Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas/ Escola de Minas
Programa de Pós-Graduação em Engenharia de Produção

MÉTODOS META-HEURÍSTICOS PARA A RESOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA DE LONGO PRAZO

Arthur Almeida Santos

João Monlevade-MG, Janeiro de 2023

Arthur Almeida Santos

MÉTODOS META-HEURÍSTICOS PARA A RESOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA DE LONGO PRAZO

Defesa de Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em em Engenharia de Produção da UFOP (Linha de Pesquisa: Modelagem de Sistemas Produtivos e Logísticos), como parte dos requisitos necessários para a obtenção do Título de Mestre em Engenharia de Produção.

Universidade Federal de Ouro Preto – UFOP

Instituto de Ciências Exatas e Aplicadas/ Escola de Minas

Programa de Pós-Graduação em Engenharia de Produção

Orientador: Alexandre Xavier Martins

Coorientador: Marcone Jamilson Freitas Souza

João Monlevade-MG

Janeiro de 2023

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S237m Santos, Arthur Almeida.
Métodos meta-heurísticos para a resolução do problema de sequenciamento de ordens de manutenção preventiva de longo prazo. [manuscrito] / Arthur Almeida Santos. Paganini Barcellos de Oliveira. Thiago Ferreira de Noronha. - 2023.
49 f.

Orientador: Prof. Dr. Alexandre Xavier Martins.
Coorientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Dissertação (Mestrado Acadêmico). Universidade Federal de Ouro Preto. Departamento de Engenharia de Produção. Programa de Pós-Graduação em Engenharia de Produção.

1. Manutenção produtiva total. 2. Greedy Randomized Adaptive Search Procedure (GRASP). 3. Iterated Local Search (ILS). 4. Algorítmos computacionais - Simulated Annealing. I. Noronha, Thiago Ferreira de. II. Oliveira, Paganini Barcellos de. III. Martins, Alexandre Xavier. IV. Souza, Marcone Jamilson Freitas. V. Universidade Federal de Ouro Preto. VI. Título.

CDU 658.5

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

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE ENGENHARIA DE PRODUCAO - ICEA

FOLHA DE APROVAÇÃO

Arthur Almeida Santos

Métodos Meta-heurísticos para a resolução do Problema de Sequenciamento de Ordens de Manutenção Preventiva de Longo Prazo

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Mestre em Engenharia de Produção.

Aprovada em 26 de janeiro de 2023

Membros da banca

Doutor Alexandre Xavier Martins - Orientador (Universidade Federal de Ouro Preto)
Doutor Paganini Barcellos de Oliveira (Universidade Federal de Ouro Preto)
Doutor Thiago Ferreira de Noronha (Universidade Federal de Minas Gerais)

Alexandre Xavier Martins, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 28/02/2023.

Documento assinado eletronicamente por **Alexandre Xavier Martins**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 09/11/2023, às 17:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0622107** e o código CRC **F2EC867A**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0817 - www.ufop.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. O presente trabalho tem como objetivo desenvolver algoritmos meta-heurísticos eficientes para tratar o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP). O trabalho se inicia com o desenvolvimento de uma heurística construtiva e de alocação, seguido do desenvolvimento de heurísticas de busca local e de meta-heurísticas, bem como a avaliação do desempenho dos algoritmos propostos frente à outros trabalhos da 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-chaves: Planejamento de manutenção de longo prazo, GRASP, *Simulated Annealing*, *Iterated Local Search*.

Abstract

The success of a company requires the proper functioning and reliability of its systems with machines and equipment in good condition. For this, a good preventive maintenance plan is essential, which tends to become more complex as the number of equipment and the planning horizon increases. The present work aims to develop efficient meta-heuristic algorithms to deal with the Long-Term Preventive Maintenance Scheduling Problem (PPOMPLP). The work begins with the development of a constructive and a allocation heuristic, followed by the development of local search heuristics and meta-heuristics, as well as the evaluation of the performance of the proposed algorithms compared to other works in the literature. To calibrate and validate the meta-heuristics, small fictitious instances were solved. After calibration, the meta-heuristics were applied to solve larger and real instances. As a result, the Iterated Local Search (ILS) was the best performing meta-heuristic and the result obtained for the real instance was 40.5% better than that presented in the literature.

Keywords: Long-term maintenance scheduling, GRASP, Simulated Annealing, Iterated Local Search.

Lista de ilustrações

Lista de tabelas

Sumário

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; NAJJAR-BASHI; SHEIKHALISHAHI, 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; XING; DAI, 2021). Normalmente, essas atividades de manutenção preventiva necessitam de uma equipe especializada e são executadas em um equipamento específico. 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. Apesar da programação da manutenção preventiva ser um tema amplamente abordado, alguns estudos (LEE; CHA, 2016; WANG; MIAO, 2021) trabalham com modelos para previsão das falhas, e não com modelos focados na programação das ordens de manutenção preventiva.

Outros trabalhos (HEDJAZI, 2015; ALFARES, 2022), tratam de modelos para a resolução de problemas de programação de ordens de manutenção. Hedjazi (2015) trabalha com um problema de programação de ordens de manutenção em máquinas em paralelo não relacionadas. Os operadores, dependendo da habilidade, realizam a manutenção em um tempo diferente. As ordens podem ser alocadas com atraso e o objetivo de minimizar a média ponderada dos tempos de atraso. Alfares (2022) propõe um problema de sequenciamento de ordens de manutenção preventiva em máquinas em paralelo durante uma parada de máquinas para a manutenção. Neste caso a programação é feita de forma a encurtar o período de parada.

No entanto, esses trabalhos não tratam a programação de ordens de manutenção envolvendo o dimensionamento das equipes, as janelas de atendimento das ordens, a habilidade das equipes em executar essas ordens, a disponibilidade dos equipamentos usados para realizar essas ordens e o longo prazo de programação.

De nosso conhecimento, o único trabalho encontrado na literatura que trata dessas

características é o de Aquino, Chagas e Souza (2019). Nomeado por esses autores de PPOM-PLP, um acrônimo para Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo, os autores propuseram uma formulação de programação matemática (*Mixed Integer Linear Programming* - MILP), assim como algoritmos meta-heurísticos baseados em *Simulated Annealing* (SA), *Variable Neighborhood Search* (VNS), *Multi-Start Variable Neighborhood Search* (MSVNS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Biased Random-Key Memetic Algorithm* (BRKMA) para tratar instâncias maiores do problema.

A justificativa para aplicar métodos heurísticos ao PPOMPLP é que este problema é NP-difícil. Um problema de sequenciamento de ordens de manutenção em uma única máquina é NP-difícil (QI; CHEN; TU, 1999). O PPOMPLP trata de sequenciamento em máquinas paralelas, que é mais complexo que os problemas de máquina única e, portanto, pode ser considerado NP-difícil também.

A formulação de programação linear inteira mista proposta por Aquino, Chagas e Souza (2019) apresenta um problema em um conjunto de restrições, fazendo com que o modelo não garanta que o número de horas alocadas para uma equipe de trabalho seja menor ou igual ao número de horas disponíveis. Além disso, o algoritmo de alocação de ordens de manutenção só considera como critério as penalidades associadas a cada ordem de manutenção, quando, na realidade, existem vários outros critérios que poderiam ser analisados. Por fim, esses autores não implementaram, por exemplo, algoritmos baseados em *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Iterated Local Search* (ILS), os quais têm sido usados com sucesso na resolução de vários outros problemas de otimização combinatória (ERTEM et al., 2022; DELORME; IORI; MENDES, 2021).

Para cobrir essa lacuna, o presente trabalho trata o PPOMPLP aplicado a um cenário real de uma empresa mineradora, assim como em Aquino, Chagas e Souza (2019). 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 tratá-lo, propõe-se inicialmente um método heurístico construtivo, composto por duas etapas: sequenciamento e alocação. No sequenciamento define-se a sequência que serão alocadas as ordens e na etapa de alocação, as ordens são alocadas às respectivas equipes ao longo do período de programação. Propusemos um algoritmo de alocação diferente e mais eficiente do que aquele usado por Aquino, Chagas e Souza (2019).

A partir da solução obtida pela heurística construtiva, são desenvolvidos métodos de busca local para refinar essa solução. Esses métodos de construção e refinamento são, então, incorporados a algoritmos meta-heurísticos como GRASP, SA e ILS.

Para avaliar os algoritmos desenvolvidos, são usadas instâncias reais, assim como

instâncias fictícias baseadas no mesmo contexto dessas instâncias reais. Os resultados obtidos são comparados com o trabalho de Aquino, Chagas e Souza (2019), que até o presente momento, é o único que trata do mesmo problema, porém com abordagens de resolução diferentes.

1.1 Problema de pesquisa

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, em um equipamento alvo.

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 alvo da manutenção. Este plano é inserido no sistema ERP (do inglês *Enterprise Resource Planning*, um sistema de gestão empresarial) utilizado pela empresa; porém, este 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 para cada equipe, conforme a disponibilidade de mão de obra e equipamentos necessários. Este método de planejamento de longo prazo faz com que, em geral, apenas 50% das ordens de manutenção sejam atendidas. O restante das ordens de manutenção é terceirizado, 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.

O problema real envolve, normalmente, mais de 33000 ordens de manutenção preventiva. Uma ordem de manutenção preventiva pode ser, por exemplo, uma troca de pneu de um caminhão utilizado na produção. O pneu deve ser trocado preventivamente após determinado tempo ou desgaste, conforme a determinação do fabricante. Portanto, para a definição do problema, cada ordem terá uma lista de informações estruturadas conforme a Tabela 1.

Tabela 1 – Exemplo de ordem de manutenção.

ID da ordem	Tipo de manutenção	Equipamento	Início da janela	Fim da janela	Tempo de processamento	Penalidade
3	10	200	48	480	7.2	216

Neste exemplo, o ID da ordem é um número inteiro sequencial que a identifica. Tipo de manutenção define qual equipe especializada pode realizar esta manutenção; neste

exemplo, só uma equipe de mecânica de veículos (com a habilidade 10) pode realizá-la. A coluna "Equipamento" define em qual equipamento será realizada a manutenção. O equipamento só pode receber uma manutenção por vez. No caso deste exemplo, o equipamento é o caminhão alvo da troca de pneu. A janela de tempo de execução, com unidade de tempo em horas, define a partir de quando a manutenção pode ser realizada e o tempo limite, a manutenção deve ser concluída até este limite. No caso da troca de pneu, este período é previsto pela empresa de acordo com o histórico de gasto do equipamento. Realizar a troca antes significa um gasto desnecessário, pois o material ainda teria condições de uso. Trocar após o fim da janela de tempo pode resultar em um acidente com o veículo. A coluna "Tempo de processamento" indica o tempo gasto para a troca. Por fim, a penalidade é o valor que será somado à função objetivo caso a manutenção não seja realizada.

O valor da penalidade foi definido como sendo o tempo de processamento multiplicado por um valor correspondente à prioridade da ordem de manutenção. Na empresa em estudo as manutenções são divididas em 4 níveis de prioridade. Para as manutenções mais prioritárias, o valor do fator multiplicador foi 40, seguido por 30, 20 e 10 para as menos prioritárias (AQUINO; CHAGAS; SOUZA, 2019).

Cada ordem de manutenção deve ser executada por apenas uma equipe de trabalho, e cada equipe não pode realizar mais de manutenção em paralelo. Além disso, cada manutenção será realizada em um equipamento, que poderá receber apenas uma manutenção por vez. No problema real são 145 equipes de trabalho e 1032 equipamentos. Assim como as ordens de manutenção, cada equipe terá uma lista de informações, estruturadas conforme a Tabela 2. O ID da equipe é um número inteiro sequencial que identifica cada equipe. O tipo da equipe identifica quais tipos de manutenção a equipe pode executar. Caso uma equipe possa executar uma ordem, este número será igual ao apresentado no campo "tipo de manutenção" da ordem. No caso do exemplo da troca de pneu, o tipo da equipe seria mecânica de veículos. E a disponibilidade da equipe identifica até qual período a equipe está disponível no ano. Neste exemplo a equipe está disponível do momento zero até a hora 4800.

Tabela 2 – Exemplo de equipe de trabalho.

ID da equipe	Tipo de equipe	Disponibilidade
2	10	4800

O objetivo do sequenciamento é minimizar o número de ordens não alocadas e o custo relacionado ao número de equipes. Cada ordem não alocada gera um custo de terceirização para a empresa. Este custo, está apresentado na modelagem do problema, como a penalidade. O custo refente às equipes de produção somam um valor unitário para

cada equipe utilizada. Isto implica que o custo referente ao número de equipes possui peso significativamente menor em relação às ordens não alocadas.

1.2 Objetivos

Este trabalho tem como objetivo geral desenvolver algoritmos meta-heurísticos eficientes para tratar o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo.

Como objetivos específicos, o trabalho busca:

- Propor melhorias na formulação de programação matemática de Aquino, Chagas e Souza (2019);
- Fazer um estudo de relaxação de variáveis na formulação desenvolvida;
- Implementar uma nova heurística construtiva para o PPOMPLP;
- Implementar um novo algoritmo de alocação de ordens de manutenção;
- Comparar o desempenho desse algoritmo de alocação com o de Aquino, Chagas e Souza (2019);
- Implementar algoritmos baseados nas meta-heurísticas GRASP, SA e ILS;
- Comparar e analisar estatisticamente o desempenho dos algoritmos meta-heurísticos desenvolvidos;

1.3 Justificativa

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.

Apenas Aquino, Chagas e Souza (2019) abordaram o problema real alvo deste trabalho, e portanto, o presente trabalho pretende criticar e, buscar obter resultados melhores para o problema apresentado por Aquino, Chagas e Souza (2019), por meio de uma abordagem diferente quanto às heurísticas construtivas e de alocação e a utilização de meta-heurísticas diferentes das já utilizadas.

Além disso, os algoritmos propostos no presente trabalho podem ser aplicados na resolução de problemas semelhantes, do tipo sequenciamento de ordens com máquinas em paralelo não relacionadas. O estudo de Vallada e Ruiz (2011), por exemplo, apresenta um problema deste tipo, porém com objetivo de minimizar *makespan*. Assim como Aquino, Chagas e Souza (2019), Vallada e Ruiz (2011) utilizam algoritmos genéticos para a resolução do problema. O resultado e a comparação dos métodos do presente trabalho com os métodos de Aquino, Chagas e Souza (2019), podem servir de base para discussão de problemas semelhantes.

