UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA DE TRANSPORTES E LOGÍSTICA

MATHEUS HENRIQUE NASCIMENTO DA SILVA

APLICAÇÕES DE META-HEURÍSTICAS NA RESOLUÇÃO DE PROBLEMAS DE SEQUENCIAMENTO DE TAREFAS

Joinville
2024

MATHEUS HENRIQUE NASCIMENTO DA SILVA

APLICAÇÕES DE META-HEURÍSTICAS NA RESOLUÇÃO DE PROBLEMAS DE SEQUENCIAMENTO DE TAREFAS

Trabalho apresentado como requisito para obtenção do título de bacharel em Engenharia de Transportes e Logística do Centro Tecnológico de Joinville da Universidade Federal de Santa Catarina.

Dra. Silvia Lopes de Sena Taglialenha

Joinville
2024

MATHEUS HENRIQUE NASCIMENTO DA SILVA

APLICAÇÕES DE META-HEURÍSTICAS NA RESOLUÇÃO DE PROBLEMAS DE SEQUENCIAMENTO DE TAREFAS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel em Engenharia de Transportes e Logística, na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Joinville (SC), 04 de dezembro de 2024.

Banca Examinadora:

_____

Dra. Silvia Lopes de Sena Taglialenha
Orientadora
Presidente

_____

Dra. Christiane Wenck Nogueira
Membro
Universidade Federal de Santa Catarina

_____

Me. Natan Bissoli
Membro

AGRADECIMENTOS

Agradeço à minha família por todo o apoio ao longo de toda a minha jornada acadêmica. Jaime, Martha e Maria, vocês foram os pilares que sustentaram cada passo meu nesse caminho. O amor, o encorajamento e a compreensão que sempre demonstraram foram fundamentais para que eu pudesse alcançar este marco em minha vida.

Aos meus professores, sou grato pela dedicação em compartilhar seus conhecimentos e orientações, que foram essenciais para o desenvolvimento deste trabalho. Em especial, gostaria de agradecer à minha Orientadora, Dra. Silvia, cujo suporte desde o início do curso, com seus projetos e conselhos, foi decisivo para minha formação. Sem ela, minha trajetória não teria sido a mesma, e sou profundamente grato por sua orientação ao longo dos anos.

Aos amigos que fiz no decorrer do curso, meu sincero agradecimento. Vocês estiveram sempre dispostos a me ouvir e dar conselhos quando precisei, tornando os desafios mais leves e compartilhando momentos que enriqueceram esta jornada.

Por fim, agradeço também aos meus gestores, colegas de trabalho e às empresas nas quais tive a oportunidade de atuar. A experiência profissional adquirida durante esse período foi crucial para complementar meus estudos e para meu crescimento como profissional.

“A qualidade dos dados e a tomada de decisão baseada em dados andam de mãos dadas. Um compromisso de toda a organização com a governança de dados reduz os riscos e impulsiona o sucesso futuro de todos na empresa."(Scott Teal, 2022) - Gerente de Marketing de Produtos da Snowflake.

RESUMO

O sequenciamento de tarefas refere-se ao problema de determinar a sequência de um conjunto de tarefas ou atividades que deve ser realizado, de forma a melhorar algum critério de desempenho, como o tempo total de execução ou a utilização dos recursos disponíveis. Esse problema é comum em áreas como o planejamento da produção, logística e gestão de projetos, onde a alocação eficiente dos recursos e a minimização de custos são essenciais para a eficiência operacional.

Este trabalho aborda a aplicação de meta-heurísticas no problema de sequenciamento de tarefas, um desafio relevante nessas áreas, onde a alocação eficiente de recursos é crucial para o desempenho do sistema. O problema, frequentemente complexo devido à grande quantidade de variáveis e à necessidade de otimizar múltiplos critérios, é tratado com o uso de técnicas computacionais avançadas, como algoritmos genéticos, GRASP (Greedy Randomized Adaptive Search Procedure) e Simulated Annealing.

O principal objetivo desta pesquisa é aplicar meta-heurísticas para a resolução de problemas de sequenciamento de tarefas, equilibrando a qualidade das soluções com a viabilidade computacional. Para tanto, o trabalho envolve a implementação desses algoritmos em Python, configurados para a minimização da soma ponderada de custos de atraso e adiantamento, considerando tanto problemas com máquinas simples quanto em paralelo. A análise de desempenho se baseia em parâmetros como a qualidade da solução e o tempo de execução, além de um processo de calibração dos parâmetros de cada meta-heurística.

Além disso, são apresentados os resultados obtidos por meio da execução de diversos casos de teste, comparando o desempenho das técnicas aplicadas, e discutindo a eficácia das abordagens em diferentes configurações do problema. Com isso, o estudo não só contribui para a compreensão do impacto das meta-heurísticas no sequenciamento de tarefas, mas também propõe alternativas viáveis para situações onde a busca exaustiva pela solução ótima é computacionalmente inviável.

As meta-heurísticas destacam-se como ferramentas eficazes para resolver o problemas de sequenciamento de tarefas, apresentando vantagens significativas no tempo computacional, especialmente em instâncias de maior porte. Enquanto o método de programação linear inteira mista (PLIM) é capaz de alcançar soluções ótimas, ele enfrenta limitações quando o tamanho das instâncias aumenta, devido ao elevado custo computacional. Em contrapartida, algoritmos como o Algoritmo Genético, GRASP e Simulated Annealing fornecem soluções de alta qualidade em um tempo significativamente menor, tornando-se alternativas adequadas para cenários em que o PLIM se torna inviável. Esses resultados ressaltam a importância de considerar a relação entre a busca pelo ótimo e eficiência computacional ao selecionar métodos de solução.

Palavras-chave: sequenciamento de tarefas; meta-heurísticas. Algoritmo Genético. GRASP. Simulated Annealing.

ABSTRACT

Scheduling refers to the problem of determining the sequence of a set of tasks or activities to be performed in order to improve some performance criterion, such as total execution time or the utilization of available resources. This problem is common in areas such as production planning, logistics, and project management, where efficient resource allocation and cost minimization are essential for operational efficiency.

This work addresses the application of metaheuristics to the task sequencing problem, a relevant challenge in these fields, where efficient resource allocation is crucial to system performance. The problem, often complex due to the large number of variables and the need to optimize multiple criteria, is tackled using advanced computational techniques, such as Genetic Algorithms, GRASP (Greedy Randomized Adaptive Search Procedure), and Simulated Annealing.

The main objective of this research is to apply metaheuristics to solve task sequencing problems, balancing the quality of solutions with computational feasibility. To achieve this, the work involves implementing these algorithms in Python, configured to minimize the weighted sum of tardiness and earliness costs, considering both single-machine and parallel-machine problems. The performance analysis is based on parameters such as solution quality and execution time, along with a calibration process for each metaheuristic's parameters.

Additionally, the results obtained through the execution of various test cases are presented, comparing the performance of the applied techniques and discussing the effectiveness of the approaches under different problem configurations. Thus, the study not only contributes to understanding the impact of metaheuristics on task sequencing but also proposes viable alternatives for situations where exhaustive search for the optimal solution is computationally infeasible.

Metaheuristics stand out as effective tools for solving task sequencing problems, offering significant advantages in computational time, especially for larger instances. While the Mixed-Integer Linear Programming (MILP) method can achieve optimal solutions, it faces limitations as instance sizes increase due to high computational costs. In contrast, algorithms such as the Genetic Algorithm, GRASP, and Simulated Annealing provide high-quality solutions in significantly less time, making them suitable alternatives for scenarios where MILP becomes infeasible. These results highlight the importance of considering the relationship between the search for optimality and computational efficiency when selecting solution methods.

Keywords: Scheduling. metaheuristics. Genetic Algorithm. GRASP. Simulated Annealing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
Grasp	*Greedy Randomized Adaptive Search*
GA	Algoritmo Genético
AG	Algoritmo Genético
SA	*Simulated Annealing*
PLIM	Programação Linear Inteira Mista
EDD	*Earliest Due Date*
LRC	Lista Restrita de Candidatos
TCC	Trabalho de Conclusão de Curso
Capes	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
VNS	*Variable Neighborhood Search*
VND	*Variable Neighborhood Descent*

SUMÁRIO

1 INTRODUÇÃO

O planejamento estratégico é uma prática fundamental para as organizações, pois permite estabelecer direções, objetivos e estratégias de longo prazo para alcançar suas metas. Esse processo envolve a análise do ambiente externo e interno da empresa, levando em consideração fatores como concorrência, tendências de mercado, recursos disponíveis e capacidades organizacionais (Mintzberg, Ahlstrand e Lampel, 2000).

No âmbito estratégico, as empresas definem sua visão, missão e valores, bem como seus objetivos de longo prazo. É nesse contexto que são definidas as estratégias amplas para garantir a competitividade e sustentabilidade da organização. Essas estratégias são desenvolvidas nos níveis táticos e operacionais da organização (Robbins e Coulter, 1988).

No nível tático, o planejamento de médio prazo traduz as estratégias em planos de ação mais concretos, alinhando as operações da empresa com as metas estratégicas estabelecidas e garantindo a execução das estratégias definidas para alcançar vantagem competitiva e sustentabilidade no mercado (Andrian, 2023).

Já no nível operacional, encontra-se a execução das atividades diárias da organização. É nesse nível que ocorre o Planejamento e Controle da Produção (PCP), onde as tarefas são sequenciadas e coordenadas para garantir a eficiência operacional e a entrega de produtos ou serviços de qualidade aos clientes (Arenales et al., 2011).

Um mau planejamento da produção pode trazer diversos impactos negativos para uma empresa, refletindo-se diretamente na eficiência, nos custos e na satisfação dos clientes. Sem uma organização adequada, pode haver um desequilíbrio na capacidade produtiva, o que leva a atrasos e ociosidade de recursos, elevando os custos operacionais e reduzindo a competitividade no mercado (Corrêa, Gianesi e Caon, 2007).

Um planejamento inadequado gera falta de insumos ou excesso de estoque, resultando em desperdício de materiais e necessidade de armazenagem, além de aumentar o risco de obsolescência de produtos. Além disso, a falta de sincronização entre os setores pode comprometer a qualidade final dos produtos, já que a pressão por prazos pode reduzir os controles de qualidade e forçar a execução apressada das etapas de produção (Cher, 1990).

Para os clientes, o mau planejamento pode significar falhas na entrega, tanto em termos de prazo quanto de qualidade, afetando a confiabilidade da marca e reduzindo a satisfação do consumidor. Assim, a empresa corre o risco de perder contratos importantes e reduzir sua base de clientes. No longo prazo, esses problemas podem resultar em diminuição da margem de lucro, dificuldades financeiras e até prejuízos

(Corrêa, Gianesi e Caon, 2007).

Existe, entretanto, uma área de pesquisa chamada sequenciamento de tarefas, que busca mitigar esses impactos negativos ao melhorar a organização e a execução das atividades produtivas. Ela oferece métodos e algoritmos que ajudam a definir a melhor ordem para as tarefas, equilibrando a capacidade produtiva e melhorando o atendimento às demandas do mercado (Arenales et al., 2011).

O problema de sequenciamento de tarefas envolve a atribuição de uma ordem específica para a execução de um conjunto de tarefas, em determinado período de tempo, considerando restrições de precedência, recursos e tempo de processamento. A falta de um sequenciamento adequado pode resultar em atrasos na produção, excesso de estoque, aumento nos custos operacionais e insatisfação dos clientes (Pinedo, 2016).

Ao melhorar o sequenciamento de tarefas em uma linha de produção, é possível reduzir os tempos de espera entre as etapas do processo, minimizando assim o tempo total necessário para concluir um determinado produto. Isso não apenas aumenta a produtividade, como também reduz os custos associados ao tempo ocioso de equipamentos e mão de obra, além de possibilitar uma resposta mais ágil às demandas do mercado (Boukedroun et al., 2023).

Métodos exatos, como programação linear inteira mista (PLIM) e programação dinâmica, têm sido tradicionalmente utilizados para resolver o problema de sequenciamento. No entanto, devido à complexidade combinatória e à explosão de possibilidades de solução, o problema é classificado como NP-hard, o que significa que não existem algortimos conhecidos que possam resolver todas as instâncias do problema em tempo polinomial em função da entrada de dados do problema (Pinedo, 2016).

