com 39 instâncias da literatura, propostas em [1]. A melhores soluções conhecidas, bem como seus tempos de execução, para cada uma destas instâncias, são providas por [1].

Os parâmetros utilizados são os seguintes: (i) critério de parada $GRASPmax = 25$, que representa um número máximo de iterações sem melhora; (ii) número máximo de ordens de manutenções na LRC, $K = 10$; e (iii) critério de parada $itmax = 500$ associado ao método de descida randômica, representa o número máximo de iterações sem melhora. Os valores para os parâmetros foram obtidos após inúmeros testes e notou-se que, com estes valores, obteve-se melhor benefício entre o tempo de execução e a solução gerada.

Na Tabela 1 são apresentados os resultados computacionais do algoritmo GRASP proposto, juntamente com os melhores resultados encontrados por [1]. Nesta tabela, a coluna id identifica a instância. As colunas #manutenções, #equipamentos e #equipes representam o número de ordens de manutenção, de equipamentos e de equipes de trabalho, respectivamente. As colunas f_{BKS} e t_{BKS} estão associadas às soluções dadas pelos algoritmos metaheurísticos de [1] e representam o valor da função objetivo da melhor solução encontrada e o tempo de execução em segundos do algoritmo, respectivamente. As colunas f_{media}, f_{melhor} e f_{desvio} estão associadas ao valor da função objetivo gerado pelo algoritmo GRASP e representam o resultado médio, o melhor resultado encontrado pelo algoritmo e o desvio padrão entre os resultados obtidos, respectivamente. A coluna t_{media} representa o tempo médio, em segundos, em 10 execuções, do algoritmo em cada instância.

Os valores em negrito na última linha (médias) da Tabela 1 destacam os melhores resultados e a diferença dos algoritmos heurísticos utilizados por [1] e o algoritmo GRASP proposto neste trabalho. Essas médias em negrito tornam notável que o algoritmo proposto encontrou melhores resultados em um tempo significativamente menor. Pelos valores destacados em negrito nas

DOI: 10.5540/03.2022.009.01.0314

Tabela 1: Resultados computacionais obtidos com o algoritmo GRASP proposto.

Instância				Aquino et al. [1]*		**GRASP****			
id	#manutenções	#equipes	#equipamentos	f_{BKS}	t_{BKS}	f_{media}	f_{desvio}	f_{melhor}	t_{medio}
1	150	148	120	**1756**	150	1756,0	0,0	**1756**	0,13
2	150	75	129	**30**	150	30,0	0,0	**30**	0,09
3	150	102	91	**3273**	150	3273,0	0,0	**3273**	0,18
4	150	57	126	**24**	150	24,0	0,0	**24**	0,20
5	150	92	93	**49**	150	49,0	0,0	**49**	0,09
6	150	71	101	**106**	150	106,0	0,0	**106**	0,08
7	300	158	179	**1776**	300	1916,4	41,1	1898	0,31
8	300	221	239	2452	300	2451,0	0,0	**2451**	0,28
9	300	112	177	**3360**	300	3360,0	0,0	**3360**	0,22
10	300	75	181	**35**	300	35,0	0,0	**35**	0,38
11	300	121	162	281	300	65,0	0,0	**65**	0,24
12	300	119	176	**308**	300	308,0	0,0	**308**	0,20
13	600	165	329	4409	600	4464,0	104,8	**4333**	2,28
14	600	256	388	3384	600	3377,6	16,6	**3349**	1,62
15	600	120	288	8578	600	8547,4	64,0	**8433**	0,91
16	600	77	215	42	600	39,0	0,0	**39**	1,24
17	600	126	279	414	600	410,0	0,0	**410**	0,69
18	600	120	292	5663	600	5659,8	0,4	**5659**	1,10
19	1200	186	519	10784	1200	10599,0	284,5	**10294**	19,33
20	1200	263	666	9527	1200	9533,6	116,0	**9357**	11,08
21	1200	122	470	21930	1200	19138,4	351,5	**18571**	6,23
22	1200	88	252	309	1200	303,0	0,0	**303**	4,62
23	1200	130	420	**526**	1200	602,8	61,7	**526**	4,06
24	1200	122	403	6841	1200	6830,0	0,0	**6830**	2,15
25	2400	188	738	26705	2400	25284,8	327,1	**24980**	183,36
26	2400	283	869	24553	2400	22262,0	81,3	**22202**	37,53
27	2400	130	603	38094	2400	32034,8	1622,8	**30737**	22,54
28	2400	90	278	1585	2400	628,8	93,9	**570**	16,11
29	2400	132	701	816	2400	680,8	33,6	**626**	15,90
30	2400	126	561	8259	2400	8234,6	0,5	**8234**	6,23
31	4800	197	1128	59580	4800	54981,8	126,6	**54859**	389,20
32	4800	78	1286	41925	4800	39044,4	842,7	**37538**	196,60
33	4800	130	720	92320	4800	66116,6	3978,6	**62395**	81,13
34	4800	91	294	198340	4800	199613,8	1327,5	**197399**	67,02
35	4800	135	990	6008	4800	2546,0	401,8	**1887**	59,19
36	4800	129	713	16745	4800	12320,6	435,2	**12024**	20,80
37	9600	132	816	35778	9600	22724,4	736,3	**21860**	129,05
38	19200	132	908	102773	19200	43873,8	266,1	**43603**	510,40
39	33483	145	1032	220048	33483	74207,4	2142,7	**72369**	1537,60
		Médias		24599,64	3050,85			**17249,79**	**85,39**