1.4 Metodologia

A metodologia de pesquisa adotada neste trabalho é baseada na modelagem quantitativa em gestão de produção, em particular, a metodologia em pesquisa operacional. Mais detalhadamente, esta pesquisa pode ser definida como uma pesquisa empírica normativa quantitativa (MIGUEL et al., 2010). Empírica pois os dados foram extraídos de um problema real, apresentados pela empresa abordada neste trabalho. Normativa porque tem como objetivo melhorar resultados presentes na literatura. E quantitativa pois o problema pode ser descrito por meio de variáveis e equações.

Esta pesquisa segue as etapas conforme esquema da Figura 1. Inicia-se pela definição do escopo da pesquisa, delimitado pelos objetivos definidos na Seção 1.2. Em sequência o desenvolvimento da heurística construtiva, que produz uma programação inicial. Com a sequência definida pela heurística construtiva, a heurística de alocação é utilizada para estruturar as atividades de cada equipe ao longo do período de programação. Os resultados desta construtiva foram comparados aos resultados do modelo exato, para sua validação.

Posteriormente, foram desenvolvidos os métodos de busca local *first improvement* e *random descent*. Estes métodos heurísticos são subordinados às meta-heurísticas. Portanto, a etapa seguinte, é 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 das instâncias fictícias. Os parâmetros testados para cada meta-heurística estão na Tabela 3. Na Seção 4.4 será detalhada a função de cada parâmetro apresentado na tabela. A presença do valor "NA" no campo indica que o parâmetro não se aplica à meta-heurística.

Tabela 3 – Parâmetros utilizados no IRACE

	α	Máximo de iterações	T_0	Níveis de perturbação
GRASP	0.05/ 0.1/ 0.2/ 0.9	500/ 2000/ 10000	NA	NA
SA	0.5/ 0.85/ 0.9/ 0.95	500/ 2000/ 10000/ 20000	10/ 25/ 100/ 1000	NA
ILS	NA	500/ 2000/ 10000/ 20000	NA	2/ 3/ 4

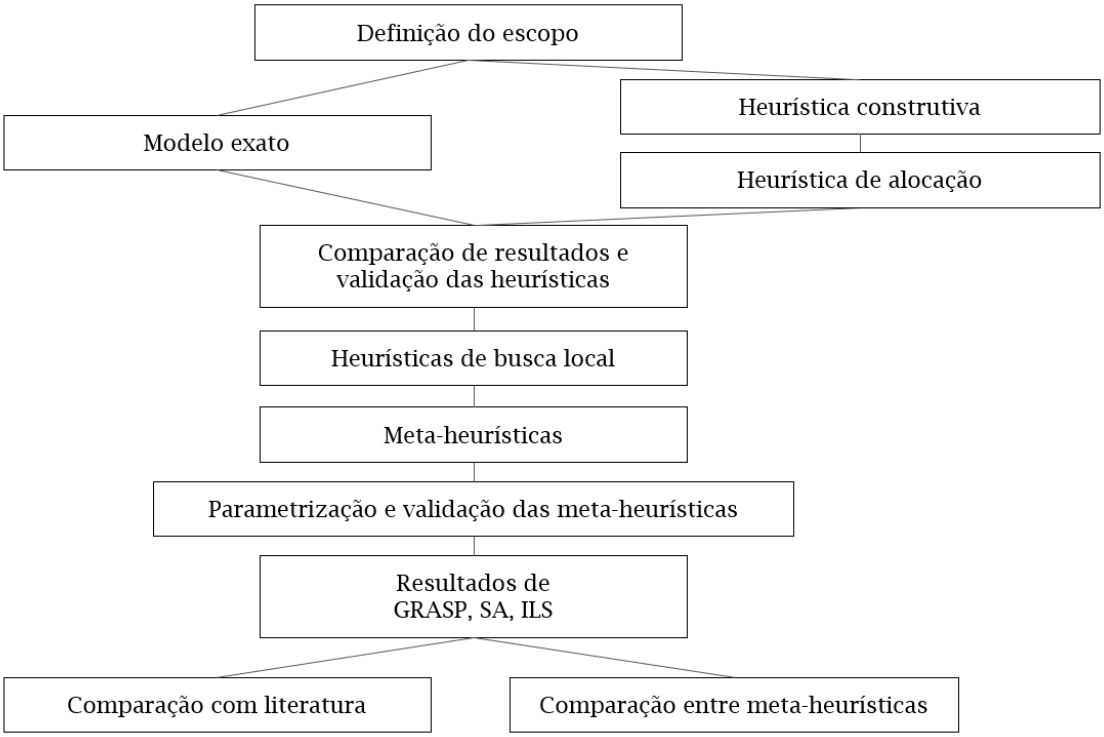


Figura 1 – Esquema da metodologia

Para a validação das meta-heurísticas são resolvidas instâncias fictícias pequenas. Após a calibragem e validação, as meta-heurísticas foram aplicadas na resolução de instâncias maiores e a real. Os resultados obtidos foram comparados entre as meta-heurísticas aplicadas e com os resultados apresentados na literatura.

2 Revisão da Literatura

As técnicas de sequenciamento podem ser aplicadas para a elaboração de planos de manutenção. Problemas de sequenciamento são descritos pela alocação de n atividades a um número m de máquinas. Este tipo de problema pode ser descrito pelo trio $\alpha \mid \beta \mid \gamma$ (PINEDO, 2016). O campo α descreve o ambiente da máquina. O campo β detalha as características de processamento e as restrições. O campo γ contém os objetivos a serem otimizados. Para cada um destes campos, adota-se uma terminologia de acordo com o problema em evidência.

Esta terminologia de Pinedo (2016), foi utilizada por Aquino, Chagas e Souza (2019) para representar uma simplificação do PPOMPLP como um problema $P_m \mid r_j M_j \mid \gamma$. Um ponto de discordância é quanto ao campo γ, uma vez que Aquino, Chagas e Souza (2019) consideram que o objetivo de otimização seria minimizar a mão de obra necessária para executar o maior número de tarefas, 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 segundo 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 com m máquinas paralelas, onde 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 paralelas, nos quais nem todas m máquinas são capazes de processar todas as atividades. O 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, as ordens não alocadas entram como custo para a função objetivo, de forma ponderada, de acordo com sua prioridade.

Problemas de sequenciamento de máquinas em paralelo têm sido estudados há décadas. Alidaee e Rosa (1997) estão entre os primeiros a publicar uma abordagem heurística para resolução deste tipo de problema. Porém os autores tratam apenas do sequenciamento da produção. Problemas de sequenciamento de produção e de manutenção, têm sido tratados de forma diferente tanto na literatura quanto na indústria. Isto, apesar de serem problemas semelhantes e conflitantes, já que a programação de manutenções preventivas interfere na programação de produção e que para a realização da manutenção preventiva, o equipamento deve cessar a produção (AVALOS-ROSALES et al., 2018).

Para a revisão de literatura foram buscados artigos que tratavam de programação

de manutenções preventivas em máquinas em paralelo. Além disso os artigos deveriam conter uma abordagem heurística para resolução do problema. Grande parte dos artigos encontrados tratam apenas de programação de produção, não tratando do tema programação de manutenções, apesar do termo "*maintenance scheduling*" ser inserido no filtro de busca dos bancos de dados. Estes artigos que não abordavam programação de manutenções foram excluídos da revisão.

Dos artigos que continham o tema programação de manutenções, há dois tipos: programação apenas das manutenções e programação de produção em conjunto com as manutenções. Os dois tipos foram considerados. Além disso, ao buscar por artigos que abordem problemas de máquinas em paralelo, parte dos trabalho trata de um subtipo, que são as máquinas em paralelo não relacionadas, que também foram incluídos na busca.

Hedjazi (2015) aborda um problema de programação de ordens de manutenção em máquinas em paralelo não relacionadas com objetivo de minimizar a média ponderada dos tempos de atraso e número de tarefas em atraso. Para resolução do problema, foi desenvolvido uma heurística em duas etapas, que são semelhantes à heurística construtiva deste trabalho. O método heurístico de Hedjazi (2015) obtém resultados melhores que os métodos antes aplicados por Marmier, Varnier e Zerhouni (2009) para o mesmo problema.

Ruiz-Torres, Paletta e M'Hallah (2016) abordam um problema de programação de produção de máquinas idênticas em paralelo, com máquinas sujeitas a desgaste e manutenções preventivas. O problema tem como objetivo minimizar o *makespan*. Os autores apresentam um modelo de programação inteira mista e três heurísticas construtivas diferentes para resolução do problema. Nenhum método de busca local ou meta-heurístico foi aplicado, porém o autor sugere que estes métodos heurísticos construtivos possam ser futuramente aprimorados desta forma.

Fakher, Nourelfath e Gendreau (2016) tratam de um problema de programação de produção em máquinas em paralelo, considerando efeito de desgaste nas máquinas. As ordens de manutenção preventiva são inseridas entre as ordens de produção para impedir o processo de desgaste. O objetivo é minimizar os custos de produção, manutenção e qualidade. Para resolução do problema, o autor propõe um modelo de programação não-linear e dois métodos meta-heurísticos: Algoritmo Genético e *Tabu Search*. O método *Tabu Search* desenvolvido apresentou melhor desempenho para este problema.

Upasani et al. (2017) abordam um problema de sequenciamento de manutenções preventivas em máquinas idênticas em paralelo, com objetivo de minimizar os custos de operação, que incluem tempo de máquina parada, custo da manutenção entre outros. Um método de força bruta foi utilizado para encontrar o resultado ótimo das instâncias pequenas e calibrar os demais algoritmos. Para resolução das instâncias maiores foram propostos um Algoritmo Memético e um *Particle Swarm Optimization* (PSO). O método de força bruta apresenta melhores resultados para as instâncias menores, porém não é

possível concluir o processamento em instâncias maiores, sendo o Algoritmo Memético melhor neste caso.

Avalos-Rosales et al. (2018) trabalharam com a inclusão de ordens de manutenção preventiva em uma programação de produção. É um problema com máquinas em paralelo não relacionadas, com objetivo de minimizar *makespan* e tempos de configuração dependentes. O autor utiliza um modelo matemático para resolução das instâncias pequenas e médias e desenvolve um algoritmo *Multi-Start* para a resolução das instâncias grandes (com até 250 ordens). O autor aplica este método *Multi-Start* em um problema semelhante e obtém melhores resultados que o Algoritmo Genético de Vallada e Ruiz (2011).

Fu et al. (2019) trabalham com um problema de sequenciamento de produção com manutenção preventiva dos moldes das máquinas. É um problema de sequenciamento de máquinas idênticas em paralelo, com objetivo de minimizar o *makespan*. Como método de resolução os autores propõem um algoritmo híbrido de *Three-level Particle Swarm Optimization* (TLPSO) e *Variable Neighbourhood Search* (VNS). O algoritmo proposto se mostra superior a PSO e VNS puros, já apresentados anteriormente na literatura.

Liu et al. (2021) abordam um problema de programação de produção com inclusão de ordens de manutenção preventiva entre as ordens de produção. É um problema com máquinas em paralelo sincronizadas, com objetivo de minimizar o custo de produção. Um tipo de algoritmo genético (*Cooperative Co-evolutionary Genetic Algorithm* - CCGA) é utilizado para resolução do problema. O algoritmo apresenta melhores resultados que um algoritmo genético tradicional.

Alfares, Mohammed e Ghaleb (2021) trabalharam com um problema de programação de produção em duas máquinas idênticas em paralelo, considerando efeito de desgaste nas máquinas. As ordens de manutenção preventiva são inseridas entre as ordens de produção para impedir o processo de desgaste, apresentando semelhança ao problema de Fakher, Nourelfath e Gendreau (2016). O objetivo é minimizar o *makespan*, que neste caso é o tempo para concluir todas as ordens em ambas as máquinas. Para resolução do problema foi desenvolvido um modelo de programação linear inteira, que obtém resultado ótimo. Para resolver instâncias maiores foram desenvolvidos seis algoritmos heurísticos. O método meta-heurístico *Variable Neighborhood Search* (VNS) obteve os melhores resultados para o problema considerando as manutenções.

Delorme, Iori e Mendes (2021) abordam um problema de sequenciamento de ordens de produção e manutenções preventivas em maquinas em paralelo não relacionadas. O objetivo é minimizar o *makespan*. Os autores utilizam o MILP proposto por Ruiz-Torres, Paletta e M'Hallah (2016) e propõe quatro novos modelos matemáticos a partir de diferentes pontos de vista do problema. Os autores propõem também um algoritmo meta-heurístico ILS para resolução do problema. Dentre os modelos exatos, uma formulação baseada em fluxo de arco obteve melhor desempenho. O método ILS teve resultados competitivos para

as instâncias grandes do problema.

Lu et al. (2021) trabalham com um problema de sequenciamento de ordens de produção em conjunto com ordens de manutenção preventiva. O problema é apresentado como um sequenciamento de bateladas em paralelo, porém, a partir da descrição do problema é possível concluir se tratar também de um problema de máquinas em paralelo. O trabalho tem como objetivo minimizar o *makespan* por meio de um algoritmo metaheurístico híbrido de *Variable Neighborhood Search* e um algoritmo populacional chamado *Discrete Black Hole*. O algoritmo híbrido apresenta resultados melhores que as metaheurísticas não híbridas já utilizadas na literatura.

Alfares (2022) estuda um problema de sequenciamento de ordens de manutenção preventiva em máquinas em paralelo, com restrição de recursos. O objetivo é minimizar o *makespan*, que neste caso, é o tempo de desligamento das máquinas para a realização do grupo de manutenções. Para resolução do problema foi desenvolvido um modelo de programação linear inteira, que obtém resultado ótimo. Porém, devido ao grau de dificuldade do problema, foi desenvolvido uma heurística em duas etapas, que propicia uma resolução mais rápida do problema. As duas etapas são semelhantes à heurística construtiva deste trabalho e de Hedjazi (2015). A primeira etapa com o sequenciamento das ordens e a segunda com a alocação das ordens.

Ertem et al. (2022) abordam um problema de sequenciamento de ordens de manutenção preventiva com máquinas em paralelo. O objetivo é minimizar o *makespan*, que neste caso é equivalente a minimizar o número de tarefas em atraso. Como método de resolução são propostos dois modelos MILP para resolução das instâncias pequenas e calibragem das heurísticas. Dois métodos heurísticos construtivos foram propostos, sendo o primeiro dependente de uma solução inicial gerado pelo método exato e o segundo utiliza um método de exploração das vizinhanças semelhante ao GRASP. Apesar das heurísticas não atingirem valores ótimos em todas instâncias pequenas, como o MILP, apresentam melhor desempenho nas instâncias grandes. A segunda heurística apresenta resultados melhores que a primeira.