Diante dessa dificuldade computacional, as heurísticas e, em particular, as meta-heurísticas, tem sido utilizadas como ferramentas para encontrar soluções aproximadas para instâncias que os métodos exatos não conseguem resolver (Boukedroun et al., 2023). As heurísticas são técnicas específicas que aplicam regras simples ou baseadas na experiência para gerar boas soluções em um tempo viável, mesmo que não garantam a solução ótima. Por exemplo, no contexto de scheduling, uma heurística comum é o EDD (Earliest Due Date), que prioriza tarefas com os prazos de entrega mais próximos. As meta-heurísticas são algoritmos aproximativos que fornecem estratégias genéricas para explorar espaços de busca complexos e encontrar soluções de alta qualidade de forma eficiente (Arenales et al., 2011).

Dado os impactos negativos que um sequenciamento ineficiente de tarefas pode gerar nas empresas, como aumento de custos, ociosidade de recursos e atrasos na entrega de produtos, neste trabalho é considerado um estudo na problemática de sequenciamento de tarefas, cujos objetivos são destacados a seguir.

1.1 OBJETIVOS

Para resolver a problemática de sequenciamento de tarefas, são propostos neste trabalho os seguintes objetivos.

1.1.1 Objetivo Geral

Aplicar meta-heurísticas para resolver problemas de sequenciamento de tarefas.

1.1.2 Objetivos Específicos

- Aplicar PLIM para resolver instâncias pequenas;
- Desenvolver um algoritmo meta-heurístico com base no Algoritmo Genético de HOLLAND (1992);
- Desenvolver um algoritmo meta-heurístico com base no GRASP de Feo, Resende e Smith (1994);
- Desenvolver um algoritmo meta-heurístico com base no Simulated Annealing de Kirkpatrick, Gelatt e Vecchi (1983);
- Comparar o desempenho do métodos propostos.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado em cinco capítulos, incluindo esta introdução.

No Capítulo 2 é apresentada a fundamentação teórica, onde são abordados conceitos essenciais para a compreensão do estudo, como a visão geral sobre o contexto de sequenciamento de tarefas e a aplicação de métodos de otimização. Além disso, são explorados os principais algoritmos e métodos relevantes para a pesquisa, com destaque para os modelos exatos e meta-heurísticas.

No Capítulo 3 é detalhada a metodologia adotada para o desenvolvimento do trabalho, englobando a definição do problema, a construção das instâncias de teste, os algoritmos empregados e as estratégias de avaliação dos resultados. Esta seção também descreve as etapas de implementação e as ferramentas utilizadas para a resolução do problema.

No Capítulo 4 são apresentados os resultados obtidos com a aplicação dos modelos propostos, seguidos por uma análise comparativa dos métodos em termos de eficiência e precisão na resolução de problemas de sequenciamento.

Finalmente, o Capítulo 5 reúne as considerações finais, destacando as principais contribuições da pesquisa, as limitações encontradas e sugestões para estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados conceitos sobre sequenciamento de tarefas e as definições de heurísticas e meta-heurísticas. Primeiramente, apresenta-se a definição do problema de sequenciamento de tarefas, destacando a importância da otimização neste contexto. Em seguida, são discutidas as técnicas tradicionalmente empregadas para a resolução desse tipo de problema, que incluem métodos exatos e aproximativos. As meta-heurísticas aplicadas na pesquisa são apresentadas como abordagens que visam encontrar soluções factíveis em um tempo computacional viável, especialmente em situações onde os métodos tradicionais podem se tornar inviáveis devido à dificuldade de resolução e tempo de processamento do problema.

2.1 SEQUENCIAMENTO DE TAREFAS

O problema de sequenciamento de tarefas (scheduling) refere-se a um desafio fundamental em áreas como a gestão da produção e o planejamento de operações. Este problema envolve a determinação da ordem ótima de execução de um conjunto de tarefas para otimizar um ou mais critérios, como tempo de conclusão, custo ou utilização de recursos (Pinedo, 2016).

Na sua forma mais básica, o problema de sequenciamento de tarefas consiste em atribuir a cada tarefa um tempo de início e um tempo de término de modo que determinadas restrições sejam satisfeitas, como dependências entre tarefas, disponibilidade de recursos e janelas de tempo específicas. Em contextos mais complexos, como em sistemas de produção ou projetos industriais, há considerações adicionais, como minimização de tempos de espera, maximização da utilização de recursos ou redução de custos operacionais (Pinedo, 2016).

O sequenciamento de tarefas é aplicável em diversos contextos que exigem o gerenciamento eficiente de recursos limitados e a organização de atividades em um cronograma otimizado. Na alocação de pessoas, por exemplo, o sequenciamento é utilizado para distribuir colaboradores entre projetos ou tarefas, considerando habilidades, prazos e carga de trabalho (Pinedo, 2016).

No setor logístico, destaca-se o berço de um porto, onde é necessário coordenar o embarque e desembarque de navios, respeitando janelas de tempo, restrições físicas e recursos disponíveis (Bierwirth e Meisel, 2010). Além disso, o sequenciamento de tarefas em linhas de produção é um dos exemplos mais clássicos, otimizando a ordem de processamento de produtos para reduzir atrasos e maximizar a eficiência (Baker Trietsch, 2009).

Essas aplicações também se estendem a cenários como agendamento de

máquinas em indústrias (Pinedo, 2016), planejamento de manutenção de equipamentos, alocação de salas em hospitais ou instituições de ensino e até no processamento de dados em sistemas computacionais (Leung, 2004), evidenciando a versatilidade e a importância do sequenciamento para a solução de problemas reais em diferentes áreas.

O artigo "Job Shop Scheduling Problem: Literature Review", escrito por Rahmatika et al. (2020), realiza uma revisão da literatura sobre o problema de Job Shop Scheduling (JSSP), destacando os principais métodos, técnicas e abordagens utilizadas para resolver esse problema clássico de otimização combinatória. O estudo fornece uma análise das estratégias exatas, heurísticas e meta-heurísticas aplicadas ao JSSP, oferecendo uma base teórica e referências importantes para quem busca aprofundar o conhecimento nesse tema. Recomenda-se a leitura deste artigo para compreender melhor os avanços recentes e os desafios associados ao problema de sequenciamento de tarefas.

2.2 MÉTODOS EXATOS

Os métodos exatos buscam a solução ótima de um problema, explorando todo o espaço de busca, entende-se por solução ótima a melhor solução possível para um determinado problema. Entre os métodos exatos mais utilizados estão a Programação Dinâmica e Programação Linear Inteira Mista . Esses métodos garantem a obtenção da solução ótima, mas sua aplicabilidade é limitada a problemas de pequena escala ou com estrutura especial, devido ao crescimento exponencial da complexidade computacional (Arenales et al., 2011).

2.2.1 Sequenciamento em máquina única

Um método exato de sequenciamento de tarefas em máquina única consiste em encontrar a melhor ordem de execução das tarefas em um único recurso, respeitando restrições como tempos de processamento e prazos de entrega, a fim de otimizar algum critério de desempenho (como minimizar o atraso total ou o tempo de conclusão). Esse tipo de problema pode ser modelado matematicamente como um problema de otimização combinatória e resolvido utilizando técnicas exatas, como a programação linear inteira mista (PLIM) (Taillard, 1993). Segundo Pinedo (2016), o problema de sequenciamento em máquina simples é considerado NP-Hard, o que significa que não existe um algoritmo conhecido que consiga resolver este problema em tempo polinomial.

Para o caso de máquina única, a Programação Linear Inteira Mista (PLIM) é utilizada visando a modelagem precisa do problema de scheduling. Esse modelo matemático permite sequenciar as tarefas de modo otimizar a métrica desejada. O

uso de PLIM em problemas de máquina única é altamente eficaz quando se busca uma solução exata, uma vez que não há a necessidade de simplificar o problema, porém, quando trata-se de capacidade computacional, o método exato pode ser inviável, a depender do tamanho da instância do problema (Pinedo, 2016). Abaixo são apresentados os parâmetros e variáveis de decisão utilizados na modelagem do problema de sequenciamento de máquina única sugerido por Taha (2014).

Parâmetros:

- m: Quantidade de tarefas;
- p_i: Tempo de execução da tarefa i, para $i = 1, \ldots, m$;
- d_i: Prazo de entrega da tarefa i, para $i = 1, \ldots, m$;
- C_{ti}: Custo ou multa de atraso associado à tarefa i, indicando o impacto de atrasos na execução, para $i = 1, \ldots, m$;
- C_{ei}: Custo ou multa de adiantamento associado à tarefa i, indicando o impacto de atrasos na execução, para $i = 1, \ldots, m$;
- M: Uma constante positiva grande, conhecido como *Big* M, que permite a formulação de restrições de precedência. Esse valor garante que, quando uma tarefa i deve preceder a tarefa j, a condição seja validada matematicamente.

Variáveis de Decisão:

- x_j: Tempo de início da tarefa j, para $j = 1, \ldots, m$;
- e_j: Tempo de adiantamento da tarefa j, ou seja, representa o tempo que que a tarefa fica pronta antes de sua data de entrega, para $j = 1, \ldots, m$;
- t_j: Tempo de atraso da tarefa j, ou seja, representa o tempo que que a tarefa fica pronta depois de sua data de entrega, para $j = 1, \ldots, m$;
- y_{ij}= $\begin{cases} 1, \text{se tarefa } i \text{ precede tarefa } j \\ 0, \text{caso contrário} \end{cases}$,$\forall i, j = 1, \ldots, m, i \neq j.$

$$\text{Minimizar } Z = \sum_{i=1}^{m} (C_{ti} \cdot t_i + C_{ei} \cdot e_i) \tag{1}$$

s.a:

$$x_i - x_j + M y_{ij} \geq p_j \quad \forall i, j = 1, \ldots, m \quad i \neq j \tag{2}$$

$$-x_i + x_j + M(1 - y_{ij}) \geq p_i \quad \forall i, j = 1, \ldots, m \quad i \neq j \tag{3}$$

$$x_j + e_j - t_j = d_j - p_j \quad \forall j = 1, \ldots, m \tag{4}$$

$$e_j, t_j, x_j \geq 0 \quad \forall j = 1, \ldots, m \tag{5}$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j = 1, \ldots, m \tag{6}$$

O modelo visa minimizar o valor total associado a uma função objetivo, o qual pode variar dependendo do objetivo da empresa, como por exemplo, o custo de atraso e adiantamento ponderado pela multa de cada tarefa (1).

Para garantir que não sejam processadas duas tarefas simultaneamente na mesma máquina, impõem uma logica de precedência por meio de uma variável binaria, indicando a sequencia entre tarefas especificas por meio das Inequações (2) e (3). Essas restrições asseguram que, caso uma tarefa preceda outra, seu início ocorra somente após o término da anterior, utilizando uma constante suficientemente grande para que a desigualdade seja válida. Assim, cada tarefa só pode ser iniciada no momento certo, de acordo com a sequência de execução estabelecida no modelo.

Há também uma restrição que define o tempo final de execução de cada tarefa (Inequação (4)), relacionando esse tempo com os tempos de espera e de atraso em função do prazo e do tempo de processamento específico de cada tarefa. Esse vínculo mantém coerência entre o tempo de início das tarefas e a margem entre seus prazos, assegurando que os tempos de espera e atraso estejam adequadamente ajustados ao cronograma.

Além disso, todas as variáveis de tempo, como as de espera, atraso e início, são não-negativas (Inequação (5)), refletindo o sentido lógico dessas variáveis no contexto do problema. Para completar, a variável binária (6) estabelece a ordem única entre pares de tarefas, o que é necessário para determinar a sequência final do modelo.

2.2.2 Sequenciamento em várias máquinas em paralelo

O método exato de sequenciamento de tarefas em máquinas paralelas busca determinar a alocação e ordem de execução das tarefas em múltiplas máquinas simultâneas de forma a otimizar uma função objetivo, como, por exemplo, minimizar o tempo total de processamento (makespan) ou o atraso total. Esses problemas de máquinas paralelas podem ser classificados como máquinas idênticas, uniformes ou heterogêneas, dependendo das características de cada máquina (Fanjul-Peyro, Ruiz e Perea, 2019).