* Executado em um Intel Xeon CPU E5-2660v2 2.20GHz x 40 e com 384GB de memória RAM;
** Executado em um Intel Core i3 1.1GHz e com 8GB de memória RAM;

colunas f_{BKS} e f_{melhor}, nota-se que o algoritmo GRASP encontrou limites superiores melhores em aproximadamente 72% das instâncias. Além disso, em 26% das instâncias o valor de função objetivo encontrado foi igual nas duas pesquisas. Apenas na instância 7 a abordagem de [1] obteve uma solução melhor. Vale destacar que nas instâncias de maior escala (instâncias 37 e 38), e principalmente na real (instância 39), o algoritmo GRASP superou os demais algoritmos e encontra melhores limites superiores com considerável diferença.

Os novos limites superiores encontrados na maior parte das instâncias indicam que as novas soluções possuem um menor número de equipes ativadas e menor percentual de ordens de manutenção não executadas. Apenas nas instâncias 8, 16, 17, 18 e 22 a diferença entre os limites superiores encontrados anteriormente e os novos 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.

5 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 metaheurística GRASP. Para a fase construtiva deste algoritmo foi elaborado um algoritmo de posicionamento de ordens de manutenção. O algoritmo proposto foi aplicado em instâncias da literatura, e os seus resultados comparados com as melhores soluções reportadas em [1]. Os resultados

computacionais mostraram que o algoritmo GRASP proposto foi capaz de obter limites superiores melhores em 72% das instâncias, em um tempo computacional significativamente inferior. Os novos e bons limites superiores obtidos estão associados principalmente com a estratégia para obter o vetor solução s pelo GRASP e com o algoritmo de posicionamento de ordens de manutenção. Este algoritmo é parte fundamental da metaheurística, e sua estratégia de tentar alocar as manutenções primeiramente no final da janela origina novas e diferentes soluções para o problema. Os baixos tempos computacionais estão justamente relacionados com a estratégia adotada, de aplicar a fase de busca local apenas na melhor das soluções construídas.

Para trabalhos futuros sugere-se testar novas estratégias para melhorar ainda mais o desempenho do algoritmo GRASP proposto, como por exemplo, acionar as buscas locais periodicamente sem prejuízo excessivo no tempo de processamento. 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

[1] Roberto Dias Aquino, Marcone Jamilson Freitas Souza e Jonatas Batista Costa das Chagas. "Abordagem exata e heurısticas para o problema de planejamento de ordens de manutenção de longo prazo: um estudo de caso industrial de larga escala". Em: **Pesquisa Operacional para o Desenvolvimento** 11.3 (2019), pp. 159–182. URL: http://doi.editoracubo.com.br/10.4322/PODes.2019.012.

[2] Thomas A. Feo e Mauricio G.C Resende. "A probabilistic heuristic for a computationally difficult set covering problem". Em: **Operations Research Letters** 8.2 (1989), pp. 67–71. ISSN: 0167-6377. DOI: https://doi.org/10.1016/0167-6377(89)90002-3.

[3] Aurélien Froger et al. "Maintenance scheduling in the electricity industry: A literature review". Em: **European Journal of Operational Research** 251.3 (2016), pp. 695–706. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2015.08.045.

[4] Ammar A. Qamhan et al. "An Exact Method and Ant Colony Optimization for Single Machine Scheduling Problem With Time Window Periodic Maintenance". Em: **IEEE Access** 8 (2020), pp. 44836–44845. DOI: 10.1109/ACCESS.2020.2977234.

[5] Diego Ruiz-Hernández, Jesús M. Pinar-Pérez e David Delgado-Gómez. "Multi-machine preventive maintenance scheduling with imperfect interventions: A restless bandit approach". Em: **Computers & Operations Research** 119 (2020), p. 104927. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2020.104927.

[6] Syed Shaukat et al. "Aircraft line maintenance scheduling and optimisation". Em: **Journal of Air Transport Management** 89 (2020), p. 101914. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2020.101914.

[7] Hongzhou Wang. "A survey of maintenance policies of deteriorating systems". Em: **European Journal of Operational Research** 139.3 (2002), pp. 469–489. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(01)00197-7.

[8] Quanjiang Yu e Ann-Brith Strömberg. **Mathematical optimization models for long-term maintenance scheduling of wind power systems**. 2021. DOI: https://doi.org/10.48550/arXiv.2105.06666. arXiv: 2105.06666 [math.OC].

[9] Chuntian Zhang et al. "Integrated optimization of train scheduling and maintenance planning on high-speed railway corridors". Em: **Omega** 87.C (2019), pp. 86–104. DOI: 10.1016/j.omega.2018.08.0.