A Tabela 4 resume os principais pontos apresentados nos artigos em discussão. A Tabela contém as colunas “Autores”, “Máquinas”, “Tipos de programação”, “Objetivo a minimizar” e “Abordagem”. “Máquinas” identifica a configuração de máquinas abordada no trabalho, sendo variações de problemas de máquinas paralelas. “Tipos de programação” aponta se é um problema de programação de produção, programação de manutenção ou uma combinação das duas. “Objetivo a minimizar” identifica quais os indicadores de desempenho foram utilizados na função objetivo. “Abordagem” indica quais abordagens heurísticas foram aplicadas para resolução do problema.

Pelo volume de trabalhos encontrados, pôde-se observar que este tema ainda é pouco explorado. E, com o aprimoramento do poder computacional, abordagens meta-

Tabela 4 – Revisão de literatura.

Autores	**Máquinas**	**Tipos de programação**	**Objetivo a minimizar**	**Abordagem**
HEDJAZI, 2015	paralelo não relacionadas	manutenção	média ponderada dos tempos de atraso	heurística construtiva
FAKHER; NOURELFATH; GENDREAU, 2016	paralelo	produção e manutenção	custos	algoritmo genético e *tabu search*
RUIZ-TORRES; PALETTA; M'HALLAH, 2016	idênticas em paralelo	produção e manutenção	*makespan*	heurísticas construtivas
UPASANI et al., 2017	idênticas em paralelo	manutenção	custos	algoritmo memético e *particle swarm optimization*
AVALOS-ROSALES et al., 2018	paralelo não relacionadas	produção e manutenção	*makespan*	*multi-start*
FU et al., 2019	idênticas em paralelo	produção e manutenção	*makespan*	*three-level particle swarm optimization e variable neighbourhood search*
LIU et al., 2021	paralelo sincronizadas	produção e manutenção	custos	*cooperative co-evolutionary genetic algorithm*
ALFARES; MOHAMMED; GHALEB, 2021	idênticas em paralelo	produção e manutenção	*makespan*	*variable neighborhood search*
DELORME; IORI; MENDES, 2021	paralelo não relacionadas	produção e manutenção	*makespan*	*iterated local search*
LU et al., 2021	paralelo	produção e manutenção	*makespan*	*variable neighborhood search e discrete black hole*
ALFARES, 2022	paralelo	manutenção	*makespan*	heurística construtiva
ERTEM et al., 2022	paralelo	manutenção	*makespan*	GRASP

heurísticas passaram a ser mais exploradas, como GRASP, SA, ILS que serão abordadas neste trabalho.

2.1 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 (FESTA; RESENDE, 2002). No método *multi-start* original (LIN; KERNIGHAN, 1973), a solução construtiva é totalmente aleatória, enquanto que o método GRASP desenvolvido por Feo e Resende (1995) utiliza como heurística construtiva um método parcialmente guloso.

Esta heurística construtiva parcialmente gulosa, recebe um parâmetro de aleatoriedade α, sendo $0 \leq \alpha \leq 1$ (FEO; RESENDE, 1989). Quanto mais próximo de zero, mais aleatória se torna a heurística construtiva e quanto mais próxima de um, mais gulosa. Para uma construção mais aleatória, maior a variância e menor a qualidade das soluções. Para uma construção mais gulosa, menor a variância e melhor a qualidade das soluções. Deve se buscar um valor α que equilibre a qualidade e a variância das soluções.

Rabadi, Moraga e Al-Salem (2006) aplicaram o algoritmo que foi desenvolvido por Feo e Resende (1995) para a resolução de um problema de sequenciamento de máquinas em paralelo não relacionadas. Ertem et al. (2022), conforme citado anteriormente, trabalhou com este método para um problema de sequenciamento de ordens de manutenção preventiva em máquinas em paralelo.

2.2 *Simulated Annealing* (SA)

O conceito de SA foi introduzido por Metropolis et al. (1953), porém, o método foi aplicado pela primeira vez para resolução de problemas de otimização por Kirkpatrick, Gelatt e Vecchi (1983). O método se baseia no processo de recozimento de metais. O termo *annealing* (recozimento, em português) se refere a um processo físico em que um sólido, geralmente um metal, é submetido a uma alta temperatura fazendo com que suas partículas se rearranjem no estado líquido. Em seguida, o material é resfriado lentamente de forma que suas partículas se acomodem em uma posição de menor energia interna. Este estado final, resulta em um sólido mais forte e estável (LAARHOVEN; AARTS, 1987).

De forma análoga a este processo físico, o algoritmo se inicia em um estado aquecido, ou seja, parâmetro de temperatura inicial alta. Neste estado aquecido, o algoritmo produz e aceita soluções mais diversas para o problema. Estas soluções diversas não necessariamente apresentam qualidade. Conforme as iterações acontecem, o valor de temperatura do algoritmo diminui, fazendo com que as soluções produzidas sejam menos diversas, convergindo assim para um ótimo local.

Este método é amplamente utilizado para problemas de sequenciamento de máquinas em paralelo, inclusive problemas com máquinas não relacionadas (KOULAMAS, 1997; SANTOS et al., 2019). Abdeljaoued, Saadani e Bahroun (2020) trabalham com o SA para um problema de sequenciamento em máquinas em paralelo.

2.3 *Iterated Local Search* (ILS)

Outra abordagem de Santos et al. (2019) para um problema de sequenciamento de máquinas em paralelo com máquinas não relacionadas, é o ILS. Os métodos heurísticos de busca local, tendem a encontrar um ótimo local dentro do conjunto de soluções, a partir de uma solução inicial. Para encontrar novas soluções e preferencialmente o ótimo global, é necessário que o método de busca local seja aplicado a partir de outras soluções iniciais.

O método mais simples seria realizar a busca local a partir de pontos de partida completamente aleatórios. Porém este método tende a gerar soluções muito aleatórias, principalmente para instâncias muito grandes, não se aproximando do ótimo global conforme seria desejado. Para evitar as desvantagens de pontos de partida completamente aleatórios, o ILS é utilizado (LOURENÇO; MARTIN; STÜTZLE, 2001). Para gerar pontos de partida diferentes, o ILS aplica níveis de perturbação à solução anterior.

A Figura 2 facilita a visualização de uma iteração do ILS. A partir de uma solução inicial s o primeiro método de busca local encontra a solução ótima local $s*$. Uma perturbação é aplicada a $s*$ gerando a nova solução inicial s'. A partir de s' é realizada uma nova busca local encontrando um novo ótimo local $s*'$, que é aceito somente se for

melhor solução que $s*$. A perturbação deve ser forte o suficiente para que o algoritmo não retorne sempre para uma mesma solução $s*$, tomando o cuidado para que não seja tão forte de forma que acabe gerando novas soluções completamente aleatórias (LOURENÇO; MARTIN; STÜTZLE, 2001).

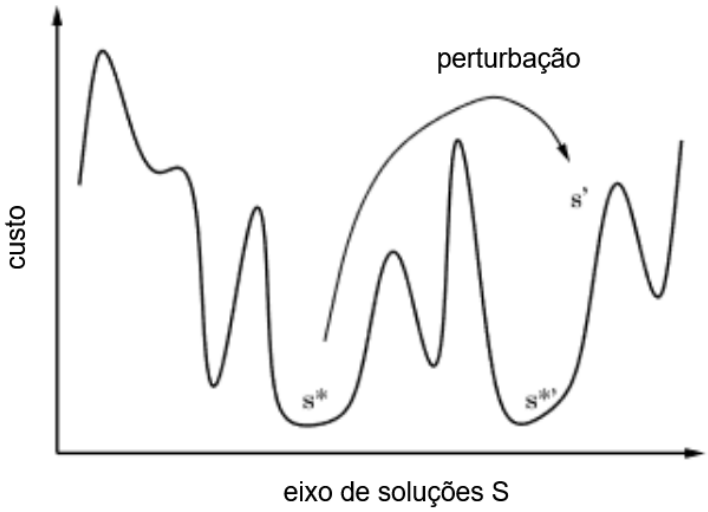


Figura 2 – Funcionamento do método ILS
Fonte: Lourenço, Martin e Stützle (2001), p. 2

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. O ILS e o *Tabu Search* foram utilizados para resolução de um problema de programação de produção com máquinas em paralelo, obtendo bons resultados para ambas meta-heurísticas (XU et al., 2019; DELORME; IORI; MENDES, 2021).

3 Formulação matemática

O modelo matemático utilizado foi o desenvolvido por Aquino, Chagas e Souza (2019). Para a descrição matemática do modelo foram utilizadas as seguintes notações. $\mathcal{T} = \{1, 2, ..., n\}$ é o conjunto de n atividades de manutenção que devem ser realizadas pelo conjunto $\mathcal{W} = \{1, 2, ..., m\}$ 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 em que a manutenção será realizada, e a 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 $k \in \mathcal{W}$. Para o modelo de programação linear inteira mista (MILP), foram definidas as variáveis de decisão do modelo:

- $x_{ij}^k = 1$, se a manutenção i precede imediatamente a manutenção j executada pela equipe de trabalho k; 0, caso contrário;
- $y_{ik} = 1$, se a manutenção i será executada pela equipe de trabalho k; 0, caso contrário;
- $z_k = 1$, se a equipe de trabalho k é utilizada; 0, caso contrário;
- $c_{ik} \geq 0$, tempo de conclusão da manutenção i pela equipe de trabalho k;
- $r_{ij} = 1$, se a manutenção i precede imediatamente a manutenção j para o caso em que ambas as manutenções se referem ao mesmo equipamento ($A_i = A_j$) e que ambas são executadas por equipes diferentes; 0, caso contrário.

Definidas as variáveis de decisão, é possível partir para a modelagem da função objetivo (Equação 3.1), que busca minimizar a soma do custo de mão de obra com as manutenções não atendidas:

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

Sujeito às restrições:

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

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

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

$$\sum_{i \in \mathcal{T}_k \cup \{0\} \setminus \{t\}} x_{it}^k = \sum_{j \in \mathcal{T}_k \cup \{0\} \setminus \{t\}} x_{tj}^k \qquad \forall k \in \mathcal{W}, t \in \mathcal{T}_k \tag{3.5}$$

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

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

$$c_{ik} \geq (e_i + p_i) \cdot y_{ik} \qquad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \tag{3.8}$$

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

$$c_{jk'} \leq c_{ik} - p_i + M''_{ij} \cdot r_{ij} \qquad \forall k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_k, k \neq k', i < j, A_i = A_j \tag{3.10}$$

$$c_{jk'} \leq c_{ik} + p_j - M'_{ij} \cdot (1 - r_{ij}) \qquad \forall k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_k, k \neq k', i < j, A_i = A_j \tag{3.11}$$

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

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

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

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

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

$$r_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{T}, j \in \mathcal{T} \mid i < j, A_i = A_j \tag{3.17}$$

O conjunto de restrições (3.2) garante que cada manutenção será executada por no máximo uma equipe de trabalho. O conjunto de restrições (3.3) garante que toda manutenção executada terá uma mão de obra alocada. As restrições (3.4) garantem que uma determinada equipe de trabalho só será alocada se ela for utilizada para execução de alguma manutenção. As restrições (3.5) garantem o sequenciamento das ordens de manutenção executadas por cada equipe de trabalho. O conjunto de restrições (3.6) asseguram que o tempo de conclusão de uma manutenção fictícia, criada para facilitar a modelagem, é nulo para todas as equipes de trabalho. As restrições (3.7, 3.10 e 3.11) garantem que a equipe de trabalho não pode executar mais de uma ordem de manutenção simultaneamente e cada equipamento não pode ter mais de uma ordem de manutenção sendo executada simultaneamente. Mais especificamente, as restrições (3.7) são ativadas apenas quando a manutenção j for realizada imediatamente após a manutenção i pela equipe k. Por sua vez, as restrições (3.10) são ativadas quando a manutenção j precede imediatamente a manutenção i para o caso em que ambas as manutenções se referem ao mesmo equipamento ($A_i = A_j$) e executadas por equipes diferentes. As restrições (3.11) são similares às (3.10) quando a manutenção i precede imediatamente a manutenção j. Para ativar ou desativar essas restrições, as constantes M'_{ij} e M''_{ij} devem assumir valores maiores ou iguais a $l_i + p_j$ e $l_j + p_i$, respectivamente. As restrições (3.8 e 3.9) garantem que cada manutenção é executada dentro de sua janela de tempo, enquanto as restrições (3.12) garantem que o número de horas alocadas para uma equipe de trabalho seja menor ou igual ao número de horas disponíveis. Por fim, as restrições (3.13-3.17) indicam o domínio e escopo das variáveis de decisão.

Importante destacar que este modelo matemático necessita da criação de uma manutenção fictícia para seu funcionamento. Os valores que serão inseridos nos parâmetros desta manutenção fictícia não importam, mas como boa prática, são utilizados valores incomuns às demais manutenções. Ao fazer a leitura dos dados, esta manutenção será o primeiro elemento ($i = 0$) e as demais manutenções são enumeradas na sequência a partir dos índices $i = \{1, 2, ..., n\}$.