Modelos matemáticos, como a programação linear inteira mista (PLIM), são comumente utilizados para formular o problema de forma exata, levando em consideração variáveis de decisão binárias que indicam a alocação de cada tarefa a uma máquina específica. O objetivo é resolver o problema garantindo que a solução encontrada seja a melhor possível (Fanjul-Peyro, Ruiz e Perea, 2019). O problema de sequenciamento de máquinas em paralelo também é NP-Hard, pois esse baseia-se no problema de máquina simples, que, segundo Pinedo (2016), é considerado NP-Hard.

Para cenários em que há múltiplas máquinas disponíveis, o modelo de Programação Linear Inteira foi adaptado para o caso de máquinas em paralelo. Neste contexto, o objetivo é alocar tarefas entre as máquinas de maneira que o atraso total seja minimizado, considerando que as tarefas podem ser distribuídas entre os recursos disponíveis. A escolha do modelo PLI para máquinas em paralelo baseia-se em sua capacidade de produzir soluções ótimas em problemas de alocação de recursos, especialmente quando os tempos de processamento e as datas de entrega das tarefas variam (Pinedo, 2016). Essa abordagem é fundamental para o problema de scheduling abordado, pois considera a flexibilidade de distribuição das tarefas entre as máquinas, tornando-o adequado para uma linha de produção com múltiplos recursos. Abaixo são apresentados os parâmetros e variáveis de decisão utilizadas na modelagem do problema de sequenciamento de máquinas em paralelo, conforme apresentado por Barbosa (2021).

Parâmetros

- n: Quantidade de máquinas;
- m: Quantidade de tarefas;
- C_{ti}: Custo de atraso da tarefa i, para $i = 1, \ldots, m$;
- C_{ei}: Custo de adiantamento da tarefa i, para $i = 1, \ldots, m$;
- p_{ik}: Tempo de processamento da tarefa k na máquina i, para $i = 1, \ldots, m$ e $k = 1, \ldots, n$;
- d_k: Data de entrega da tarefa k, para $k = 1, \ldots, n$;
- M: Um valor suficientemente grande (*Big* M) para representar as condições de precedência.

Variáveis de Decisão

- x_{ik}: Tempo de início da tarefa k na máquina i, para $i = 1, \ldots, m$ e $k = 1, \ldots, n$;
- e_{ik}: Tempo de adiantamento da tarefa i na máquina k, com $i = 1, \ldots, m$ e $k = 1, \ldots, n$;
- t_{ik}: Tempo de atraso da tarefa i na máquina k, com $i = 1, \ldots, m$ e $k = 1, \ldots, n$;
- y_{ijk}: Variável binária que indica a ordem de execução entre tarefas na máquina k. $y_{ijk} = \begin{cases} 1, \text{se tarefa } i \text{ precede tarefa } j \text{ na máquina } k \\ 0, \text{caso contrário} \end{cases}, \forall i, j = 1, \ldots, m, k = 1, \ldots, n, i \neq j$;
- z_{ik}: Variável binária que indica se a tarefa k é atribuída à máquina i ($z_{ik} = 1$) ou não ($z_{ik} = 0$), para $i = 1, \ldots, m$ e $k = 1, \ldots, n$.

O modelo matemático proposto por Barbosa (2021) visa minimizar o custo total associado ao processamento de tarefas em máquinas paralelas. A função objetivo, apresentada em (7), define Z como a soma dos custos multiplicados pelos tempos de

processamento nas diferentes máquinas. O objetivo é encontrar a alocação de tarefas que minimize esse custo total.

$$\text{Minimize } Z = \sum_{i=1}^{m} \sum_{k=1}^{n} (C_{ti} \cdot t_{ik} + C_{ei} \cdot e_{ik}) \tag{7}$$

s.a:

$$MM \cdot (1 - y_{ijk}) + (x_{jk} - x_{ik}) \geq p_{ik}, \quad \forall i, j, k, i \neq j \tag{8}$$

$$MM \cdot y_{ijk} + (x_{ik} - x_{jk}) \geq p_{jk}, \quad \forall i, j, k, \, i \neq j \tag{9}$$

$$x_{ik} + e_{ik} - t_{ik} = d_i - p_{ik} + MM \cdot (1 - yz_{ik}), \quad \forall i, k \tag{10}$$

$$z_{ik} \leq MM \cdot yy_{ik}, \quad \forall i, k \tag{11}$$

$$\sum_{k=1}^{n} z_{ik} = 1, \quad \forall i, k \tag{12}$$

$$e_{ik}, t_{ik}, x_{ik} \geq 0, \quad \forall i, k \tag{13}$$

$$y_{ijk}, yz_{ik}, z_{ik} \in \{0, 1\}, \quad \forall i, j, k \tag{14}$$

$$\forall i, j = 1, \ldots, m, \quad \forall k = 1, \ldots, n. \tag{15}$$

As restrições do modelo garantem que as tarefas sejam alocadas e processadas corretamente nas máquinas. A inequação (8) estabelece que, para cada par de máquinas i e j, se a tarefa k for atribuída à máquina j, então o tempo de processamento na máquina i não pode ser menor do que o tempo necessário para concluir a tarefa k na máquina j com base na precedência definida por p_{ik}.

De forma análoga, a inequação (9) garante que, caso a tarefa k esteja sendo processada na máquina i, a máquina j também respeite a mesma lógica de precedência em relação a p_{jk}. Ambas as restrições utilizam uma constante M (um grande número) para representar as condições que permitem a flexibilidade na alocação das tarefas.

A equação (10) relaciona o tempo de conclusão de cada tarefa ao tempo em que a tarefa foi alocada e à sua duração em cada máquina. Essa relação assegura que a tarefa só pode ser concluída se o tempo de início e o tempo de execução forem considerados corretamente.

Além disso, a inequação (11) impõe uma condição que limita o valor da variável z_{ik} com base na variável binária que indica se a tarefa k está atribuída à máquina i ou não. A equação (12) é uma restrição de designação, que garante que cada tarefa seja atribuída a exatamente uma máquina, somando o valor de z_{ik} igual a 1 para cada máquina.

As variáveis e_{ik}, t_{ik}, x_{ik} devem ser não negativas, conforme indicado pela equação (13), enquanto as variáveis binárias y_{ijk}, yz_{ik}, z_{ik} são restritas a valores binários, conforme descrito na equação (14). Por fim, a equação (15) especifica que as restrições são aplicáveis para todas as máquinas e para todas as tarefas, garantindo a generalidade do modelo.

2.3 HEURÍSTICAS E META-HEURÍSTICAS

Como já comentado anteriormente, o problema de sequenciamento de tarefas pertence à classe de problemas NP-HARD, o que implica que não existe um algoritmo que possa encontrar a solução ótima para todos os cenários em um tempo polinomial (Pinedo, 2016). Portanto, as abordagens para resolver este problema geralmente envolvem o desenvolvimento de métodos aproximados que fornecem soluções viáveis em um tempo aceitável (Branke et al., 2015).

2.3.1 Heurísticas

Ao contrário dos métodos exatos que buscam a solução ótima, as heurísticas são algoritmos simplificados que priorizam a eficiência e a rapidez na busca por soluções subótimas que atendam aos critérios de otimização estabelecidos. As heurísticas podem ser classificadas em diferentes categorias, como aleatórias, construtivas, gulosas ou de refinamento. Heurísticas aleatórias exploram o espaço de soluções de forma não determinística, enquanto as construtivas constroem soluções a partir do zero, adicionando elementos progressivamente. Já as heurísticas gulosas tomam decisões locais que parecem ser as melhores no momento, e as de refinamento buscam melhorar uma solução já existente.

Exemplos comuns incluem heurísticas baseadas em regras simples, como a regra do menor tempo de processamento (SPT) ou a regra do maior tempo de processamento (LPT), que ordenam as tarefas com base em características específicas, como o tempo de execução (Pinedo, 2016). Embora não garantam a solução ótima, as heurísticas são amplamente utilizadas na prática devido à sua capacidade de lidar com problemas de grande escala e complexidade, fornecendo resultados aceitáveis em um tempo razoável (Arenales et al., 2011).

Nesse contexto, a heurística Earliest Due Date (EDD) se destaca ao ordenar tarefas pela proximidade das datas de entrega, garantindo uma solução ótima para

minimizar o atraso máximo, já que processa as tarefas na ordem de suas datas de entrega, reduzindo assim a possibilidade de grandes atrasos acumulados (Pinedo, 2016).

Segundo Pinedo (2016) a heurística EDD minimiza o atraso total e fornece a melhor solução possível em problemas de sequenciamento de tarefas com um único recurso, desde que as tarefas possam ser ordenadas livremente com base nas suas datas de entrega. A implementação desta heurística (Algoritmo 1) pode servir como base de comparação para estudos onde o objetivo da empresa seja minimizar o atraso total, dado que ela garante a solução ótima para essas condições.

Algoritmo 1: EDD (*Earliest Due Date*)

Entrada:
  T: Conjunto de tarefas, onde cada tarefa T_i tem um tempo de processamento p_i e um prazo de entrega D_i.
Saída:
  S: Sequência de tarefas ordenadas pelo prazo de entrega.
Ordenar as tarefas T em ordem crescente dos prazos de entrega D_i.
$S \leftarrow T$ ▷ Armazena a sequência ordenada
Inicializa o tempo de conclusão $C \leftarrow 0$.
Inicializa o atraso total $Atraso\_Total \leftarrow 0$.
Para cada tarefa T_i em S **Faça**
  $C \leftarrow C + p_i$ ▷ Atualiza o tempo de conclusão
  $A_i \leftarrow C - D_i$ ▷ Calcula o atraso da tarefa i
  Se $A_i > 0$ **Então**
    $Atraso\_Total \leftarrow Atraso\_Total + A_i$ ▷ Soma o atraso total
  Fim Se
Fim Para
Retornar S, $Atraso\_Total$ ▷ Retorna a sequência de tarefas e o atraso total

Fonte: Autor, 2024

2.3.2 Meta-heurísticas

As meta-heurísticas representam uma abordagem para lidar com problemas complexos e de grande escala. Ao contrário das heurísticas simples, as meta-heurísticas são estratégias de alto nível que buscam explorar o espaço de soluções de forma mais eficiente (Sorensen e Glover, 2013).

As meta-heurísticas podem ser classificadas em três categorias principais, dependendo da forma como manipulam as soluções. A primeira delas inclui as meta-heurísticas baseadas em busca local, como o *Simulated Annealing* (SA) (Kirkpatrick, Gelatt e Vecchi, 1983). Nessa abordagem, pequenas alterações são realizadas iterativamente em uma única solução inicial, explorando as proximidades dessa solução

na tentativa de encontrar uma alternativa superior. A simplicidade e a eficiência local tornam essa classe especialmente útil em problemas onde soluções próximas têm alto potencial de melhoria (Sorensen e Glover, 2013).

A segunda categoria é formada pelas meta-heurísticas construtivas, como o *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo, Resende e Smith, 1994). Nesse caso, as soluções são construídas gradualmente a partir de suas partes constituintes, seguindo um equilíbrio entre escolhas determinísticas e aleatórias. Essa abordagem é particularmente eficaz para problemas combinatórios, nos quais a construção incremental de soluções permite uma análise mais direcionada de partes do espaço de busca.

Por fim, as meta-heurísticas baseadas em populações, como o Algoritmo Genético (AG) (Holland, 1992), trabalham com um conjunto de soluções simultaneamente. Essas soluções são combinadas iterativamente para gerar novas, em um processo que simula a evolução biológica. A diversidade mantida pela população inicial e as combinações realizadas ao longo do algoritmo permitem explorar amplamente o espaço de busca, aumentando as chances de encontrar soluções de alta qualidade (Sorensen e Glover, 2013).