3.1 Análise crítica do modelo

Ao realizar uma análise crítica do modelo desenvolvido por Aquino, Chagas e Souza (2019), foi possível apontar alguns erros no modelo. No modelo programado em C++ havia uma restrição não descrita no trabalho publicado. A restrição seria conforme indicado na restrições (3.18). Estas restrições garantem que as equipes com menor índice tenham prioridade na alocação das ordens.

$$\sum_{j \in \mathcal{T}_k} (j \cdot x_{0j}^{k}) \geq \sum_{j \in \mathcal{T}_k} (j \cdot x_{0j}^{k'}) \qquad \forall k \in \mathcal{W}, k' \in \mathcal{W}, k < k' \tag{3.18}$$

Outro problema diz respeito às restrições (3.12), presentes no trabalho de Aquino, Chagas e Souza (2019). Tais restrições deveriam estar ativas somente se a equipe k estiver executando a atividade i. Portanto há necessidade de utilização da variável de decisão y, e a forma correta da restrição seria conforme indicado nas restrições (3.19).

$$c_{ik} - M \cdot (1 - y_{ik}) \leq h_k \qquad \forall k \in \mathcal{W}; i \in \mathcal{T}_k \tag{3.19}$$

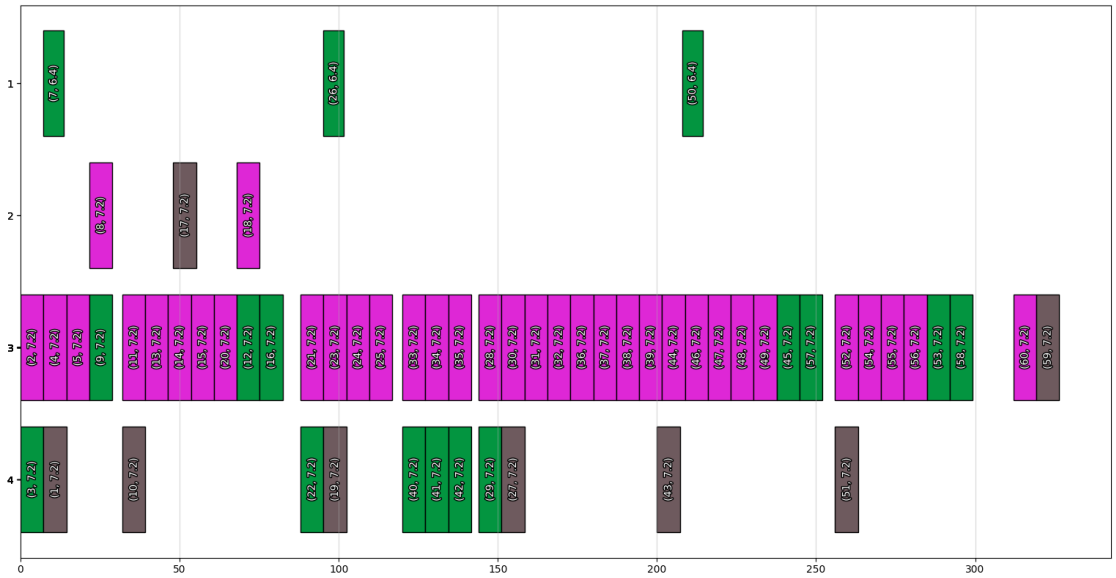
Para a comparação das soluções obtidas por este trabalho e o de Aquino, Chagas e Souza (2019), foi desenvolvido um algoritmo para a apresentação das soluções em um gráfico de Gantt. Para o desenvolvimento deste algoritmo foi utilizada a biblioteca Python matplotlib. A Figura 3 apresenta um exemplo de dois gráficos de Gantt gerados a partir de uma das instâncias fictícias, com 60 ordens. A Figura 3(a) é o gráfico de Gantt gerado a partir de uma solução obtida por meio do ILS deste trabalho e a Figura 3(b) é o gráfico de Gantt gerado a partir de uma solução obtida por meio do modelo exato. Cada linha representa uma equipe de trabalho, cada cor representa um equipamento e em cada ordem está em texto o ID da ordem e o tempo de processamento. O eixo x indica o tempo da programação.

No exemplo da Figura 3, por se tratar de uma instância pequena, ambos métodos de resolução atingem o valor ótimo, apesar da forma que os dois métodos agrupam as ordens serem diferentes. É possível notar que a solução do ILS prioriza o máximo de ocupação para uma equipe, antes de alocar as ordens restantes em uma equipe que compartilhe o tipo. Na imagem é possível notar isto com ocupação da “Equipe 3"em detrimento da “Equipe 4".

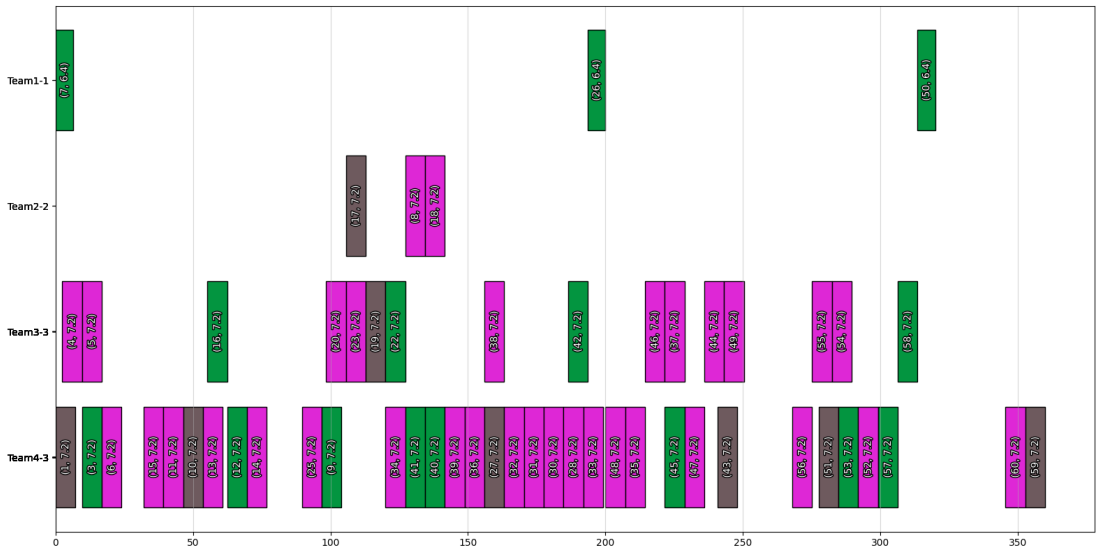
3.2 Testes com o modelo e relaxações

Ao longo da testagem do modelo exato, foram realizadas rodadas teste considerando a relaxação de cada uma das variáveis de decisão do modelo. Para a comparação do processamento relaxado com os resultados do processamento não relaxado, foram utilizadas apenas as instâncias que tiveram o processamento concluído com menos de uma hora. Isto garante que a comparação seja feita apenas entre resultados ótimos. Os resultados obtidos estão apresentados na Tabela 5.

Na Tabela 5 o *gap* representa a diferença entre o resultado obtido por meio da relaxação e o resultado não relaxado. Quanto maior o *gap*, mais distantes são as soluções obtidas por meio da relaxação.



(a) Exemplo de gráfico de Gantt do ILS



(b) Exemplo de gráfico de Gantt do modelo exato

Figura 3 – Comparação entre gráficos resultantes do ILS e do modelo exato

A relaxação da variável x resulta em soluções inviáveis, sem sequer um sequenciamento lógico. Relaxar a variável r produz soluções que não respeitam a restrição de que um equipamento não possa ser utilizado por mais de uma equipe ao mesmo tempo. Relaxar as variáveis z e y resulta em soluções viáveis, porém, na média não há ganho no tempo de processamento, nem nos resultados obtidos.

Tabela 5 – Testes com variáveis relaxadas

Instância				Solução não	Solução com variável relaxada							
ID	n	m	Q	relaxada	x	Gap	r	Gap	y	Gap	z	Gap
1	20	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
2	20	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
3	20	4	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
4	20	5	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
5	20	10	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
6	30	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
7	30	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
8	30	4	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
11	40	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
12	40	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
16	60	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
17	60	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
26	20	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%
27	20	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
28	20	5	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
29	20	6	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
31	30	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%
32	30	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
33	30	5	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
36	40	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%
37	40	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
42	60	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
51	20	3	4	723	0	100,00%	723	0,00%	723	0,00%	723	0,00%
52	20	4	4	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
76	20	4	5	724	0	100,00%	724	0,00%	724	0,00%	724	0,00%
77	20	5	5	221	219	0,90%	5	97,74%	221	0,00%	221	0,00%

4 Métodos heurísticos

4.1 Heurísticas construtivas

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 equipamento". 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 l_c e maior w_c tem prioridade de alocação. O algoritmo retorna a sequência de ordens N.

Algoritmo 1: Construção do sequenciamento simples das ordens

```
   Entrada: Lista de ordens de manutenção
   Saída: N
 1 início
 2 |  N ← ∅;
 3 |  C ← Inicializa conjunto de ordens candidatas;
 4 |  enquanto C ≠ ∅ faça
 5 |  |  melhor_tempo ← ∞;
 6 |  |  melhor_penalidade ← −1;
 7 |  |  para c ∈ C faça
 8 |  |  |  se (l_c < melhor_tempo) ou (l_c = melhor_tempo e
   |  |  |   w_c > melhor_penalidade) então
 9 |  |  |  |  c* ← c;
10 |  |  |  |  melhor_tempo ← l_c;
11 |  |  |  |  melhor_penalidade ← w_c;
12 |  |  |  fim
13 |  |  fim
14 |  |  N ← N ∪ {c*};
15 |  |  C ← C \ {c*};
16 |  fim
17 fim
```

O segundo algoritmo construtivo, realiza primeiro uma avaliação de qual é a equipamento-alvo de maior número de manutenções. O sequenciamento das ordens realizadas no equipamento mais utilizado é definido primeiro, seguindo os mesmos critérios do Algoritmo 1. Ordens com menor l_c e maior w_c tem prioridade de alocação. Após sequenciar todas as ordens do equipamento mais utilizado, é realizado o sequenciamento do segundo mais utilizado, até que o sequenciamento de todos seja concluído.

Para exemplificar, uma instância fictícia foi apresentada no Quadro 4.1. Para a

instância em questão a sequência resultado do Algoritmo 1 seria $N = \{1,4,6,2,5,3\}$. O primeiro elemento desta sequência é a ordem 1 porque tem se o tempo limite da ordem (l_c) menor, com o valor 4. O segundo elemento (ordem 4) é o segundo da sequência por ter o segundo menor l_c. Para o terceiro elemento há empate do valor de l_c entre as ordens 2 e 6. Como a ordem 6 tem maior penalidade, esta ocupa a posição, seguido da ordem 2 que será o quarto elemento. Os dois últimos elementos da sequência são posicionados de acordo com os respectivos valores de l_c, já que não há empate.

Para o segundo algoritmo construtivo, como o equipamento carro é o mais utilizado, a sequência se iniciaria por ele. Isto resultaria na seguinte sequência: $N = \{1,5,3,6,2,4\}$. Para a construção desta sequência, avalia-se para as primeira posições apenas ordens executadas no equipamento carro (ordens 1, 3 e 5). Portanto, o primeiro elemento da sequência é a ordem 1, com menor l_c, seguido da ordem 5, com segundo menor l_c e por último a ordem 3. O segundo equipamento mais utilizado é o caminhão (ordens 2 e 6). As ordens 2 e 6 têm l_c igual, porém a penalidade da 6 é maior, fazendo com que a ordem 6 seja o quarto elemento da sequência, seguida pela ordem 2. O último equipamento é a moto com apenas a ordem 4, que é então o último elemento da sequência.

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 4.1 – 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, Chagas e Souza (2019)

4.2 Heurística de alocação

Com a sequência das ordens definida, é necessário realizar a sua alocação. A alocação é realizada por equipes, iniciando pela equipe, ou grupo de equipes, mais ocupada. A Figura 4 complementa o exemplo apresentado no Quadro 4.1, 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 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) Especialidade das equipes

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

(b) Ocupação das equipes

Figura 4 – Cálculo de qual grupo de equipes está mais ocupado

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 no mesmo equipamento em outra equipe no mesmo horário. As variáveis *inicio* e fim utilizadas no Algoritmo 2 são os horários de inicio 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 5 exemplifica o funcionamento da alocação de ordens. O primeiro gráfico de Gantt representa a alocação das ordens do Quadro 4.1 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: $N' = \{1,6,2,3\}$. A janela para alocação da ordem 1 tem os valores [0, 4]. Como não há nenhuma ordem alocada no instante 0, a ordem 1 é alocada neste instante. A ordem 6 tem janela [4, 7] e pode ser alocada no instante 4. A ordem 2 tem janela [2, 7] e pode ser alocada no instante 2. A ordem 3 tem janela [3, 9], porém o instante 3 está ocupado pela ordem 3. O instante 4 está ocupado pela ordem 6. Portanto a melhor alocação para a ordem 3 é no instante 5.

Após a alocação de todas ordens das equipes de mecânica, a alocação é realizada para as equipes de elétrica, seguindo o restante da sequência $N'' = \{4,5\}$. A ordem 4 tem janela [2, 6] e pode ser alocada no instante 2. A ordem 5 tem janela [3, 8] e pode ser alocada no instante 3.

O mesmo procedimento de alocação é realizado para a "construção do sequenciamento das ordens por equipamento" (segundo Gantt), sendo apenas a sequência diferente.

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

Algoritmo 2: Aloca ordens

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

início
  $s \leftarrow \emptyset$;
  Organiza equipes, começando pelas equipes mais ocupadas;
  para $k \in W$ **faça**
    $C \leftarrow$ Inicializa conjunto de ordens candidatas, seguindo sequência N, apenas com ordens que a equipe é capaz de realizar e com ordens $\notin$ s;
    enquanto $C \neq \emptyset$ *e* $fim_c \leq h_k$ **faça**
      $c \leftarrow C[0]$;
      $inicio_c \leftarrow e_c$;
      $fim_c \leftarrow inicio_c + p_c$;
      enquanto *Houver conflito de horários entre c e s[i] e* $fim_c \leq l_c$ *e* $fim_c \leq h_k$ **faça**
        para $i \in s$ **faça**
          se *Ordem s[i] alocada na mesma equipe e mesmo horário* **então**
            $inicio_c \leftarrow s[i][fim_c]$;
            $fim_c \leftarrow inicio_c + p_c$;
          fim
          se *Ordem s[i] no mesmo equipamento, alocada em outra equipe e mesmo horário* **então**
            $inicio_c \leftarrow s[i][fim_c]$;
            $fim_c \leftarrow inicio_c + p_c$;
          fim
        fim
      fim
      se $fim_c > l_c$ *ou* $fim_c > h_k$ **então**
        $C \leftarrow C \setminus c$;
      senão
        Insere c na solução s, com horário de início e término definidos por $inicio_c$ e fim_c respectivamente;
        $C \leftarrow C \setminus c$;
      fim
    fim
  fim
  Calcula penalidade e ordens não utilizadas;
fim

gerada dentro dos métodos de busca local, é refeita apenas a alocação dentro de períodos em que hajam ordens não alocadas, mantendo a solução anterior (s') para os demais períodos, o que resultou na implementação do Algoritmo 3.