Esses métodos buscam encontrar soluções próximas ao ótimo global através da combinação de exploração (busca ampla) e exploração (busca intensiva) do espaço de soluções. Embora não garantam a solução ótima, as meta-heurísticas são altamente eficazes na prática, permitindo aos pesquisadores e profissionais encontrar soluções de boa qualidade para problemas complexos de sequenciamento de tarefas dentro de um tempo computacionalmente viável (Abdel-Basset, Abdel-Fatah e Sangaiah, 2018).

2.3.2.1 Algoritmo Genético

O Algoritmo Genético é uma técnica de otimização inspirada no processo de evolução biológica das espécies. O Algoritmo Genético é utilizado para resolver problemas de busca e otimização, onde uma população de soluções candidatas evolui ao longo de gerações, sujeita a operadores genéticos como seleção, recombinação (*crossover*) e mutação (Holland, 1992).

Cada solução candidata é representada como um indivíduo no espaço, geralmente codificado como um cromossomo que contém genes representando diferentes partes da solução (Holland, 1992). Esses indivíduos são avaliados com base em uma função de aptidão, que mede o quão boa é a solução em relação ao problema em questão. Ao longo de várias iterações os indivíduos mais aptos têm maior probabilidade de serem selecionados para reprodução, combinando seus genes através de *crossover* para gerar descendentes, que também sofrem mutações aleatórias. O Algoritmo Genético permite explorar amplamente o espaço de soluções em busca de uma solução ótima ou próxima dela (Holland, 1992). São explicados a seguir os passos

para implementação do Algoritmo Genético para um problema genérico, bem como no Algoritmo 2.

- **Inicialização da População:** No início, uma população inicial é gerada de forma aleatória ou com alguma heurística para cobrir o espaço de soluções possíveis. O tamanho da população (T) deve ser suficiente para garantir a diversidade de soluções, o que é fundamental para evitar a convergência prematura. A função objetivo de cada indivíduo é calculada para avaliar a qualidade inicial da população.
- **Seleção dos Pais:** Em cada geração, ocorre um processo de seleção dos indivíduos mais aptos para se tornarem pais e, posteriormente, gerarem a próxima população. Esse processo simula a seleção natural, onde os indivíduos com melhores desempenhos têm mais chances de transmitir suas características. Métodos comuns de seleção incluem torneio, roleta e ranking, os quais podem ser utilizados para escolher os indivíduos de maior aptidão.
- **Operador de Cruzamento:** O cruzamento é responsável por combinar pares de indivíduos selecionados, gerando novos indivíduos (filhos) que herdam características dos pais. A operação de cruzamento deve ser projetada para gerar filhos válidos dentro das restrições do problema. Em alguns casos, se o filho gerado não for uma solução viável, um mecanismo de correção é aplicado, garantindo que a nova geração respeite as condições do problema.
- **Avaliação da Função Objetivo:** Para cada filho gerado é calculado o valor da função objetivo, que mede a qualidade da solução representada por aquele indivíduo. O valor é comparado com o valor atual da melhor solução encontrada, e, se o novo indivíduo apresentar uma função objetivo superior (ou inferior, dependendo do problema), a melhor solução é atualizada.
- **Mutação:** Após o cruzamento, uma pequena quantidade de indivíduos na população pode ser alterada por meio de uma operação de mutação. Esse operador é fundamental para manter a diversidade genética na população e evitar que o algoritmo fique preso em um ótimo local. A mutação altera de forma aleatória alguma característica do indivíduo, permitindo a exploração de áreas novas do espaço de busca.
- **Critério de Parada:**
  O processo de evolução continua até que um critério de parada seja atingido. Esse critério pode ser o número máximo de gerações, um limite de tempo ou a convergência dos valores da função objetivo. Assim que o critério é atingido, o algoritmo retorna a melhor solução encontrada (S^*) e o valor da função objetivo associado (FO^*).

Algoritmo 2: Algoritmo Genético para Problemas de Otimização Genéricos

Entrada:
  Parâmetros do problema
  Critério de parada
  Tamanho da população
Saída: Melhor solução encontrada S^* e valor da função objetivo FO^*
$S^* \leftarrow$ NULL ▷ Inicializa a melhor solução como nula
$FO^* \leftarrow \infty$ ▷ Inicializa o valor da função objetivo como infinito
$populacao \leftarrow$ GerarPopulacaoInicial(T) ▷ Gera a população inicial
Avaliar $populacao$ ▷ Calcula a função objetivo para cada indivíduo
Enquanto não atingir o critério de parada **Faça**
  $pais \leftarrow$ SelecionarPais$(populacao)$ ▷ Seleciona os indivíduos mais aptos para reprodução
  $novos\_filhos \leftarrow \emptyset$ ▷ Inicializa lista para armazenar os filhos gerados
  Enquanto `tamanho de` $novos\_filhos < T$ **Faça**
    $individuo_1, individuo_2 \leftarrow$ SelecionarParaCruzamento$(pais)$ ▷ Seleciona dois indivíduos para cruzamento
    $filho \leftarrow$ Cruzamento$(individuo_1, individuo_2)$ ▷ Gera um novo indivíduo
    Se `não é uma solução válida`$(filho)$ **Então**
      $filho \leftarrow$ CorrigirSolução$(filho)$ ▷ Aplica correção, se necessário
    Fim Se
    $FO\_filho \leftarrow$ CalcularFuncaoObjetivo$(filho)$ ▷ Calcula a função objetivo do filho
    Se $FO\_filho < FO^*$ ou S^* é NULL **Então**
      $S^* \leftarrow filho$ ▷ Atualiza a melhor solução encontrada
      $FO^* \leftarrow FO\_filho$ ▷ Atualiza o melhor valor da função objetivo
    Fim Se
    $novos\_filhos \leftarrow novos\_filhos \cup \{filho\}$ ▷ Adiciona o filho à lista de novos indivíduos
  Fim Enquanto
  $populacao \leftarrow$ AtualizarPopulacao$(pais, novos_filhos)$ ▷ Atualiza a população com os novos indivíduos
  $populacao \leftarrow$ AplicarMutacao$(populacao)$ ▷ Aplica mutação em alguns indivíduos
  Avaliar $populacao$ ▷ Reavalia a população após a mutação
Fim Enquanto
Retornar S^* e FO^*

Fonte: Autor, 2024

2.3.2.2 GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é uma meta-heurística amplamente utilizada em problemas de otimização combinatória. Sua abordagem combina aspectos de construção gulosa e aleatoriedade controlada para encontrar soluções de alta qualidade de maneira eficiente. Durante a busca, o GRASP

constrói iterativamente soluções parciais com base em uma heurística gulosa, e então aplica uma componente de aleatoriedade para explorar novas possibilidades. Esse processo é repetido ao longo de múltiplas iterações, permitindo que o algoritmo escape de ótimos locais e alcance soluções mais próximas do ótimo global (Feo, Resende e Smith, 1994). A seguir são explicados os passos para implementação do GRASP para um problema genérico, bem como no Algoritmo 3.

- **Inicialização da Melhor Solução**: Inicialmente, a solução e o valor da função objetivo são indefinidos (`NULL`) e ∞, respectivamente. Define-se T, o tamanho da Lista de Candidatos Restrita (LRC), que determina o número de opções disponíveis para a construção da solução. Essa lista permite que a seleção dos elementos para a solução inicial seja balanceada, incorporando qualidade e diversidade.

- **Iteração Principal**: Para cada iteração i, o algoritmo segue duas etapas principais:
  1. **Construção da Solução**: A solução inicial S é criada de forma aleatória e gulosa, utilizando a LRC, que seleciona candidatos com base em uma medida de qualidade e diversidade. Essa abordagem balanceada permite que o algoritmo explore diferentes regiões do espaço de busca.
  2. **Refinamento (Busca Local)**: Uma busca local é aplicada para ajustar S e aprimorar o valor da função objetivo, resultando em uma solução mais otimizada. Esse passo permite que o algoritmo refine a solução inicial e alcance uma melhor performance.
- **Avaliação e Atualização da Melhor Solução**: Após cada iteração, a função objetivo é calculada para a solução refinada. Caso o valor encontrado seja menor que o atual (ou se for a primeira solução válida), a melhor solução S^* e a função objetivo FO^* são atualizados para armazenar a melhor solução. Essa atualização ocorre sempre que uma solução melhor é encontrada.
- **Critério de Parada**: As iterações continuam até atingir o número máximo de iterações N. No final, o algoritmo retorna a melhor solução S^* e o valor da função objetivo FO^*. Esse critério pode ser ajustado para outros limites, como tempo máximo de execução ou convergência da função objetivo, dependendo da aplicação.

Algoritmo 3: GRASP Genérico

Entrada:
  Parâmetros do problema
  N: Número de iterações
Saída: Melhor solução encontrada S^* e valor da função objetivo FO^*
$S^* \leftarrow$ NULL ▷ Inicializa a melhor solução
$FO^* \leftarrow \infty$ ▷ Inicializa o valor da função objetivo
$T \leftarrow$ Tamanho da Lista de Candidatos Restrita (LRC)

Para $i = 1$ to N **Faça**
  $S \leftarrow$ ConstruirSolucao(T) ▷ Gera uma solução inicial aleatória e gulosa
  Avaliar S ▷ Calcula a função objetivo da solução inicial
  $S \leftarrow$ Refinar(S) ▷ Aplica busca local para melhorar a solução
  $FO \leftarrow$ CalcularFO(S) ▷ Obtém valor da função objetivo para a solução refinada
  Se $FO < FO^*$ ou S^* é NULL **Então**
    $S^* \leftarrow S$ ▷ Atualiza a melhor solução encontrada
    $FO^* \leftarrow FO$ ▷ Atualiza o valor da função objetivo
  Fim Se
Fim Para
Retornar S^* e FO^*

2.3.2.3 Simulated Annealing

O Simulated Annealing (SA) é uma meta-heurística inspirada no processo físico de recozimento térmico, uma técnica utilizada na metalurgia para alterar as propriedades de metais e ligas. O recozimento consiste em aquecer o material para aumentar a energia de seus átomos e, em seguida, resfriá-lo gradualmente, permitindo que sua estrutura cristalina atinja um estado mais estável e com menor energia. Analogamente, o SA utiliza esse conceito para encontrar soluções aproximadas para problemas de otimização, equilibrando a exploração do espaço de busca e a convergência para soluções de alta qualidade (Kirkpatrick, Gelatt e Vecchi, 1983).

A partir de um mecanismo gerador de solução inicial, o algoritmo realiza movimentos aleatórios no espaço de busca, aceitando mudanças que melhoram a solução atual ou que, ocasionalmente, pioram-na, dependendo de um parâmetro de temperatura que controla a probabilidade de aceitação de soluções piores. Com o tempo, a temperatura é reduzida gradualmente, reduzindo a probabilidade de aceitação de soluções piores e permitindo ao algoritmo convergir para uma solução ótima ou próxima do ótimo global, mesmo em espaços de busca complexos ou com múltiplos ótimos locais (Kirkpatrick, Gelatt e Vecchi, 1983). Explica-se a seguir os passos para implementação do Simulated Annealing para um problema genérico, bem como no Algoritmo 4.

- **Iteração Principal:** O algoritmo entra em um loop que se repete enquanto a

temperatura T for maior que T_{min}. Para cada iteração i, o algoritmo realiza as seguintes etapas principais:

1. **Perturbação da Solução**: Uma nova solução S é gerada a partir da solução atual perturbando-a. Essa perturbação cria uma variação que permite ao algoritmo explorar o espaço de soluções, aumentando as chances de escapar de mínimos locais.
2. **Avaliação da Nova Solução**: A função objetivo é calculada para a nova solução S. Este passo é crucial para determinar a qualidade da solução em relação à melhor solução encontrada até o momento.