Em relação ao Algoritmo 2, três procedimentos diferentes foram adotados no Algoritmo 3:

Sequência Simples: N = {1,4,6,2,5,3} Sequência Mecânica: N' = {1,6,2,3} Sequência Elétrica: N'' = {4,5}

Equipes\período	0	1	2	3	4	5	6	7	8	9
Mecânica A	1		2	2	6	3	3	3		
Mecânica B										
Elétrica			4	5	5					

Ferramentas mais usadas: Carro > Caminhão > Moto
Sequência Equipamentos: N = {1,5,3,6,2,4} Sequência Mecânica: N' = {1,3,6,2} Sequência Elétrica: N'' = {5,4}

Equipes\período	0	1	2	3	4	5	6	7	8	9
Mecânica A	1			3	3	3	6	2	2	
Mecânica B										
Elétrica			4				5	5		

Legenda dos Equipamentos: Carro Caminhão Moto

Figura 5 – Exemplo de alocação de ordens

- Primeiro retira-se as ordens impossíveis de ser alocadas da lista de ordens não alocadas. A existência deste tipo de ordem será melhor explicado na seção de resultados. Como são impossíveis de se alocar, não é necessário desperdiçar iterações tendo estas como candidatas.

- Segundo sorteia-se uma ordem aleatória (x) dentre as não alocadas. A janela de tempo $[e_x, l_x]$ desta ordem será o único período que a programação será refeita. O restante da nova solução s é mantida da solução anterior s'.

- E por último é a possibilidade de embaralhar a ordem de alocação dentre as equipes. O Algoritmo 2 sempre começa a alocar pela equipe mais ocupada. Geralmente isto produz soluções melhores. Porém, em alguns casos, a mudança da ordem das equipes produz soluções melhores. Então foi definido que em 20% das iterações da busca local a ordem das equipes é alterada.

4.3 Busca local

Como métodos de busca local foram implementados dois: o *random descent* e o *first improvement*. O *random descent* desenvolvido realiza trocas aleatórias entre duas ordens (i, j) da sequência dada N. Só é avaliada a troca de i e j se 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, aceita-se a troca, modificando N.

O método *first improvement* realiza a troca de 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

Algoritmo 3: Realocação de ordens

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

início
  Retira ordens impossíveis de alocar da lista de ordens não utilizadas;
  $x \leftarrow escolha\_aleatória(ordens\_não\_utilizadas)$;
  para $y \in s'$ **faça**
    se $fim_y < e_x$ *ou* $inicio_y > l_x$ **então**
      $s \leftarrow y$
    fim
  fim
  20% de chance de embaralhar a ordem das equipes;
  para $k \in W$ **faça**
    $C \leftarrow$ Inicializa conjunto de ordens candidatas, seguindo sequência N, apenas com ordens que a equipe é capaz de realizar e com ordens $\notin$ s;
    enquanto $C \neq \emptyset$ *e* $fim_c \leq h_k$ **faça**
      $c \leftarrow C[0]$;
      $inicio_c \leftarrow e_c$;
      $fim_c \leftarrow inicio_c + p_c$;
      enquanto *Houver conflito de horários entre c e s[i] e* $fim_c \leq l_c$ *e* $fim_c \leq h_k$ **faça**
        para $i \in s$ **faça**
          se *Ordem s[i] alocada na mesma equipe e mesmo horário* **então**
            $inicio_c \leftarrow s[i][fim_c]$;
            $fim_c \leftarrow inicio_c + p_c$;
          fim
          se *Ordem s[i] no mesmo equipamento, alocada em outra equipe e mesmo horário* **então**
            $inicio_c \leftarrow s[i][fim_c]$;
            $fim_c \leftarrow inicio_c + p_c$;
          fim
        fim
      fim
      se $fim_c > l_c$ *ou* $fim_c > h_k$ **então**
        $C \leftarrow C \setminus c$;
      senão
        Insere c na solução s, com horário de início e término definidos por $inicio_c$ e fim_c respectivamente;
        $C \leftarrow C \setminus c$;
      fim
    fim
  fim
  Calcula penalidade e ordens não utilizadas;
fim

troca representa melhora no resultado da função objetivo, a troca é realizada, atualizando

N, e reiniciando o método.

4.4 Meta-heurísticas

O Algoritmo 4 representa a implementação do modelo meta-heurístico GRASP. Os parâmetros desta e das demais meta-heurísticas foram obtidos por meio de calibração utilizando o pacote IRACE, sendo $\alpha = 0.1$ e máximo de iterações de 2000. O único parâmetro fixo e comum à todas 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 4 foi desenvolvido com base no trabalho de Feo e Resende (1995), utilizando como método de busca local o *random descent*.

Algoritmo 4: GRASP

Entrada: Função de avaliação f(.), lista de ordens de manutenção, lista de equipes, controlador de aleatoriedade α, tempo limite;
Saída: s*, penalidade*;

```
1  início
2      f(s*) ← ∞;
3      enquanto tempo limite não ultrapassado faça
4          s' ← ContrucaoParcGulosa(α);
5          s ← RandomDescent(s');
6          se f(s) < f(s*) então
7              s* ← s;
8          fim
9      fim
10 fim
```

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 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. Mesmo a modificação sendo feita apenas em parte da programação, o custo da função objetivo continua sendo do horizonte de programação completo.

O Algoritmo 5 é o SA, desenvolvido com base no trabalho de Kirkpatrick, Gelatt e Vecchi (1983). O vizinho aleatório é gerado trocando dois indivíduos aleatórios (i, j) da sequência de programação. Para limitar o número de trocas, i e j só são trocados se estiverem dentro do mesmo período de programação. Ordens em períodos de tempos diferentes têm pouca influência na alocação da outra. Os parâmetros utilizados foram $\alpha = 0.9$, $T_0 = 25$ e máximo de iterações de 10000.

Algoritmo 5: Simulated Annealing

Entrada: Função de avaliação f(.), lista de ordens de manutenção, lista de equipes, taxa de resfriamento α, máximo de iterações it_{max}, temperatura inicial T_0, tempo limite;
Saída: s*, penalidade*;

início
  $s* \leftarrow MelhorConstrutiva()$;
  $iterT \leftarrow 0$;
  $T \leftarrow T_0$;
  enquanto $T > 0.001$ *e tempo limite não ultrapassado* **faça**
    enquanto $iterT < it_{max}$ **faça**
      $iterT \leftarrow iterT + 1$;
      Gere um vizinho aleatório s';
      $\Delta = f(s') - f(s)$;
      se $\Delta < 0$ **então**
        $s \leftarrow s'$;
        se $f(s') < f(s*)$ **então**
          $s* \leftarrow s$;
        fim
      fim
      se $\Delta \geq 0$ **então**
        Gere aleatoriamente $x \in [0, 1]$;
        se $x < e^{(-\Delta/T)}$ **então**
          $s \leftarrow s'$;
        fim
      fim
    fim
    $T \leftarrow T * \alpha$;
    $iterT \leftarrow 0$;
  fim
fim

O Algoritmo 6 é o ILS, que foi adaptado do proposto por Lourenço, Martin e Stützle (2019). Como métodos de busca local foram utilizados o *random descent* e o *first improvement*, sendo sorteado, com igual probabilidade, qual será utilizado em cada iteração. Os parâmetros utilizados foram $p = 3$ e máximo de iterações de 10000.

Algoritmo 6: ILS

Entrada: Função de avaliação f(.), lista de ordens de manutenção, lista de equipes, tempo limite, nível da perturbação p;
Saída: s*, penalidade*;

```
1  início
2      s_0 ← MelhorConstrutiva();
3      s* ← BuscaLocal(s_0);
4      enquanto tempo limite não ultrapassado faça
5          s' ← Perturbacao(p);
6          s'' ← BuscaLocal(s');
7          se penalidade(s'') < penalidade(s*) então
8              s* ← s'';
9          fim
10     fim
11 fim
```

5 Resultados e Discussão

O capítulo de resultados será apresentado da seguinte forma: primeiro a apresentação das configurações e sistemas utilizados no processamento. Em seguida, uma seção com a comparação entre os três algoritmos desenvolvidos neste trabalho. Uma comparação será realizada em relação a todas as instâncias e outra agrupada por tamanho. Posteriormente, serão verificadas o desvio-padrão médio dos algoritmos e se há diferença estatística entre eles. Na seção seguinte, os algoritmos desenvolvidos serão comparados com os disponíveis na literatura. Nesta seção, a comparação envolve também, a comparação do resultado global, resultados agrupados por tamanho de instância, desvio-padrão médio e diferença estatística entre os algoritmos.

Os processamentos foram realizados em um Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 15GB de memória RAM sob o sistema operacional Ubuntu 16.04.6 LTS (Xenial Xerus). O computador pertence ao Laboratório de Simulação e Otimização de Sistemas (LASOS) da Universidade Federal de Ouro Preto (UFOP). Vale destacar que o presente trabalho não utiliza o processamento em *threads* paralelas. Em contraponto, o trabalho de Aquino, Chagas e Souza (2019) faz utilização do processamento em paralelo, utilizando uma máquina com 39 *threads* paralelas, resultando em ganho significativo de tempo de processamento. As heurísticas e meta-heurísticas foram todas desenvolvidas em Python, versão 3.9. Cada método meta-heurístico foi parametrizado com o tempo de processamento igual a n segundos, sendo n o número de ordens de manutenção da instância e com cinco repetições para cada instância.

Para análise dos resultados, as instâncias foram divididas em grupos, conforme realizado também por Aquino, Chagas e Souza (2019):

- P: instâncias de 20 a 80 ordens de manutenção;
- M: instâncias de 150 a 600 ordens de manutenção;
- G: instâncias de 1200 a 4800 ordens de manutenção;
- GG: instâncias de 9600 a 33484 ordens de manutenção;

5.1 Comparativo entre GRASP, SA, ILS

Os resultados obtidos na comparação entre as meta-heurísticas implementadas com os resultados obtidos para as instâncias tamanho P encontram-se no Apêndice A. Já a comparação entre as meta-heurísticas implementadas com os resultados obtidos para as

instâncias tamanho M, G, GG encontram-se no Apêndice B. As colunas “n”, “m” e “Q” são respectivamente o número de ordens de manutenção, número de equipes e número de equipamentos em cada instância. A coluna “Média” mostra o valor de média das repetições para cada instância. A coluna “Melhor” apresenta o menor valor obtido dentre as repetições para cada instância. A coluna “Desvio” indica o desvio-padrão das repetições para cada instância.

A Tabela 6 apresenta o desvio-padrão médio do *gap* das instâncias. O *gap* foi calculado pela diferença percentual entre o melhor valor obtido pelo algoritmo e o melhor valor disponível na literatura. O ILS apresenta um desvio-padrão médio menor dentre as meta-heurísticas aplicadas, o que indica que esta é a implementação com a melhor repetibilidade dentre as três. Na Tabela 7 os desvios-padrão são apresentados agrupados por tamanho de instância. Para os tamanhos P, M e GG o ILS ainda é o algoritmo com melhor repetibilidade, sendo o SA melhor neste quesito apenas para instâncias GG.

Tabela 6 – Desvio-padrão médio das meta-heurísticas GRASP, SA, ILS

	GRASP	SA	ILS
Todas instâncias	10,6%	7,6%	7,1%

Tabela 7 – Desvio-padrão médio por grupos das meta-heurísticas GRASP, SA, ILS

	GRASP	SA	ILS
P	10,8%	8,3%	7,8%
M	0,9%	4,5%	0,9%
G	14,0%	6,5%	6,1%
GG	1,6%	0,1%	0,9%

A Tabela 8 apresenta as instâncias agrupadas por tamanho para uma análise detalhada. A coluna “Médias” mostra a média dos valores médios das repetições obtidos para cada tamanho de instância. A coluna “Melhores” apresenta a média dos menores valores obtido dentre todas repetições para cada tamanho de instância. A coluna “contagem melhores” indica em quantas instâncias de um mesmo tamanho cada meta-heurística obteve o melhor resultado.

De acordo com os dados da Tabela 8, o ILS obteve resultados melhores em 99% das instâncias P, 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 ligeiramente melhor entre as instâncias M e G. Para as instâncias M, G e GG o ILS também foi a meta-heurística que obteve melhores resultados dentre as três, sendo melhor em 71% das instâncias M, G e GG. SA obteve o melhor resultado em 50% e o GRASP em 36,8%. Estes números somados são maiores que 100%, porque pode haver empate entre as meta-heurísticas em uma ou mais instâncias. O melhor resultado para a instância real foi obtido pelo GRASP.

Tabela 8 – Comparativo entre as meta-heurísticas GRASP, SA, ILS

	Número de ordens	**Número de instâncias**	**GRASP**			**SA**			**ILS**		
			Médias	Melhores	Contagem melhores	Médias	Melhores	Contagem melhores	Médias	Melhores	Contagem melhores
	20	20	**299**	**299**	**20**	**299**	**299**	**20**	**299**	**299**	**20**
	30	20	**299**	**299**	**20**	**299**	**299**	18	**299**	**299**	**20**
P	40	20	332	331	19	339	332	14	**330**	**324**	**20**
	60	20	504	493	18	502	486	12	**482**	**475**	**19**
	80	20	1.345	1.315	9	1.345	1.304	7	**1.290**	**1.282**	**20**
	150	6	891	**889**	**6**	**889**	**889**	5	**889**	**889**	**6**
M	300	6	1.387	1.371	2	**1.371**	**1.370**	4	1.377	**1.370**	**5**
	600	6	3.903	3.815	1	3.867	3.803	2	**3.826**	**3.798**	**5**
	1.200	6	9.311	9.002	1	9.047	8.916	3	**8.801**	**8.669**	**4**
G	2.400	6	18.148	17.770	1	17.543	**17.025**	3	**17.531**	17.107	**4**
	4.800	6	38.389	**36.166**	**3**	38.420	37.810	1	**38.305**	37.692	2
	9.600	1	31.664	30.451	0	30.199	30.009	0	**30.011**	**30.002**	**1**
GG	19.200	1	74.257	70.963	0	**69.200**	**68.711**	**1**	69.672	68.720	0
	33.484	1	134.610	**128.856**	**1**	**133.291**	129.082	0	134.507	130.906	0

Para verificar se as meta-heurísticas possuem uma diferença estatisticamente significativa entre si, foi aplicado o teste de Friedman. O teste Friedman (FRIEDMAN, 1940) pode ser utilizado para analisar a presença de diferenças estatisticamente relevantes entre três ou mais modelos de dados ordinais. Caso o *p-valor* resultante do teste seja maior que o nível de significância, usualmente estabelecido em 0,05, a hipótese nula de que as amostras são estatisticamente diferentes é descartada. Neste caso, pode-se aplicar um teste *post-hoc* como o teste Nemenyi para verificar entre quais pares de medidas há diferença significativa (SHELDON; FILLYAW; THOMPSON, 1996).