- **Avaliação e Atualização da Melhor Solução**: Após a avaliação da nova solução, verifica-se se o valor da função objetivo FO da nova solução é menor que o valor atual FO^*. Se for, isso indica que uma solução melhor foi encontrada, e, portanto, atualizamos S^* e FO^* com a nova solução e seu valor. Caso contrário, mesmo que a nova solução não seja melhor, o algoritmo calcula a probabilidade de aceitação baseada na diferença de valores da função objetivo e na temperatura atual. Se um número aleatório for menor que essa probabilidade, a nova solução é aceita como a atual, permitindo que o algoritmo continue explorando áreas menos promissoras de maneira controlada.
- **Critério de Parada**: O algoritmo continua a iterar até que a temperatura T diminua para T_{min}. Ao final do processo, o algoritmo retorna a melhor solução S^* encontrada e o respectivo valor da função objetivo FO^*. O critério de parada é flexível e pode ser ajustado conforme necessário, seja com base em um número fixo de iterações, no tempo máximo de execução, ou em outros critérios que indiquem que a solução está se estabilizando.

Algoritmo 4: Simulated Annealing Genérico

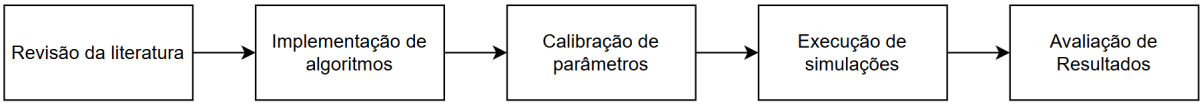
Entrada:
  Parâmetros do problema
  T_0: Temperatura inicial
  T_{min}: Temperatura mínima
  α: Taxa de resfriamento
  N: Número de iterações por temperatura
Saída: Melhor solução encontrada S^* e valor da função objetivo FO^*
$S^* \leftarrow$ NULL ▷ Inicializa a melhor solução
$FO^* \leftarrow \infty$ ▷ Inicializa o valor da função objetivo
$T \leftarrow T_0$ ▷ Define a temperatura inicial
Enquanto $T > T_{min}$ **Faça**
  Para $i = 1$ to N **Faça**
    $S \leftarrow$ Perturbar(S) ▷ Gera uma nova solução perturbando S
    $FO \leftarrow$ CalcularFO(S) ▷ Calcula a função objetivo da nova solução
    Se $FO < FO^*$ **Então**
      $S^* \leftarrow S$ ▷ Atualiza a melhor solução encontrada
      $FO^* \leftarrow FO$ ▷ Atualiza o valor da função objetivo
    Senão
      $p \leftarrow$ Exp $\left(\frac{FO^* - FO}{T}\right)$ ▷ Calcula a probabilidade de aceitação
      Se Random(0, 1) < p **Então**
        $S \leftarrow S$ ▷ Aceita a nova solução
      Fim Se
    Fim Se
  Fim Para
  $T \leftarrow T \cdot \alpha$ ▷ Resfriamento
Fim Enquanto
Retornar S^* e FO^*

Fonte: Autor, 2024

3 MATERIAIS E MÉTODOS

Nesta seção são detalhadas as metodologias utilizadas para alcançar os objetivos desta pesquisa, conforme os passos ilustrados na Figura 1.

Figura 1 – Fluxograma das etapas de desenvolvimento do TCC



Fonte: Autor, 2024

Inicialmente, é apresentada a aplicação da metodologia Prisma, reconhecida e estruturada para a realização de revisões sistemáticas da literatura, garantindo uma análise abrangente e criteriosa dos estudos relevantes ao tema em questão. Em seguida, será discutida a aplicação da modelagem matemática linear inteira para o problema de sequenciamento de tarefas.

Além disso, são exploradas três meta-heurísticas, incluindo algoritmos genéticos, GRASP e Simulated Annealing. Essas abordagens avançadas são utilizadas para lidar com a complexidade computacional do problema, permitindo a busca por soluções de alta qualidade dentro de um tempo computacionalmente viável. Também são realizadas etapas complementares, como calibração de parâmetros, execução de simulações e avaliação dos resultados obtidos (Figura 1).

3.1 REVISÃO SISTEMÁTICA DE LITERATURA

Para melhor compreensão do tema em estudo, aplicou-se uma revisão sistemática da literatura sobre o scheduling, métodos de solução utilizados e aplicações práticas, adotando-se as diretrizes da Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) (PAGE et al., 2021). Buscou-se compreender como a problematica de scheduling vem sendo aplicada como ferramenta de auxílio de tomada de decisão no contexto do planejamento da produção.

A metodologia PRISMA é um guia utilizado tanto em revisões sistemáticas, quanto em meta-análises (PAGE et al., 2021). Seu propósito é fornecer uma estrutura transparente durante a realização desses estudos, visando promover a qualidade e a integridade dos relatórios de pesquisa.

Ao seguir as diretrizes da PRISMA, os pesquisadores podem garantir uma abordagem sistemática na identificação e avaliação da evidência disponível em

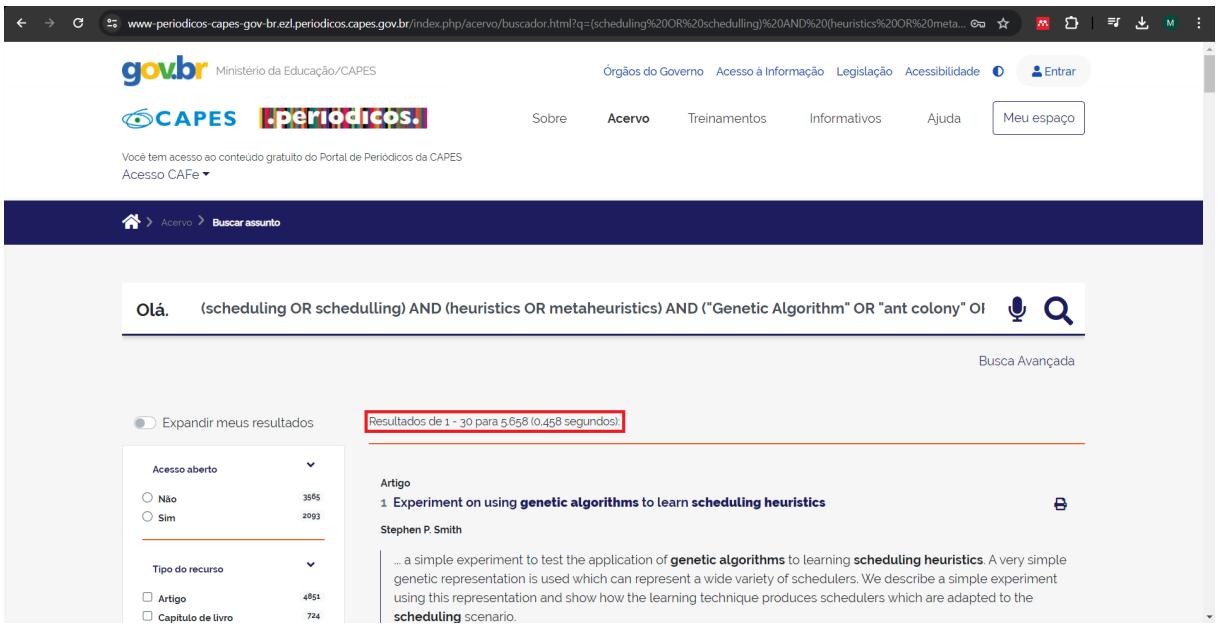
uma área específica de estudo. Além disso, a PRISMA facilita a replicabilidade e a comparabilidade dos estudos, permitindo que outros pesquisadores avaliem e reproduzam resultados de forma consistente (PAGE et al., 2021).

Para a aplicação da metodologia PRISMA neste trabalho, utilizou-se a base de dados de Periódicos da Capes, devido à sua cobertura e acesso disponível em 396 bases de dados com mais de 38 mil periódicos com texto completo (Capes, 2020). As palavras-chave selecionadas foram escolhidas para abranger o escopo da pesquisa, com foco em termos como "scheduling"e "*heuristics*", juntamente com variações de algoritmos conhecidos, como "*Genetic Algorithm*", "*ant colony*", "*Simulated Annealing*", "GRASP"e "VNS/VND", conforme observado na Figura 2.

Embora os métodos de otimização por colônia de formigas (Ant Colony) e busca por vizinhança variável (VNS/VND) não façam parte do escopo principal deste trabalho, optou-se por incluí-los na pesquisa bibliográfica inicial. Essa decisão foi tomada pois tais meta-heurísticas fazem parte do universo de algoritmos frequentemente aplicados ao problema de scheduling. A análise da literatura revelou que, embora esses métodos sejam aplicáveis, a aplicação do Algoritmo Genético (GA), GRASP e Simulated Annealing (SA) mostrou-se mais alinhada aos objetivos do estudo e com maior presença na literatura especializada, justificando sua escolha para implementação no presente trabalho.

A busca inicial, realizada no dia 09 de julho de 2024, resultou em um conjunto significativo de 5658 textos.

Figura 2 – Captura de tela de pesquisa no portal Capes



Fonte: Autor, 2024

Foi estabelecido o limite de tempo de publicação, restringindo a busca aos últimos 5 anos (2019-2024), o que reduziu o número para 1850 textos. Em seguida, os textos foram filtrados por idioma, optando-se por incluir apenas aqueles em inglês e português, resultando em 1847 textos. Com o intuito de garantir a qualidade e relevância dos artigos selecionados, optou-se por utilizar o filtro de revisão por pares, totalizando 1537 textos. As áreas específicas de interesse do trabalho foram usadas como critério adicional, concentrando-se em "Multidisciplinar", "Engenharias"e "Ciências Exatas e da Terra", resultando em 52 textos.

Como um dos principais objetivos ao aplicar a metodologia PRISMA é garantir a replicabilidade do estudo, independentemente das limitações de acesso dos pesquisadores, optou-se por selecionar exclusivamente artigos de acesso aberto. Essa abordagem resultou em um total de 26 registros, assegurando que todos os estudos incluídos na revisão estivessem disponíveis para consulta pública.

Utilizando o gerenciador de referências Mendeley, os textos foram coletados e a análise foi iniciada pelo título, seguida pelo resumo, introdução e conclusão, nessa ordem. Durante cada etapa da leitura, os artigos foram examinados e selecionados ou descartados de acordo com sua relevância para o escopo do estudo. Ao final do processo, 21 artigos foram importados para o Mendeley, dos quais 10 foram removidos após a análise dos títulos, 4 após a leitura dos resumos e 3 após a análise das introduções e conclusões, resultando na seleção de 4 artigos finais, de acordo com os critérios estabelecidos pela metodologia Prisma (Apêndice A).

Essa abordagem metodológica sistemática permitiu a seleção objetiva dos artigos, garantindo a qualidade e relevância dos estudos incluídos na revisão bibliográfica deste trabalho.

3.2 DADOS UTILIZADOS

Os dados utilizados neste trabalho foram gerados de forma aleatória, com o intuito de representar diferentes cenários de sequenciamento de tarefas. As instâncias incluem informações sobre a quantidade de tarefas, quantidade de máquinas, tempos de processamento por máquina, prazos de entrega e multas por adiantamento e atraso. A geração aleatória desses dados visa criar uma variedade de cenários que permitam avaliar o desempenho das meta-heurísticas aplicadas ao problema, garantindo assim que o algoritmo seja testado sob diferentes condições e seja capaz de lidar com uma gama ampla de situações. Os parâmetros gerados aleatoriamente são descritos na Tabela 1, bem como os limites mínimos e máximos de geração, a quantidade de máquinas e tarefas utilizadas são descritas na subseção 3.4.1.

Tabela 1 – Parâmetros gerados aleatoriamente

Parâmetro	Limite Inferior	Limite Superior
Tempo de Processamento P (Dias)	10	100
Prazo D (Dias)	100	300
Multa Atraso C_t (Reais/Dia)	1	10
Multa Adiantamento C_e (Reais/Dia)	1	10

Fonte: Autor, 2024

3.3 MÉTODOS DE SOLUÇÃO

Para resolver o problema de sequenciamento de tarefas duas abordagens distintas foram adotadas neste trabalho: métodos exatos e meta-heurísticos. Cada uma dessas técnicas foi selecionada devido às suas particularidades e vantagens no contexto de scheduling. O método exato, baseado na Programação Linear Inteira Mista, visa encontrar a solução ótima ao modelar o problema com precisão matemática (Arenales et al., 2011), no entanto não é aplicável para problemas de grandes instâncias, dado que o problema de scheduling é classificado como NP-Hard.

As meta-heurísticas, GRASP (Feo, Resende e Smith, 1994), Simulated Annealing (Kirkpatrick, Gelatt e Vecchi, 1983) e Algoritmo Genético (Holland, 1992), oferecem soluções de alta qualidade em cenários de maior complexidade, balanceando entre eficiência e viabilidade de execução em tempos computacionais reduzidos. Essa combinação de métodos proporciona uma base sólida para análise comparativa, permitindo a avaliação de cada abordagem em diferentes cenários do problema estudado.

3.3.1 Método Exato

Os métodos exatos foram considerados para resolver o problema de sequenciamento de tarefas em instâncias menores com o objetivo de obter soluções ótimas. Nesse sentido foram explorados dois modelos principais: Programação Linear Inteira Mista (PLIM) para máquina única e PLIM para máquinas em paralelo. Os algoritmos foram aplicados conforme apresentados nas subseções 2.2.1 e 2.2.2. A codificação deu-se por meio da linguagem Python e foi utilizada a biblioteca de otimização PULP, com o Solver Gurobi.

3.3.2 Meta-heurísticas

Para aplicar meta-heurísticas para resolver problemas complexos de sequenciamento de tarefas, é fundamental entender a abordagem que essas técnicas utilizam. Diferentes dos métodos que usam PLIM, que podem determinar a solução ótima do problema, as meta-heurísticas são métodos aproximados que buscam

boas soluções em um tempo computacional menor. Elas se destacam por balancear a exploração de novas áreas do espaço de busca e a intensificação em regiões promissoras, permitindo que soluções de qualidade sejam encontradas de maneira eficiente, especialmente em problemas grandes ou com muitas restrições. Nesta seção, apresentam-se três meta-heurísticas consideradas para resolver o problema de scheduling: GRASP (*Greedy Randomized Adaptive Search Procedure*), Algoritmo Genético (GA) e *Simulated Annealing* (SA), são detalhadas as características de cada método, bem como o papel que desempenham na busca por soluções eficientes. Optou-se por explicar as meta-heurísticas para o caso de máquinas em paralelo, pois o caso de máquina simples é alcançado ao considerar a existência de apenas uma máquina, enquanto o inverso não é verdadeiro. A estrutura de dados utilizadas nas duas abordagens são detalhadas a seguir:

3.3.2.1 Representação de uma solução para o caso de máquinas simples

Para o problema de sequenciamento em máquina simples com penalidades tanto para atrasos quanto para adiantamentos, a estrutura de dados do modelo deve conter informações sobre cada tarefa a ser processada. Os parâmetros considerados são descritos a seguir:

- Quantidade de tarefas (m);
- ID da Tarefa (i): Identificação única de cada tarefa;
- Tempo de Processamento (P_i): Tempo necessário para completar a tarefa;
- Prazo (D_i): Data de entrega da tarefa;
- Multa por Atraso (C_{Ti}): Custo associado ao atraso, caso a tarefa não seja concluída dentro do prazo;
- Multa por Adiantamento (C_{Ei}): Custo associado ao adiantamento, caso a tarefa seja concluída antes do prazo.

Apresenta-se na Tabela 2 uma instância com cinco tarefas:

Tabela 2 – Exemplo - instância com cinco tarefas

Tarefa	Processamento	Prazo de Entrega	Multa por Atraso	Multa por Adiantamento
1	3	5	2	1
2	4	7	3	2
3	2	4	5	1
4	6	10	1	3
5	5	8	4	2

Fonte: Autor, 2024

A solução para o problema de sequenciamento pode ser representada por

um vetor de dimensão da quantidade de tarefas, em que em cada posição i do vetor, tem-se a tarefa ID_i realizada na i-ésima posição. Cada solução corresponde a uma sequência, que impacta diretamente a função objetivo, pois o tempo de conclusão de cada tarefa deve ser comparado ao seu prazo de entrega para calcular as penalidades.

Por exemplo, o vetor de dimensão 5 ilustrado na Figura 3, indica que a tarefa 3 será processada primeiro, seguida pela tarefa 1, depois pela tarefa 5, pela 2 e por fim pela tarefa 4.

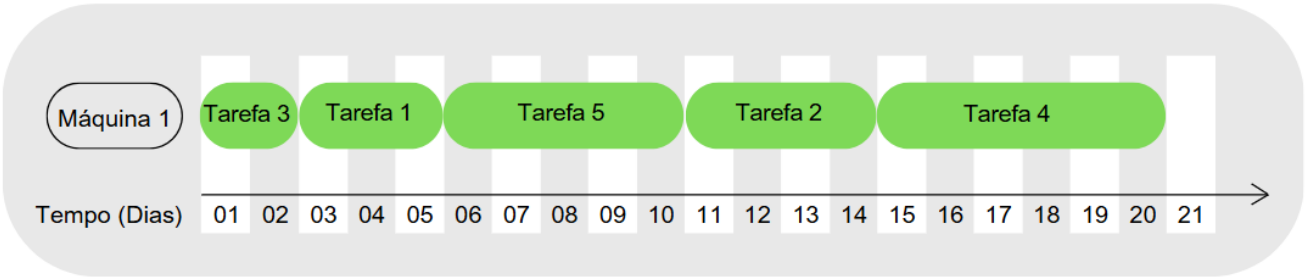
Figura 3 – Representação vetorial da solução do sequenciamento em máquina simples

Sequência	3	1	5	2	4

Fonte: Autor, 2024

A representação gráfica da solução do problema de sequenciamento de tarefas em máquina simples é apresentada na Figura 4.

Figura 4 – Representação gráfica da solução do sequenciamento em máquina simples



Fonte: Autor, 2024

A função objetivo (FO) do problema pode considerar penalidades tanto para atrasos quanto para adiantamentos. A seguir estão as etapas para o cálculo da FO:

- Tempo de Início da tarefa (x_i): O tempo de início de cada tarefa é calculado cumulativamente, conforme a sequência definida;
- Atraso (T_i): O atraso é dado por $T_i = \max(0, x_i + P_i - D_i)$, onde x_i é o tempo de início da tarefa i, P_i é o tempo de processamento e D_i é o prazo de entrega;
- Adiantamento (E_i): O adiantamento é dado por $E_i = \max(0, D_i - x_i - P_i)$, ocorrendo quando a tarefa é concluída antes do prazo.

A função objetivo total é a soma das penalidades de atraso e adiantamento para todas as tarefas, representada pela Equação (16).

$$FO = \sum_{i=1}^{m} C_{Ti} \cdot T_i + C_{Ei} \cdot E_i \tag{16}$$

No processo de geração de soluções, não ocorre infactibilidade, pois as técnicas utilizadas foram ajustadas para evitar a duplicação de indivíduos dentro de uma mesma solução. Para isso, duas abordagens foram aplicadas: a primeira é uma técnica pseudo aleatória controlada, que impede que um indivíduo seja selecionado mais de uma vez na mesma solução. A segunda técnica empregada é uma abordagem gulosa, na qual o melhor indivíduo é selecionado a cada iteração. Após a escolha, esse indivíduo é removido da lista de candidatos, o que garante que ele não seja selecionado novamente, e que todos os indivíduos sejam escolhidos exatamente uma vez.

3.3.2.2 Representação de uma solução para o caso de sequenciamento em máquinas em paralelo

Para o problema de sequenciamento em máquinas em paralelo com penalidades tanto para atrasos quanto para adiantamentos, a estrutura de dados do modelo deve conter informações sobre cada tarefa a ser processada. Os parâmetros considerados são descritos a seguir:

- Quantidade de máquinas (n);
- Quantidade de tarefas (m);
- ID da Tarefa (i): Identificação única de cada tarefa;
- Tempo de Processamento (P_{ik}): Tempo necessário para completar a tarefa (i) em cada máquina (k);
- Prazo (D_i): Data de entrega da tarefa;
- Multa por Atraso (C_{Ti}): Custo associado ao atraso, caso a tarefa não seja concluída dentro do prazo;
- Multa por Adiantamento (C_{Ei}): Custo associado ao adiantamento, caso a tarefa seja concluída antes do prazo.

Apresenta-se na Tabela 3 uma instância com 5 tarefas e duas máquinas:

Tabela 3 – Exemplo - Instância de 5 tarefas e 2 máquinas

Tarefa	Máquina	Processamento	Prazo de Entrega	Multa por Atraso	Multa por Adiantamento
1	1	3	5	2	1
1	2	3	5	2	1
2	1	4	7	3	2
2	2	4	7	3	2
3	1	2	4	5	1
3	2	2	4	5	1
4	1	6	10	1	3
4	2	6	10	1	3
5	1	5	8	4	2
5	2	5	8	4	2

Fonte: Autor, 2024

A solução para o problema de sequenciamento pode ser representada por uma matriz $2 \times m$, em que a primeira linha apresenta a sequência das tarefas, ou seja, a ordem em que elas serão processadas, e a segunda linha indica em qual máquina cada tarefa será processada. Aqui, m representa o número total de tarefas. Cada solução corresponde a uma sequência e escolha de máquina, que impacta diretamente a função objetivo, pois o tempo de conclusão de cada tarefa deve ser comparado ao seu prazo de entrega para calcular as penalidades.

Por exemplo, a sequência de tarefas e escolhas de máquinas apresentadas na Figura 5 indica que na máquina 1 a sequência será a tarefa 3, seguida pela 1 e 2, enquanto na máquina 2 a sequência será a tarefa 5, seguida pela tarefa 4.

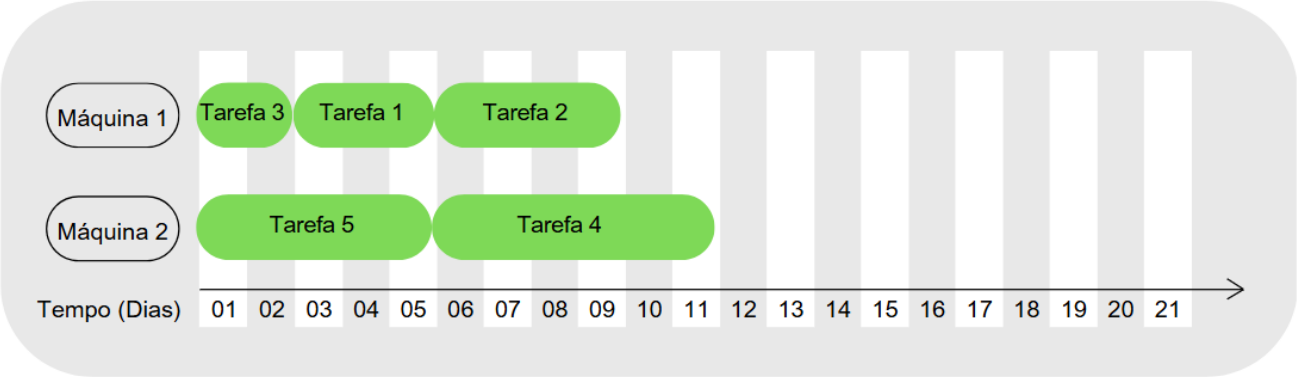
Figura 5 – Representação matricial da solução do sequenciamento em máquinas em paralelo

Sequência	3	1	5	2	4
Máquina	1	1	2	1	2

Fonte: Autor, 2024

A representação gráfica da solução do problema de sequenciamento de tarefas em máquinas em paralelo é apresentada na Figura 6.

Figura 6 – Representação gráfica da solução do sequenciamento em máquinas em paralelo



Fonte: Autor, 2024

A função objetivo (FO) do problema pode considerar penalidades tanto para atrasos quanto para adiantamentos. A seguir estão as etapas para o cálculo da FO:

- Tempo de início (x_{ik}): O tempo de início da tarefa i na máquina k é calculado cumulativamente, conforme a sequência definida, a primeira tarefa inicia no tempo 0;
- Atraso (T_ik): O atraso é dado por $T_i = \max(0, x_{ik} + P_{ik} - D_i)$, onde x_{ik} é o tempo de início da tarefa i na máquina k, P_{ik} o tempo de processamento da tarefa i na máquina k e D_i é o prazo de entrega da tarefa i;
- Adiantamento (E_{ik}): O adiantamento é dado por $E_{ik} = \max(0, D_i - x_{ik} - P_{ik})$, ocorrendo quando a tarefa é concluída antes do prazo.