Ambos testes utilizam como parâmetro de desempenho a classificação da meta-heurística em cada aplicação. Desta forma, são definidas classificações (*ranks*) para cada heurística em cada instância. O teste de Friedman considera que se o *rank* médio de um modelo é aproximadamente igual a outro, então os modelos seriam estatisticamente iguais. Já o teste de Nemenyi considera que o *rank* médio de uma meta-heurística deve apresentar no mínimo uma certa distância em valor, definida como diferença crítica, do *rank* médio de outra meta-heurística para serem consideradas diferentes (WANG et al., 2018).

Os testes de Friedman junto ao teste *post-hoc* Nemenyi tem sido utilizados para a comparação de resultados de meta-heurísticas (MA et al., 2022). Como vantagem, ao considerar apenas os *ranks*, os testes se tornam menos suscetíveis a *outliers* (BROWN; MUES, 2012). Porém, ao utilizarem medidas diferentes, os testes podem apresentar resultados diferentes. Além disso, ao utilizarem apenas o *rank*, a dimensão de diferença entre as meta-heurísticas não é considerada na análise. Desta forma, a média e desvio padrão também serão utilizadas neste estudo como base para a comparação estatística entre as meta-heurísticas.

Aplicando o teste de Friedman, com nível de significância de 0,05, para as três meta-heurísticas (GRASP, SA e ILS), considerando todas as instâncias, obtém-se um *p-valor* de $2,23 * 10^{-5}$, indicando que há uma diferença estatística significativa entre as três implementações. Para verificar quais pares são diferentes entre si, foi aplicado um

teste *post-hoc* de Nemenyi. Este teste retornou os *p-valores* conforme a Tabela 9, indicando que apenas GRASP e ILS são diferentes estatisticamente entre si.

Tabela 9 – Teste *post-hoc* entre as meta-heurísticas GRASP, SA, ILS para todas instâncias

Algoritmo 1	Algoritmo 2	*p-valor*
GRASP	SA	0,9573
GRASP	ILS	0,0307
SA	ILS	0,0633

Dependendo do tamanho da instância, os três algoritmos produzem resultados melhores ou piores em relação aos outros. Portanto, para uma análise estatística mais detalhada, os mesmos testes foram aplicados aos três algoritmos em cada tamanho de instância. Estes resultados estão apresentados na Tabela 10. A coluna "Friedman" *p-valor* apresenta o resultado obtido pela aplicação do teste de Friedman com nível de significância de 0,05. Apenas para o tamanho GG as três meta-heurísticas não possuem diferença estatística em seus resultados. Por não possuir diferença, não haveria necessidade de um teste *post-hoc*. Além disso, as instâncias GG apresentam uma amostra pequena (três instâncias) para que o teste seja confiável, sendo recomendado pelo menos uma amostra maior do que cinco. Para os demais tamanhos de instâncias, o teste de Friedman indica diferença estatística entre os algoritmos, por isso há necessidade do teste *post-hoc* Nemenyi, que indica entre quais pares há diferença.

Tabela 10 – Teste *post-hoc* entre as meta-heurísticas GRASP, SA, ILS por grupos de instâncias

	Friedman	Teste Nemenyi		
	p-valor	Algoritmo 1	Algoritmo 2	*p-valor*
P	$2 * 10^{-8}$	GRASP	SA	0,3712
		GRASP	ILS	0,2653
		SA	ILS	0,0105
M	0,025961	GRASP	SA	0,6298
		GRASP	ILS	0,0935
		SA	ILS	0,4733
G	0,029508	GRASP	SA	0,1590
		GRASP	ILS	0,0416
		SA	ILS	0,8291
GG	0,716531	NA		

Na Tabela 10 na coluna "Teste Nemenyi" os *p-valores* obtidos apontam diferença estatística entre SA e ILS para tamanho P e entre GRASP e ILS para tamanho G de instância. Para o tamanho M, o teste de Nemenyi não apontou diferença entre nenhum par. Neste caso, a comparação será feita pelos valores de média dentre os pares. Para o tamanho M, GRASP obteve média de 2024,8, SA média de 2020,6 e ILS média de 2018,8.

Como GRASP e ILS estão mais distantes entre si, estes são os dois algoritmos diferentes estatisticamente.

5.2 Comparativo com a literatura

Ao iniciar a comparação das meta-heurísticas com a literatura, foi observado que mesmo rodando o algoritmo por tempos longos, alguns resultados não eram alcançáveis. Após uma análise detalhada dos resultados obtidos por Aquino, Chagas e Souza (2019), 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, ultrapassam a janela de tempo [e_c, l_c].

Em instâncias que possuam estas ordens não alocáveis, não é possível obter valores da função objetivo menores que a soma das penalidades. 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. Apesar destas restrições apontadas, Aquino, Chagas e Souza (2019) apresenta valores da função objetivo menores que a soma das penalidades destas ordens não alocáveis. Portanto, é possível concluir que seu algoritmo produz soluções inviáveis. Desta forma, foram levantadas quais instâncias possuíam soluções inviáveis e estas não foram utilizadas nas comparações por grupos, porém seus resultados estão apresentados tachados no Apêndice C. Ao total, foram dez instâncias tamanho P, três tamanho M e três tamanho G nesta situação.

O comparativo entre GRASP, SA e ILS por instância é apresentado no Apêndice C. Na coluna "Melhor valor conhecido" está o melhor obtido para a instância dentre todos os métodos desenvolvidos por Aquino, Chagas e Souza (2019), inclusive o MILP, e todos os métodos desenvolvidos no presente trabalho. A coluna "Objetivo" apresenta o melhor valor obtido pelo *Biased Random-Key Memetic Algorithm* (BRKMA) de Aquino, Chagas e Souza (2019) e pelos métodos desenvolvidos neste trabalho. A coluna "*Gap*" apresenta a diferença percentual entre o melhor valor obtido pelo método e o melhor valor conhecido para a instância.

Para realizar a comparação com a literatura, foi escolhido o melhor algoritmo dentre os três implementados neste trabalho. O algoritmo escolhido foi o ILS, que obtém melhores

resultados de forma geral, conforme evidenciado na Tabela 8. No trabalho de Aquino, Chagas e Souza (2019) algoritmos diferentes também obtiveram diferentes performances dependendo do tamanho da instância. Para fins de comparação, foi escolhido o BRKMA, porque é o algoritmo que obteve melhores resultados nas instâncias maiores e na real.

Conforme apresentado na Tabela 11, no conjunto de todas instâncias, o ILS apresenta resultados melhores em 14,63% das instâncias, piores em 10,57% e iguais em 74,8%. Para as instâncias pequenas, os resultados são 4,44% piores, o restante são resultados iguais. Porém, para instâncias M, G e GG, o número de instâncias com melhores resultados ultrapassa os piores, chegando a 100% de resultados melhores para as instâncias GG.

Tabela 11 – Comparativo entre ILS e literatura por resultados individuais

Resultados	**Todas instâncias**	**P**	**M**	**G**	**GG**
Piores	10,57%	4,44%	9,09%	20,00%	0,00%
Melhores	14,63%	0,00%	18,18%	30,00%	100,00%
Iguais	74,80%	95,56%	72,73%	50,00%	0,00%

De forma semelhante à análise realizada entre as meta-heurísticas, as instâncias foram agrupadas por tamanho e apresentadas na Tabela 12. A coluna "ILS" mostra a média dos menores valores obtidos pelo ILS para cada tamanho de instância. A coluna "BRKMA Aquino" apresenta a média dos menores valores obtidos pelo BRKMA de Aquino, Chagas e Souza (2019) para cada tamanho de instância. A coluna "*Gap*" apresenta a diferença entre a média do ILS e do BRKMA. A coluna "*Gap* médio" indica a média dos *gaps*.

Tabela 12 – Comparativo com a literatura entre grupos de instâncias

	Número de ordens	**Número de instâncias**	**ILS**	**BRKMA Aquino**	***Gap***	***Gap* médio**
P	20	20	299	299	0,0%	**1,4%**
	30	20	299	299	0,0%	
	40	20	324,2	313,4	3,4%	
	60	20	475,4	464,6	2,3%	
	80	10	586,5	579,3	1,2%	
M	150	5	696,4	696,4	0,0%	**-1,0%**
	300	5	1.246	1.288,2	-3,3%	
	600	5	3.631	3.619,2	0,3%	
G	1.200	5	8.191,4	7.960,8	2,9%	**-16,3%**
	2.400	5	15.191,2	15.089	0,7%	
	4.800	5	33.924,8	71.455,6	-52,5%	
GG	9.600	1	30.002	38.004	-21,1%	**-31,8%**
	19.200	1	68.720	103.863	-33,8%	
	33.484	1	130.906	220.048	-40,5%	

Para verificar se há diferença estatística entre os resultados obtidos neste trabalho e os resultados de Aquino, Chagas e Souza (2019), foram realizados os mesmo testes estatísticos aplicados na seção 5.2. Em um cenário de comparação envolvendo todas as instâncias, não havia diferença estatística entre o GRASP e o ILS deste trabalho. Então foi realizado o teste de Friedman envolvendo GRASP, ILS e o BRKMA de Aquino, Chagas e Souza (2019). O *p-valor* obtido foi de 0,0158, o que indica que há diferença estatística entre os três. Ao realizar o teste *post-hoc* de Nemenyi, o teste não apontou em qual par está esta diferença. Ao analisar a média dos valores temos: GRASP com média 4681,6, ILS com média 4698,0 e BRKMA com média 7282,7. Há considerável diferença entre as médias de BRKMA/GRASP e BRKMA/ILS, portanto, pode haver diferença estatisticamente significativa entre estes dois pares.

Como o tamanho das instâncias influencia o desempenho dos algoritmos implementados neste trabalho, é necessário detalhar a análise estatística por grupos de instâncias. Na Tabela 13 estão apresentados os resultados do teste de Friedman entre os algortimos deste trabalho e o BRKMA. Ressalta-se que, para o tamanho P, o BRKMA foi comparado apenas com o GRASP e o ILS, deixando de lado o SA. Isto porque a melhor média para este tamanho foi do ILS, e o SA é diferente estatisticamente do ILS, sendo assim, um algoritmo pior para este caso. Por isso, o SA foi retirado da comparação para o tamanho P. A mesma justificativa se estende para os demais tamanhos, sendo desconsiderados os algoritmos estatisticamente diferentes e piores na média.

Tabela 13 – Teste de Friedman comparativo com a literatura para grupos de instâncias

	Algoritmos comparados	*p-valor*
P	GRASP, ILS, BRKMA	0,0006
M	SA, ILS, BRKMA	0,3067
G	GRASP, ILS, BRKMA	0,3438
GG	GRASP, SA, ILS, BRKMA	0,1218

Os *p-valores* obtidos na Tabela 13 indicam que apenas nas instâncias tamanho P há diferença estatística entre GRASP, ILS e BRKMA. O teste *post-hoc* de Nemenyi não aponta qual par apresenta a diferença, portanto, foi realizada a análise da média dos valores. GRASP apresenta média 386,2, ILS média 375,7 e BRKMA média 370,1. Como a maior diferença entre as médias envolve o BRKMA e o GRASP, conclui-se estar entre estes dois a diferença estatística.

Conclusões

Limitações do trabalho

Na calibragem realizada no pacote IRACE, foram utilizadas somente instâncias fictícias pequenas, devido ao elevado tempo de processamento que seria necessário para rodar os testes com instâncias de todos os tamanhos. Isto faz com que os valores encontrados para os parâmetros utilizados, sejam os melhores apenas para as instâncias pequenas, podendo prejudicar os resultados obtidos em instâncias maiores. Além disso, há um parâmetro que realiza o embaralhamento das equipes no algoritmo de realocação, que teve seu valor definido em 20%, realizando apenas testes preliminares. Este parâmetro poderia ser também incluído na calibragem pelo IRACE.

Conclusão

Neste trabalho foram implementadas heurísticas construtivas e meta-heurísticas para gerar soluções competitivas para o PPOMPLP, visando a redução de manutenções não realizadas e de custo com equipes. Os resultados obtidos pelos três métodos meta-heurísticos foram comparados entre si e com a literatura disponível. Dentre os algoritmos implementados neste trabalho, o ILS é o que apresentou melhor repetibilidade dos resultados e soluções melhores de forma geral, obtendo o melhor resultado em 99% das instâncias tamanho P e em 71% das instâncias tamanho M, G e GG. Apesar disso, o melhor resultado obtido para a instância real foi alcançado pelo GRASP. Estatisticamente, apenas GRASP e ILS são diferentes entre si, quando os algoritmos são analisados em relação a todas as instâncias.

Na comparação dos resultados obtidos pelo ILS e o BRKMA de Aquino, Chagas e Souza (2019), os resultados do ILS foram melhores em 14,63% das instâncias, iguais para 74,8% das instâncias e piores para 10,57%, quando comparados em relação à todas as instâncias. Quando a comparação é feita de acordo com o tamanho das instâncias o ILS apresenta pior desempenho nas instâncias tamanho P e melhor desempenho para as instâncias tamanho M, G e GG. Para a instância real, o valor obtido pelo ILS é 40,5% melhor que o valor obtido pelo BRKMA.

O desvio-padrão médio dos algoritmos de Aquino, Chagas e Souza (2019) é menor que dos algoritmos implementados neste trabalho, o que indica pior repetibilidade destes em relação à literatura. GRASP e ILS foram comparados com o BRKMA e os dois pares são diferentes estatisticamente, sendo a média dos valores de GRASP e ILS consideravelmente

menores que do BRKMA.

A comparação dos algoritmos desenvolvidos neste trabalho com a literatura tem como limitação a utilização de linguagens de programação diferentes, o processamento em *threads* paralelas, realizado por Aquino, Chagas e Souza (2019) e não neste trabalho e a utilização de máquinas diferentes para o processamento.

Para trabalhos futuros, sugere-se considerar a programação das ordens de manutenção não atendidas, inserindo-as na programação com atraso, ao invés de simplesmente não alocá-las. Isso geraria um novo problema com um objetivo de otimização diferente.

Referências

ABDELJAOUED, M. A.; SAADANI, N. E. H.; BAHROUN, Z. Heuristic and metaheuristic approaches for parallel machine scheduling under resource constraints. *Operational Research*, Springer, v. 20, n. 4, p. 2109–2132, 2020.