A função objetivo total é a soma das penalidades de atraso e adiantamento para todas as tarefas em todas as máquinas, representada pela Equação (17).

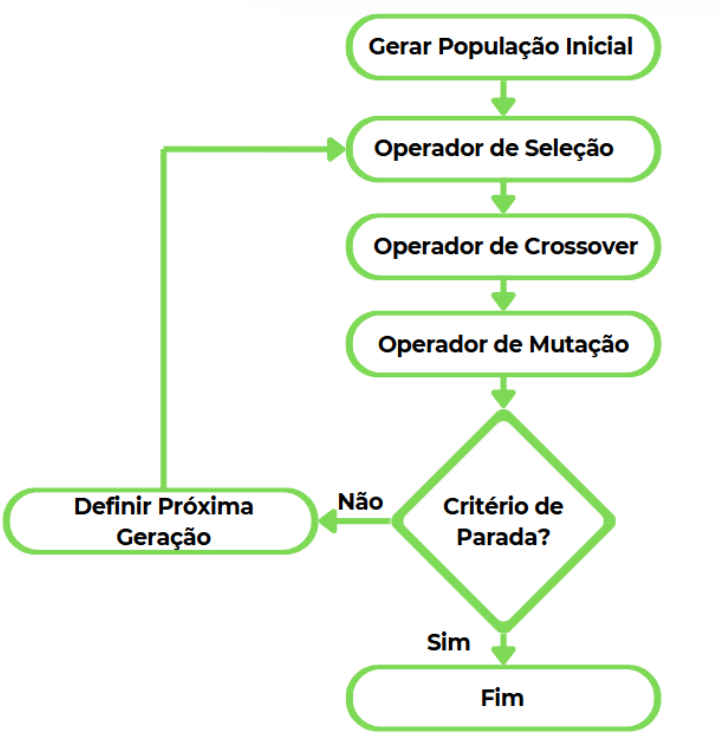
$$FO = \sum_{i=1}^{m} \sum_{k=1}^{n} C_{Ti} \cdot T_{ik} + C_{Ei} \cdot E_{ik} \tag{17}$$

No processo de geração de soluções não ocorrem infactibilidades, pois as técnicas utilizadas foram ajustadas para evitar a duplicação de tarefas dentro de uma mesma solução. Para isso, duas abordagens foram aplicadas: a primeira é uma técnica pseudo aleatória controlada, que impede que uma tarefa seja selecionada mais de uma vez na mesma solução. A segunda técnica empregada é uma abordagem gulosa, na qual o melhor indivíduo é selecionado a cada iteração. Após a escolha, esse indivíduo é removido da lista de candidatos, o que garante que ele não seja selecionado novamente, e que todos os indivíduos sejam escolhidos exatamente uma vez.

3.3.3 Algoritmo Genético

A Figura 7 apresenta os passos principais considerados na aplicação do Algoritmo Genético (GA) neste trabalho. Esses passos são descritos em detalhes nas subseções seguintes e compõem os elementos essenciais da abordagem utilizada para o sequenciamento de tarefas.

Figura 7 – Etapas do Algoritmo Genético



Fonte: Autor, 2024

Os exemplos ilustrados nas próximas seções tem como base o exemplo cujos dados são apresentados na Tabela 6.

3.3.3.1 Gerar população inicial

A geração da população inicial ocorre de forma aleatória, onde cada indivíduo representa uma sequência de tarefas (Figura 8) e escolhas de máquinas (Figura 9). Essa abordagem aleatória permite a criação de um conjunto diversificado de soluções iniciais, o que amplia o espaço de busca e aumenta a probabilidade de convergência para uma solução eficiente.

No caso específico do Algoritmo Genético foi optado por utilizar duas matrizes separadas para representar as soluções. Isso facilita o gerenciamento das populações durante as iterações do algoritmo. A primeira matriz contém as informações relativas às tarefas, como o índice de cada tarefa a ser executada. A segunda matriz é responsável

por armazenar as máquinas nas quais cada tarefa será alocada.

Figura 8 – Exemplo de população inicial do Algoritmo Genético - Sequência

ID	Sequência				
1	1	2	3	4	5
2	2	4	1	5	3
3	5	3	4	1	2
4	3	1	2	5	4
5	4	5	2	3	1

Fonte: Autor, 2024

Figura 9 – Exemplo de população inicial aleatória do Algoritmo Genético - Máquinas

ID	Máquinas				
1	1	2	2	1	1
2	2	2	1	1	2
3	1	1	2	1	2
4	1	1	1	2	2
5	2	2	2	1	1

Fonte: Autor, 2024

3.3.3.2 Operador de Seleção

A etapa de seleção utiliza o método de torneio, que consiste na escolha aleatória de um grupo de indivíduos da população para disputar a competição, onde apenas o mais apto entre eles é selecionado para o cruzamento, análogo à teoria da evolução. Esse processo contribui para a seleção de melhores indivíduos e acelera o processo de busca pela solução ideal. O processo de seleção de candidatos nesta aplicação sempre seleciona dois indivíduos, que são utilizados como base para o operador de crossover.

É apresentado na Figura 10 um exemplo de torneio, onde três indivíduos foram selecionados aleatoriamente e comparados entre si de acordo com o valor de função objetivo associado, calculado de acordo com a Equação (17) apresentada na subseção 3.3.3.2. Destaque para os indivíduos 2 e 4, que apresentam os melhores valores de função objetivo e portanto são selecionados para a próxima etapa do algoritmo.

Figura 10 – Exemplo de torneio

ID	Valor da FO
2	**16**
4	**14**
5	50

Fonte: Autor, 2024

3.3.3.3 Operador de crossover

O cruzamento é realizado utilizando o operador de crossover de um ponto de corte, no qual uma posição aleatória no vetor de solução dos indivíduos é selecionada para a troca de segmentos entre os pais. Este procedimento gera novos indivíduos que herdam características de ambos os progenitores, promovendo a variabilidade genética necessária para explorar o espaço de soluções. Observa-se na Figura 11 um exemplo de crossover entre os indivíduos 2 e 4, com ponto de corte na posição 2, definida randomicamente. Os novos indivíduos herdam as características dos indivíduos selecionados, por exemplo, o "Filho 1" herda o sequenciamento inicial do indivíduo 2 e o final do 4, o "Filho 2" o contrário.

Figura 11 – Exemplo de crossover

ID	Sequência				
2	2	4	1	5	3
4	3	1	2	5	4
Filho 1	2	4	2	5	4
Filho 2	3	1	1	5	3

Fonte: Autor, 2024

Nota-se que no exemplo ilustrado na Figura 11 os novos indivíduos podem apresentar infactibilidades para o problema de sequenciamento de tarefas, pois pode repetir a execução de uma ou mais tarefas e não executar outras. Nesse sentido, foi aplicado um cruzamento por mapeamento, em que os indivíduos gerados durante a operação de cruzamento são corrigidos para garantir a viabilidade das soluções. Durante esse processo, as tarefas duplicadas nos novos indivíduos são realocadas, enquanto as tarefas ausentes são inseridas, respeitando, sempre que possível, a ordem e estrutura da solução original.

O método do mapeamento consiste em corrigir as infactibilidades garantindo que cada tarefa seja atribuída exatamente uma vez em cada indivíduo. Para isso, o mapeamento realoca tarefas duplicadas e insere aquelas que ficaram ausentes, preservando o máximo possível a ordem e estrutura da solução original (Figura 12).

Essa abordagem assegura que os indivíduos gerados sejam válidos e que o operador de cruzamento produza soluções adequadas para o problema, evitando desperdícios computacionais e aumentando a eficiência da busca por boas soluções.

Figura 12 – Exemplo de crossover do Algoritmo Genético com correção por mapeamento

ID	Sequência				
2	2	4	1	5	3
4	3	1	2	5	4
Filho 1	2	4	1	5	3
Filho 2	3	1	2	5	4

Fonte: Autor, 2024

3.3.3.4 Operador de mutação

A mutação desempenha um papel importante na diversidade da população, sendo implementada neste trabalho como uma operação de inversão da ordem de duas tarefas selecionadas aleatoriamente (Figura 13). Esse mecanismo atua como uma medida de correção que impede a convergência prematura, ajudando a escapar de mínimos locais ao introduzir novas possibilidades de combinação entre tarefas.

Figura 13 – Exemplo de mutação

Filho 1	2	4	1	5	3
Filho 1 Mutado	2	1	4	5	3

Fonte: Autor, 2024

3.3.3.5 Critério de parada

Para este trabalho, o GA foi configurado para executar até um número máximo de gerações, definido através de um processo de calibração descrito na subseção 3.4.3.1. A quantidade foi determinada com base em testes preliminares, que analisaram o equilíbrio entre o tempo computacional e a qualidade das soluções obtidas. Durante as execuções, gráficos de convergência foram gerados para monitorar a evolução dos valores da função objetivo ao longo das gerações, permitindo avaliar o desempenho do algoritmo e a eficácia dos ajustes realizados nos parâmetros principais, como tamanho da população e taxa de mutação.

3.3.3.6 Definir próxima geração

Para definir a próxima população no Algoritmo Genético, é realizada uma seleção baseada no desempenho das soluções da geração atual. Nesse processo, as

soluções são avaliadas de acordo com a função objetivo que reflete a qualidade de cada indivíduo em relação ao problema sendo resolvido. Após a avaliação, as 50% melhores soluções são selecionadas para compor a nova população. Essa estratégia, conhecida como seleção elitista parcial, garante que as soluções de maior qualidade sejam preservadas para a próxima geração, evitando a perda de boas características.

3.3.4 GRASP

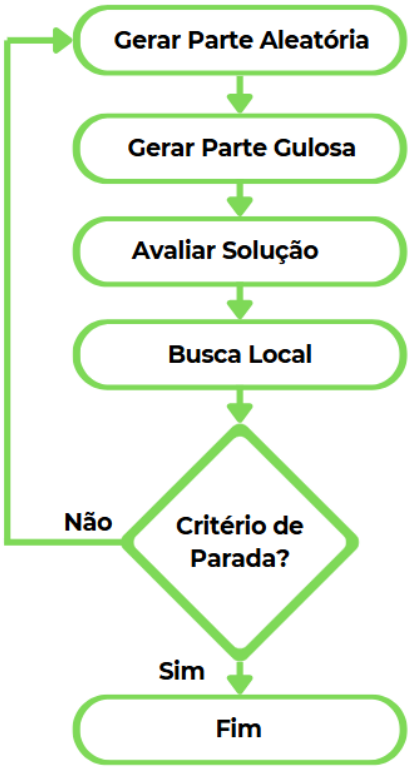
De acordo com Glover e Kochenberger (2003), o GRASP (*Greedy Randomized Adaptive Search Procedure*) é uma meta-heurística iterativa e de múltiplos inícios, já que reinicia sua busca diversas vezes ao longo do processo. Ele combina uma fase construtiva, baseada em heurísticas adaptativas semigulosas, com uma fase de refinamento, que emprega busca local para melhorar as soluções encontradas. Na fase de construção, o objetivo é gerar uma solução viável utilizando um critério de gulosidade adaptado, enquanto na fase de busca local, a solução gerada anteriormente é refinada para alcançar melhores resultados.

Glover e Kochenberger (2003) também destacam que as heurísticas puramente gulosas tendem a explorar de forma limitada o espaço de busca, o que pode levar a soluções subótimas. A combinação das duas fases no GRASP busca superar esses desafios ao balancear a diversificação das soluções com a qualidade das configurações geradas. Feo, Resende e Smith (1994) reforçam que a fase de construção do GRASP pode adotar diferentes estratégias, sendo a semigulosa uma das mais conhecidas. No problema de scheduling, por exemplo, a ordenação dos elementos com base em critérios de "maior valor"pode guiar a escolha, mas o GRASP introduz flexibilidade ao usar uma Lista Restrita de Candidatos (LRC). Essa lista, composta por um subconjunto dos melhores elementos de acordo com um critério adaptativo, restringe as opções disponíveis, permitindo uma combinação controlada entre gulosidade e aleatoriedade.