ALFARES, H.; MOHAMMED, A.; GHALEB, M. Two-machine scheduling with aging effects and variable maintenance activities. *Computers & Industrial Engineering*, Elsevier BV, v. 160, p. 107586, out. 2021.

ALFARES, H. K. Plant shutdown maintenance workforce team assignment and job scheduling. *Journal of Scheduling*, Springer Science and Business Media LLC, v. 25, n. 3, p. 321–338, abr. 2022.

ALIDAEE, B.; ROSA, D. Scheduling parallel machines to minimize total weighted and unweighted tardiness. *Computers & Operations Research*, Elsevier, v. 24, n. 8, p. 775–788, 1997.

AQUINO, R. D.; CHAGAS, J. B. C.; SOUZA, M. J. F. Abordagem exata e heurísticas para o problema de planejamento de ordens de manutenção de longo prazo: Um estudo de caso industrial de larga escala. *Pesquisa Operacional para o Desenvolvimento*, v. 11, n. 3, p. 159–182, 2019.

AVALOS-ROSALES, O. et al. Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times. *Computers & Industrial Engineering*, Elsevier BV, v. 123, p. 364–377, set. 2018.

BROWN, I.; MUES, C. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, Elsevier BV, v. 39, n. 3, p. 3446–3453, fev. 2012.

DELORME, M.; IORI, M.; MENDES, N. F. Solution methods for scheduling problems with sequence-dependent deterioration and maintenance events. *European Journal of Operational Research*, Elsevier BV, v. 295, n. 3, p. 823–837, dez. 2021.

EBRAHIMIPOUR, V.; NAJJARBASHI, A.; SHEIKHALISHAHI, M. Multi-objective modeling for preventive maintenance scheduling in a multiple production line. *Journal of Intelligent Manufacturing*, Springer, v. 26, n. 1, p. 111–122, 2015.

ERTEM, M. et al. Workers-constrained shutdown maintenance scheduling with skills flexibility: Models and solution algorithms. *Computers & Industrial Engineering*, Elsevier, p. 108575, 2022.

FAKHER, H. B.; NOURELFATH, M.; GENDREAU, M. A cost minimisation model for joint production and maintenance planning under quality constraints. *International Journal of Production Research*, Informa UK Limited, v. 55, n. 8, p. 2163–2176, jun. 2016.

FEO, T. A.; RESENDE, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, Elsevier, v. 8, n. 2, p. 67–71, 1989.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, Springer, v. 6, n. 2, p. 109–133, 1995.

FESTA, P.; RESENDE, M. G. Grasp: An annotated bibliography. In: *Essays and surveys in metaheuristics*. [S.l.]: Springer, 2002. p. 325–367.

FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, JSTOR, v. 11, n. 1, p. 86–92, 1940.

FU, X. et al. A three-level particle swarm optimization with variable neighbourhood search algorithm for the production scheduling problem with mould maintenance. *Swarm and Evolutionary Computation*, Elsevier, v. 50, p. 100572, 2019.

HEDJAZI, D. Scheduling a maintenance activity under skills constraints to minimize total weighted tardiness and late tasks. *International Journal of Industrial Engineering Computations*, Growing Science, v. 6, n. 2, p. 135–144, 2015.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

KOULAMAS, C. Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Naval Research Logistics (NRL)*, Wiley Online Library, v. 44, n. 1, p. 109–125, 1997.

LAARHOVEN, P. J. V.; AARTS, E. H. Simulated annealing. In: *Simulated annealing: Theory and applications*. [S.l.]: Springer, 1987. p. 7–15.

LEE, H.; CHA, J. H. New stochastic models for preventive maintenance and maintenance optimization. *European Journal of Operational Research*, Elsevier, v. 255, n. 1, p. 80–90, 2016.

LEVITIN, G.; XING, L.; DAI, Y. Optimal operation and maintenance scheduling in m-out-of-n standby systems with reusable elements. *Reliability Engineering & System Safety*, Elsevier, v. 211, p. 107582, 2021.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, Informs, v. 21, n. 2, p. 498–516, 1973.

LIU, Y. et al. Integrated production planning and preventive maintenance scheduling for synchronized parallel machines. *Reliability Engineering & System Safety*, Elsevier BV, v. 215, p. 107869, nov. 2021.

LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. A beginner's introduction to iterated local search. In: *Proceedings of MIC*. [S.l.: s.n.], 2001. v. 4, p. 1–6.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search: Framework and applications. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2019. p. 129–168.

LU, S. et al. A hybrid DBH-VNS for high-end equipment production scheduling with machine failures and preventive maintenance activities. *Journal of Computational and Applied Mathematics*, Elsevier BV, v. 384, p. 113195, mar. 2021.

MA, J. et al. A comprehensive comparison among metaheuristics (MHs) for geohazard modeling using machine learning: Insights from a case study of landslide displacement prediction. *Engineering Applications of Artificial Intelligence*, Elsevier BV, v. 114, p. 105150, set. 2022.

MARMIER, F.; VARNIER, C.; ZERHOUNI, N. Static et dynamic scheduling of maintenance activities under the constraints of skills. *Journal of Operations and Logistics*, v. 2, n. 3, p. I–1, 2009.

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, American Institute of Physics, v. 21, n. 6, p. 1087–1092, 1953.

MIGUEL, P. A. C. et al. Metodologia de pesquisa em engenharia de produção e gestão de operações. *Rio de Janeiro: Elsevier*, 2010.

PACHECO, J. et al. Variable neighborhood search with memory for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. *Knowledge-Based Systems*, Elsevier, v. 145, p. 236–249, 2018.

PINEDO, M. *Scheduling: theory, algorithms, and systems. 5-th ed. Cham.* [S.l.]: Springer, 2016.

QI, X.; CHEN, T.; TU, F. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, Springer, v. 50, n. 10, p. 1071–1078, 1999.

RABADI, G.; MORAGA, R. J.; AL-SALEM, A. Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, Springer, v. 17, n. 1, p. 85–97, 2006.

RUIZ-TORRES, A. J.; PALETTA, G.; M'HALLAH, R. Makespan minimisation with sequence-dependent machine deterioration and maintenance events. *International Journal of Production Research*, Informa UK Limited, v. 55, n. 2, p. 462–479, maio 2016.

SANTOS, H. G. et al. Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, Wiley Online Library, v. 26, n. 2, p. 707–724, 2019.

SHELDON, M. R.; FILLYAW, M. J.; THOMPSON, W. D. The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International*, Wiley Online Library, v. 1, n. 4, p. 221–228, 1996.

SWANSON, L. Linking maintenance strategies to performance. *International Journal of Production Economics*, Elsevier, v. 70, n. 3, p. 237–244, 2001.

UPASANI, K. et al. Distributed maintenance planning in manufacturing industries. *Computers & Industrial Engineering*, Elsevier, v. 108, p. 1–14, 2017.

VALLADA, E.; RUIZ, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, Elsevier, v. 211, n. 3, p. 612–622, 2011.

WANG, J.; MIAO, Y. Optimal preventive maintenance policy of the balanced system under the semi-markov model. *Reliability Engineering & System Safety*, Elsevier, v. 213, p. 107690, 2021.

WANG, Y. et al. Source term estimation of hazardous material releases using hybrid genetic algorithm with composite cost functions. *Engineering Applications of Artificial Intelligence*, Elsevier BV, v. 75, p. 102–113, out. 2018.

XU, J. et al. 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*, Taylor & Francis, v. 40, n. 3, p. 751–766, 2019.

Apêndices

APÊNDICE A – Resultados das implementações GRASP, SA, ILS para as instâncias P

			GRASP			SA			ILS		
n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
20	2	2	**434**	**434**	0,0	**434**	**434**	0,0	**434**	**434**	0,0
20	3	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
20	4	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
20	5	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
20	10	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
30	2	2	**434**	**434**	0,0	**434**	**434**	0,0	**434**	**434**	0,0
30	3	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
30	4	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
30	5	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
30	10	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
40	2	2	**650**	**650**	0,0	**650**	**650**	0,0	**650**	**650**	0,0
40	3	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
40	4	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
40	5	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
40	10	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
60	2	2	693,2	**650**	96,6	**650**	**650**	0,0	**650**	**650**	0,0
60	3	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
60	4	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
60	5	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
60	10	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
80	2	2	**1082**	**1082**	0,0	**1082**	**1082**	0,0	**1082**	**1082**	0,0
80	3	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
80	4	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
80	5	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
80	10	2	**219**	**219**	0,0	**219**	**219**	0,0	**219**	**219**	0,0
20	3	3	**579**	**579**	0,0	**579**	**579**	0,0	**579**	**579**	0,0
20	4	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
20	5	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
20	6	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
20	12	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
30	3	3	**579**	**579**	0,0	**579**	**579**	0,0	**579**	**579**	0,0
30	4	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
30	5	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
30	6	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
30	12	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
40	3	3	723	723	0,0	723	723	0,0	**694,2**	**579**	64,4
40	4	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
40	5	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
40	6	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
40	12	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
60	3	3	1774,2	1731	39,4	1745,4	1587	93,9	**1615,8**	**1515**	96,6
60	4	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
60	5	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
60	6	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
60	12	3	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0

80	3	3	4294,2	4251	64,4	4323	4251	72,0	**4251**	**4179**	50,9
80	4	3	**2524**	**2524**	0,0	2783,2	2740	96,6	**2524**	**2524**	0,0
80	5	3	2567,8	**2524**	96,8	2697,8	2525	180,7	**2524**	**2524**	0,0
80	6	3	2567,8	**2524**	96,3	2698	2525	96,7	**2524,2**	**2524**	0,4
80	12	3	2611,6	2525	118,1	2656	2525	118,2	**2524**	**2524**	0,0
20	3	4	**723**	**723**	0,0	**723**	**723**	0,0	**723**	**723**	0,0
20	4	4	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
20	5	4	**220**	**220**	0,0	220,2	**220**	0,4	**220**	**220**	0,0
20	6	4	**220**	**220**	0,0	220,2	**220**	0,4	**220**	**220**	0,0
20	12	4	**220**	**220**	0,0	220,8	**220**	0,4	**220**	**220**	0,0
30	3	4	**723**	**723**	0,0	**723**	**723**	0,0	**723**	**723**	0,0
30	4	4	**220**	**220**	0,0	**220**	**220**	0,0	**220**	**220**	0,0
30	5	4	**220**	**220**	0,0	220,6	**220**	0,5	**220**	**220**	0,0
30	6	4	**220**	**220**	0,0	220,8	**220**	0,4	**220**	**220**	0,0
30	12	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
40	3	4	**867**	**867**	0,0	**867**	**867**	0,0	**867**	**867**	0,0
40	4	4	**220**	**220**	0,0	306,4	**220**	78,9	**220**	**220**	0,0
40	5	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
40	6	4	**220**	**220**	0,0	220,8	**220**	0,4	**220**	**220**	0,0
40	12	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
60	3	4	2263,8	**2163**	64,4	2307	2307	0,0	**2235**	2235	0,0
60	4	4	**220**	**220**	0,0	306,4	**220**	78,9	**220**	**220**	0,0
60	5	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
60	6	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
60	12	4	**220**	**220**	0,0	221	221	0,0	**220**	**220**	0,0
80	3	4	3387	3315	88,2	3113,4	3099	32,2	**3070,2**	**3027**	64,4
80	4	4	364	364	0,0	277,6	**220**	78,9	**220**	**220**	0,0
80	5	4	221	221	0,0	221	221	0,0	**220**	**220**	0,0
80	6	4	221	221	0,0	221	221	0,0	**220**	**220**	0,0
80	12	4	221	221	0,0	221	221	0,0	**220**	**220**	0,0
20	4	5	**724**	**724**	0,0	**724**	**724**	0,0	**724**	**724**	0,0
20	5	5	**221**	**221**	0,0	**221**	**221**	0,0	**221**	**221**	0,0
20	6	5	**221**	**221**	0,0	221,2	**221**	0,4	**221**	**221**	0,0
20	7	5	**221**	**221**	0,0	221,4	**221**	0,5	**221**	**221**	0,0
20	14	5	**221**	**221**	0,0	221,6	**221**	0,5	**221**	**221**	0,0
30	4	5	**724**	**724**	0,0	**724**	**724**	0,0	**724**	**724**	0,0
30	5	5	**221**	**221**	0,0	**221**	**221**	0,0	**221**	**221**	0,0
30	6	5	**221**	**221**	0,0	221,4	**221**	0,5	**221**	**221**	0,0
30	7	5	**221**	**221**	0,0	221,8	**221**	0,4	**221**	**221**	0,0
30	14	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
40	4	5	882,4	**868**	32,2	**868**	**868**	0,0	**868**	**868**	0,0
40	5	5	**221**	**221**	0,0	278,6	**221**	78,9	**221**	**221**	0,0
40	6	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
40	7	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
40	14	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
60	4	5	1832,8	1804	39,4	1717,6	1660	32,2	**1616,8**	**1588**	39,4
60	5	5	**221**	**221**	0,0	**221**	**221**	0,0	**221**	**221**	0,0
60	6	5	**221**	**221**	0,0	221,8	**221**	0,4	**221**	**221**	0,0
60	7	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
60	14	5	**221**	**221**	0,0	222	222	0,0	**221**	**221**	0,0
80	4	5	3805,6	3604	164,2	3575,2	3532	39,4	**3517,6**	**3460**	60,2
80	5	5	624,2	**509**	64,4	624,2	**509**	64,4	**509**	**509**	0,0
80	6	5	510	510	0,0	510	510	0,0	**509**	**509**	0,0
80	7	5	510	510	0,0	510,2	510	0,4	**509**	**509**	0,0
80	14	5	510,2	510	0,4	511,4	510	1,5	**509,4**	**509**	0,5

APÊNDICE B – Resultados das implementações GRASP, SA, ILS para as instâncias M, G, GG

			GRASP			SA			ILS		
n	m	Q	Média	Melhor	Desvio	Média	Melhor	Desvio	Média	Melhor	Desvio
150	148	120	1.864,8	**1.850**	28,7	1.852,0	1.852	0,0	**1.850,4**	**1.850**	0,5
300	158	179	2.089,4	1.993	92,1	**1.991,8**	**1.991**	0,8	2.031,8	**1.991**	60,5
600	165	329	4.772,4	4.684	65,9	4.680,2	4.636	32,3	**4.676,2**	**4.630**	44,9
1200	186	519	11.381,0	11.167	254,1	11.156,4	11.089	69,4	**11.152,0**	**11.056**	79,6
2400	188	738	26.885,4	26.680	136,9	26.731,0	**26.665**	37,0	**26.729,8**	26.687	27,5
4800	197	1128	57.090,2	56.888	202,3	56.676,0	56.564	97,2	**56.605,6**	**56.527**	78,9
150	75	129	**30,0**	**30**	0,0	**30,0**	**30**	0,0	**30,0**	**30**	0,0
300	221	239	2.459,8	2.459	1,1	**2.458,6**	**2.456**	2,1	2.460,2	2.459	1,1
600	256	388	3.535,0	3.501	28,9	**3.500,4**	3.478	23,7	3.512,8	**3.465**	33,6
1200	263	666	10.068,2	9.972	115,0	9.699,4	**9.657**	27,8	**9.691,8**	9.675	17,1
2400	283	869	25.221,6	24.965	192,9	24.983,2	24.938	41,0	**24.958,4**	**24.826**	102,6
4800	78	1286	45.172,2	44.827	352,7	**44.632,6**	43.366	764,4	44.676,4	**42.735**	1.089,8
150	102	91	**3.273,0**	**3.273**	0,0	**3.273,0**	**3.273**	0,0	**3.273,0**	**3.273**	0,0
300	112	177	**3.363,4**	**3.363**	0,5	3.365,6	3.364	1,3	3.363,8	**3.363**	0,4
600	120	288	8.989,2	8.587	478,5	8.894,4	8.586	511,3	**8.657,2**	**8.580**	85,0
1200	122	470	26.628,4	25.189	1.879,0	25.751,0	25.081	540,7	**24.293,2**	**23.621**	510,8
2400	130	603	46.370,6	45.057	1.326,5	43.693,0	**40.773**	1.632,3	**43.666,4**	41.374	1.320,3
4800	130	720	**102.997,2**	**93.995**	5.916,6	105.183,0	103.086	1.711,3	104.483,6	102.933	1.938,5
150	57	126	**24,0**	**24**	0,0	**24,0**	**24**	0,0	**24,0**	**24**	0,0
300	75	181	35,8	**35**	0,8	36,0	36	0,0	**35,0**	**35**	0,0
600	77	215	**40,8**	**40**	0,4	57,2	47	6,8	41,0	41	0,0
1200	88	252	301,0	**300**	1,2	**300,2**	**300**	0,4	303,0	303	0,0
2400	90	278	603,8	**567**	49,9	567,8	**567**	0,4	**567,2**	**567**	0,4
4800	91	294	1.474,6	**1.422**	74,4	**1.428,0**	1.426	2,8	1.428,8	1.426	5,7
150	92	93	**49,0**	**49**	0,0	**49,0**	**49**	0,0	**49,0**	**49**	0,0
300	121	162	66,0	66	0,0	**65,0**	**65**	0,0	**65,0**	**65**	0,0
600	126	279	415,6	412	6,9	**410,0**	**410**	0,0	**410,0**	**410**	0,0
1200	130	420	607,4	543	64,3	545,2	537	7,3	**537,2**	**529**	4,7
2400	132	701	1.313,0	866	510,8	766,8	732	25,1	**732,8**	**715**	25,1
4800	135	990	4.157,2	3.247	925,0	**2.312,2**	**2.130**	122,6	2.347,8	2.249	98,1
150	71	101	106,8	**106**	0,8	**106,0**	**106**	0,0	**106,0**	**106**	0,0
300	119	176	310,0	310	0,0	**308,0**	**308**	0,0	**308,0**	**308**	0,0
600	120	292	5.665,6	5.665	0,5	5.659,8	**5.659**	0,4	**5.659,0**	**5.659**	0,0
1200	122	403	6.881,6	6.839	93,6	**6.829,0**	**6.829**	0,0	**6.829,0**	**6.829**	0,0
2400	126	561	**8.495,2**	8.484	16,4	8.514,0	8.476	21,3	8.534,2	**8.474**	94,0
4800	129	713	**19.441,2**	**16.617**	2.597,3	20.288,4	20.286	2,3	20.288,4	20.281	6,3
9600	132	816	31.663,6	30.451	770,5	30.199,2	30.009	404,1	**30.010,8**	**30.002**	7,1
19200	132	908	74.257,2	70.963	2.144,4	**69.199,8**	**68.711**	583,3	69.671,8	68.720	898,3
33484	145	1032	134.610,4	**128.856**	4.960,8	**133.291,2**	129.082	2.931,8	134.507,4	130.906	2.673,8

APÊNDICE C – Resultados das implementações GRASP, SA, ILS em comparação com a literatura

n	m	Q	Melhor valor conhecido	Objetivo				*Gap*			
				BRKMA Aquino	GRASP	SA	ILS	BRKMA Aquino	GRASP	SA	ILS
20	2	2	**434**	**434**	**434**	**434**	**434**	0,0%	0,0%	0,0%	0,0%
20	3	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
20	4	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
20	5	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
20	10	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
30	2	2	**434**	**434**	**434**	**434**	**434**	0,0%	0,0%	0,0%	0,0%
30	3	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
30	4	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
30	5	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
30	10	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
40	2	2	**434**	**434**	650	650	650	0,0%	49,8%	49,8%	49,8%
40	3	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
40	4	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
40	5	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
40	10	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
60	2	2	**434**	650	650	650	650	49,8%	49,8%	49,8%	49,8%
60	3	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
60	4	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
60	5	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
60	10	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
80	2	2	**866**	1.082	1.082	1.082	1.082	24,9%	24,9%	24,9%	24,9%
80	3	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
80	4	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
80	5	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
80	10	2	**219**	**219**	**219**	**219**	**219**	0,0%	0,0%	0,0%	0,0%
20	3	3	**579**	**579**	**579**	**579**	**579**	0,0%	0,0%	0,0%	0,0%
20	4	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	5	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	6	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	12	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	3	3	**579**	**579**	**579**	**579**	**579**	0,0%	0,0%	0,0%	0,0%
30	4	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	5	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	6	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	12	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	3	3	**579**	**579**	723	723	**579**	0,0%	24,9%	24,9%	0,0%
40	4	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	5	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	6	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	12	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
60	3	3	**1.443**	**1.443**	1.731	1.587	1.515	0,0%	20,0%	10,0%	5,0%
60	4	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
60	5	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
60	6	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%

60	12	3	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
80	3	3	~~**1.443**~~	~~1.443~~	4.251	4.251	4.179	0,0%	194,6%	194,6%	189,6%
80	4	3	~~**220**~~	~~220~~	2.524	2.740	2.524	0,0%	1047,3%	1145,5%	1047,3%
80	5	3	~~**220**~~	~~220~~	2.524	2.525	2.524	0,0%	1047,3%	1047,7%	1047,3%
80	6	3	~~**220**~~	~~220~~	2.524	2.525	2.524	0,0%	1047,3%	1047,7%	1047,3%
80	12	3	~~**220**~~	~~220~~	2.525	2.525	2.524	0,0%	1047,7%	1047,7%	1047,3%
20	3	4	**723**	**723**	**723**	**723**	**723**	0,0%	0,0%	0,0%	0,0%
20	4	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	5	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	6	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
20	12	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	3	4	**723**	**723**	**723**	**723**	**723**	0,0%	0,0%	0,0%	0,0%
30	4	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	5	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	6	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
30	12	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
40	3	4	**867**	**867**	**867**	**867**	**867**	0,0%	0,0%	0,0%	0,0%
40	4	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	5	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
40	6	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
40	12	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
60	3	4	**2.091**	**2.091**	2.163	2.307	2.235	0,0%	3,4%	10,3%	6,9%
60	4	4	**220**	**220**	**220**	**220**	**220**	0,0%	0,0%	0,0%	0,0%
60	5	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
60	6	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
60	12	4	**220**	**220**	**220**	221	**220**	0,0%	0,0%	0,5%	0,0%
80	3	4	**2.955**	**2.955**	3.315	3.099	3.027	0,0%	12,2%	4,9%	2,4%
80	4	4	**220**	**220**	364	**220**	**220**	0,0%	65,5%	0,0%	0,0%
80	5	4	**220**	**220**	221	221	**220**	0,0%	0,5%	0,5%	0,0%
80	6	4	**220**	**220**	221	221	**220**	0,0%	0,5%	0,5%	0,0%
80	12	4	**220**	**220**	221	221	**220**	0,0%	0,5%	0,5%	0,0%
20	4	5	**724**	**724**	**724**	**724**	**724**	0,0%	0,0%	0,0%	0,0%
20	5	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
20	6	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
20	7	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
20	14	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
30	4	5	**724**	**724**	**724**	**724**	**724**	0,0%	0,0%	0,0%	0,0%
30	5	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
30	6	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
30	7	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
30	14	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
40	4	5	**868**	**868**	**868**	**868**	**868**	0,0%	0,0%	0,0%	0,0%
40	5	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
40	6	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
40	7	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
40	14	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
60	4	5	**1.588**	**1.588**	1.804	1.660	**1.588**	0,0%	13,6%	4,5%	0,0%
60	5	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
60	6	5	**221**	**221**	**221**	**221**	**221**	0,0%	0,0%	0,0%	0,0%
60	7	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
60	14	5	**221**	**221**	**221**	222	**221**	0,0%	0,0%	0,5%	0,0%
80	4	5	~~**3.028**~~	~~3.028~~	3.604	3.532	3.460	0,0%	19,0%	16,6%	14,3%
80	5	5	~~**221**~~	~~221~~	509	509	509	0,0%	130,3%	130,3%	130,3%
80	6	5	~~**221**~~	~~221~~	510	510	509	0,0%	130,8%	130,8%	130,3%
80	7	5	~~**221**~~	~~221~~	510	510	509	0,0%	130,8%	130,8%	130,3%
80	14	5	~~**221**~~	~~221~~	510	510	509	0,0%	130,8%	130,8%	130,3%
150	148	120	~~**1.756**~~	~~1.756~~	1.850	1.852	1.850	0,0%	5,4%	5,5%	5,4%
300	158	179	~~**1.776**~~	~~1.932~~	1.993	1.991	1.991	8,8%	12,2%	12,1%	12,1%
600	165	329	~~**4.409**~~	~~4.609~~	4.684	4.636	4.630	4,5%	6,2%	5,1%	5,0%

1200	186	519	**~~10.784~~**	~~11.077~~	11.167	11.089	11.056	2,7%	3,6%	2,8%	2,5%
2400	188	738	**~~26.665~~**	~~27.179~~	26.680	26.665	26.687	1,9%	0,1%	0,0%	0,1%
4800	197	1128	**~~56.527~~**	~~61.138~~	56.888	56.564	56.527	8,2%	0,6%	0,1%	0,0%
150	75	129	**30**	**30**	**30**	**30**	**30**	0,0%	0,0%	0,0%	0,0%
300	221	239	**2.452**	**2.452**	2.459	2.456	2.459	0,0%	0,3%	0,2%	0,3%
600	256	388	**3.384**	3.388	3.501	3.478	3.465	0,1%	3,5%	2,8%	2,4%
1200	263	666	**9.527**	9.575	9.972	9.657	9.675	0,5%	4,7%	1,4%	1,6%
2400	283	869	**24.553**	24.713	24.965	24.938	24.826	0,7%	1,7%	1,6%	1,1%
4800	78	1286	**41.925**	42.781	44.827	43.366	42.735	2,0%	6,9%	3,4%	1,9%
150	102	91	**3.273**	**3.273**	**3.273**	**3.273**	**3.273**	0,0%	0,0%	0,0%	0,0%
300	112	177	**3.360**	3.362	3.363	3.364	3.363	0,1%	0,1%	0,1%	0,1%
600	120	288	**8.578**	8.584	8.587	8.586	8.580	0,1%	0,1%	0,1%	0,0%
1200	122	470	**21.930**	22.318	25.189	25.081	23.621	1,8%	14,9%	14,4%	7,7%
2400	130	603	**38.094**	38.764	45.057	40.773	41.374	1,8%	18,3%	7,0%	8,6%
4800	130	720	**92.320**	**92.320**	93.995	103.086	102.933	0,0%	1,8%	11,7%	11,5%
150	57	126	**24**	**24**	**24**	**24**	**24**	0,0%	0,0%	0,0%	0,0%
300	75	181	**35**	37	**35**	36	**35**	5,7%	0,0%	2,9%	0,0%
600	77	215	**40**	43	**40**	47	41	7,5%	0,0%	17,5%	2,5%
1200	88	252	**300**	390	**300**	**300**	303	30,0%	0,0%	0,0%	1,0%
2400	90	278	**567**	1.585	**567**	**567**	**567**	179,5%	0,0%	0,0%	0,0%
4800	91	294	**1.422**	198.340	**1.422**	1.426	1.426	13848,0%	0,0%	0,3%	0,3%
150	92	93	**49**	**49**	**49**	**49**	**49**	0,0%	0,0%	0,0%	0,0%
300	121	162	**65**	282	66	**65**	**65**	333,8%	1,5%	0,0%	0,0%
600	126	279	**410**	416	412	**410**	**410**	1,5%	0,5%	0,0%	0,0%
1200	130	420	**526**	679	543	537	529	29,1%	3,2%	2,1%	0,6%
2400	132	701	**715**	1.608	866	732	**715**	124,9%	21,1%	2,4%	0,0%
4800	135	990	**2.130**	6.088	3.247	**2.130**	2.249	185,8%	52,4%	0,0%	5,6%
150	71	101	**106**	**106**	**106**	**106**	**106**	0,0%	0,0%	0,0%	0,0%
300	119	176	**308**	**308**	310	**308**	**308**	0,0%	0,6%	0,0%	0,0%
600	120	292	**5.659**	5.665	5.665	**5.659**	**5.659**	0,1%	0,1%	0,0%	0,0%
1200	122	403	**6.829**	6.842	6.839	**6.829**	**6.829**	0,2%	0,1%	0,0%	0,0%
2400	126	561	**8.259**	8.775	8.484	8.476	8.474	6,2%	2,7%	2,6%	2,6%
4800	129	713	**16.617**	17.749	**16.617**	20.286	20.281	6,8%	0,0%	22,1%	22,0%
9600	132	816	**30.002**	38.004	30.451	30.009	**30.002**	26,7%	1,5%	0,0%	0,0%
19200	132	908	**68.711**	103.863	70.963	**68.711**	68.720	51,2%	3,3%	0,0%	0,0%
33484	145	1032	**128.856**	220.048	**128.856**	129.082	130.906	70,8%	0,0%	0,2%	1,6%