A LRC é definida como um dos parâmetros chave do GRASP e determina quantos candidatos farão parte dessa lista restrita, influenciando diretamente a qualidade e a diversidade das soluções geradas durante a fase inicial do algoritmo.

A Figura 14 apresenta os passos principais considerados na aplicação do GRASP neste trabalho. Esses passos são descritos em detalhes nas subseções seguintes e compõem os elementos essenciais da abordagem utilizada para o sequenciamento de tarefas.

Figura 14 – Etapas do GRASP



Fonte: Autor, 2024

3.3.4.1 Gerar parte aleatória

Na construção da solução inicial, primeiro é formada uma Lista Restrita de Candidatos (LRC), que seleciona aleatoriamente um conjunto de tarefas e atribuições de máquinas aleatórias (Figura 15).

Figura 15 – Exemplo de LRC do GRASP

Tarefas	1	4	5
Máquinas	1	2	1

Fonte: Autor, 2024

Em seguida, as tarefas da LRC são ordenadas de forma gulosa, priorizando aquelas com maior potencial para melhorar a função objetivo, como por exemplo, o menor tempo de processamento (Figura 16)

Figura 16 – Exemplo de LRC ordenada pelo tempo de processamento

Sequência	1	5	4
Tempo de Processamento	3	5	6

Fonte: Autor, 2024

3.3.4.2 Gerar parte gulosa

Após a definição da LRC, o restante da solução também é construído de maneira gulosa (Figura 17), onde cada nova tarefa é escolhida com base em sua contribuição para otimizar a função objetivo. Esse processo permite uma exploração variada e adaptativa das combinações de sequências possíveis.

Figura 17 – Exemplo solução inicial do GRASP

Sequência	1	5	4	3	2	FO = 60
Máquina	1	1	2	1	1	
Tempo de Processamento	3	5	6	2	4	

Fonte: Autor, 2024

3.3.4.3 Busca local

Após a construção inicial, o GRASP avança para uma fase de busca local, na qual é utilizada a heurística de primeira melhora para refinar a solução obtida. A estratégia de vizinhança adotada consiste em trocar a tarefa i com a tarefa $i+1$ (Figura 18) e escolher aleatoriamente a máquina $\mathbb{ß}$, reavaliando a função objetivo para cada solução vizinha gerada. Sempre que uma solução vizinha apresenta um valor de função objetivo inferior ao da solução atual, ela é imediatamente aceita e passa a ser o novo ponto de partida para as próximas iterações. Como na Figura 18 a primeira solução vizinha encontrada é melhor que a solução inicial, ela passa a ser a nova solução base para encontrar novas soluções vizinhas.

Figura 18 – Exemplo de vizinhança

Sequência solução inicial	1	5	4	3	2	FO=60
Designação de máquina solução inicial	1	1	2	1	1	
Sequência vizinho 1	5	1	4	3	2	FO=20
Designação de máquina vizinho 1	2	1	2	1	1	

Fonte: Autor, 2024

3.3.4.4 Critério de parada

Neste trabalho, o GRASP foi configurado para utilizar um critério de parada baseado na ausência de melhorias significativas após um número definido de iterações consecutivas, o limite foi calibrado e é apresentado em detalhes na subseção 3.4.3.2. Essa abordagem possibilita que o algoritmo explore o espaço de soluções de forma adaptativa, com os resultados intermediários sendo analisados em gráficos de convergência.

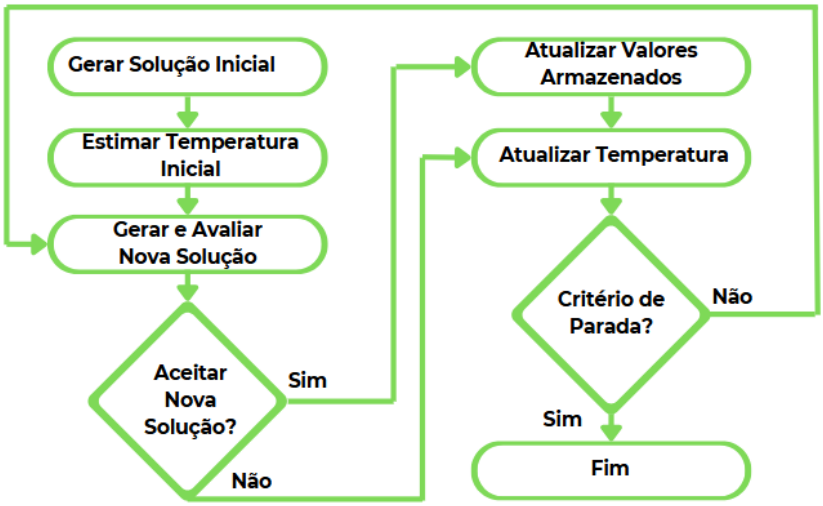
3.3.5 Simulated Annealing

O Simulated Annealing (SA) é inspirado no processo físico de recozimento (*annealing*), que é um fenômeno da termodinâmica utilizado para otimizar a estrutura cristalina de metais. Durante o recozimento, a temperatura do metal é elevada para permitir que as partículas se reorganizem, e depois é resfriado lentamente, permitindo que a estrutura do metal se ajuste ao estado de menor energia possível (Kirkpatrick, Gelatt e Vecchi, 1983).

Analogamente, no Simulated Annealing, a temperatura é o parâmetro central que controla a busca por soluções. Durante o processo de otimização, o algoritmo busca a melhor solução de um problema variando a temperatura, que inicialmente é alta e diminui progressivamente. As temperaturas altas permitem a exploração de soluções piores, o que é crucial para evitar que o algoritmo se prenda a ótimos locais (Goldbarg, Goldbarg e Luna, 2016). O processo se assemelha ao recozimento, pois, quando a temperatura está mais alta, há uma maior chance de aceitar soluções piores, possibilitando a diversificação e evitando que o algoritmo se prenda a soluções locais.

A Figura 19 apresenta os passos principais considerados na aplicação do Simulated Annealing (SA) neste trabalho. Esses passos são descritos em detalhes nas subseções seguintes e compõem os elementos essenciais da abordagem utilizada para o sequenciamento de tarefas.

Figura 19 – Etapas do Simulated Annealing



Fonte: Autor (2024)

3.3.5.1 Gerar solução inicial

A solução inicial é gerada por meio de uma heurística construtiva aleatória controlada, que assegura a construção de sequências factíveis ao evitar o sorteio repetido de uma mesma tarefa. Essa abordagem garante que todas as tarefas sejam alocadas apenas uma vez, respeitando as restrições do problema. Além disso, a designação de máquinas para cada tarefa é realizada de forma aleatória, promovendo maior diversidade inicial no espaço de soluções e contribuindo para uma exploração mais ampla durante as etapas subsequentes do algoritmo (Figura 20).

Figura 20 – Exemplo solução inicial do Simulated Annealing

Sequência	1	5	4	3	2
Máquina	1	1	2	1	1

Fonte: Autor, 2024

3.3.5.2 Estimar temperatura inicial

A temperatura inicial foi estimada com base nas diferenças entre os valores da função objetivo (FO) da solução inicial e de suas m soluções vizinhas geradas aleatoriamente, onde m é a quantidade de tarefas. Para isso, foi calculada a diferença média (Δ) entre essas funções objetivo e, utilizando a Equação (18), determinou-se a temperatura inicial (T_0) que assegurasse a aceitação de soluções subótimas com uma probabilidade pré-definida (P_0, geralmente 0,8 ou 0,9). A fórmula sugerida por Kirkpatrick, Gelatt e Vecchi (1983) é apresentada abaixo:

$$T_0 = \frac{-\Delta}{\ln(P_0)} \tag{18}$$

Essa abordagem permite calibrar adequadamente o algoritmo, garantindo que, nas primeiras iterações, haja uma exploração eficaz do espaço de soluções ao aceitar soluções piores de maneira controlada. Isso favorece a diversidade e evita a convergência prematura a ótimos locais.

3.3.5.3 Gerar e avaliar nova solução

Novas soluções são geradas utilizando dois tipos de vizinhança para a sequência das tarefas. A primeira realiza a troca entre as posições i e $i+1$, enquanto a segunda considera três elementos consecutivos, efetuando a troca entre i, $i+1$ e $i+2$. Essas estratégias promovem alterações estruturais na sequência, explorando diferentes combinações de ordem.

Além das alterações na sequência, a designação de máquinas também é ajustada em cada nova solução. Essa troca de designação é realizada de maneira randômica, garantindo uma diversificação adicional na busca por soluções. Este procedimento ocorre independentemente das mudanças de sequência, sendo uma etapa fixa do processo.

Após a geração, todas as novas soluções são avaliadas com base no cálculo da função objetivo (FO), conforme descrito na Fórmula 17 da Seção 3.3.3.2. Esse cálculo orienta o algoritmo na comparação da qualidade entre as soluções geradas e a solução atual.

3.3.5.4 Aceitar nova solução?

A decisão de aceitar a nova solução gerada é baseada em uma regra probabilística, que leva em consideração a diferença entre a função objetivo da solução atual (FO_{atual}) e da nova solução (FO_{nova}). Se a nova solução apresentar uma função objetivo melhor (ou seja, uma redução no valor da função objetivo no caso de um problema de minimização), ela será aceita automaticamente. No caso de uma solução pior, a aceitação dependerá de uma probabilidade P, que é dada pela equação 19.

$$P = \exp\left(\frac{-(FO_{nova} - FO_{atual})}{T}\right) \tag{19}$$

onde T é a temperatura atual, que decresce ao longo das iterações, conforme descrito na seção anterior. Isso significa que, à medida que a temperatura diminui, a probabilidade de aceitar soluções piores também diminui, permitindo ao algoritmo convergir para uma solução ótima ou próxima disso. Esse processo de aceitação de soluções subótimas, especialmente nas fases iniciais, é crucial para evitar a convergência prematura e para explorar uma maior diversidade no espaço de soluções.

3.3.5.5 Atualizar valores armazenados

Após a avaliação e possível aceitação da nova solução, o algoritmo atualiza os valores armazenados, substituindo a solução atual pela nova solução gerada, caso ela seja aceita. Isso inclui a atualização dos parâmetros relacionados à função objetivo e à solução de sequência de tarefas e máquinas. Esses valores são então utilizados na próxima iteração para calcular novas soluções vizinhas. Além disso, o algoritmo armazena a melhor solução encontrada até o momento, permitindo que ela seja recuperada ao final da execução, caso seja necessária para a análise de resultados.

3.3.5.6 Atualizar temperatura

A temperatura é atualizada a cada iteração do algoritmo, seguindo um processo de resfriamento controlado, que é fundamental para a convergência do Simulated Annealing. O resfriamento é realizado com base em uma taxa de resfriamento α, que determina a redução da temperatura a cada iteração. A atualização da temperatura é dada pela fórmula:

$$T = \alpha \cdot T \tag{20}$$

onde α é um valor entre 0 e 1, frequentemente escolhido entre 0,8 e 0,99. A taxa de resfriamento mais baixa resulta em uma exploração mais prolongada do espaço de soluções, enquanto uma taxa de resfriamento maior acelera a convergência, possivelmente em direção a ótimos locais.

Esse processo de resfriamento gradual permite que o algoritmo inicialmente explore mais amplamente o espaço de soluções, aceitando soluções subótimas, e posteriormente refine a busca, focando nas melhores soluções encontradas até o momento, evitando picos de temperatura e garantindo maior estabilidade na convergência.

3.3.6 Critério de parada

O critério de parada do algoritmo Simulated Annealing é atingido quando a temperatura chega ao valor de congelamento T_f (Kirkpatrick, Gelatt e Vecchi, 1983), o qual define o ponto em que o algoritmo deixa de aceitar novas soluções ou faz uma exploração suficientemente ampla do espaço de soluções. O valor de T_f pode ser determinado empiricamente com base em experimentos realizados para o problema específico. Assim, o critério de parada é:

$$T \leq T_f \tag{21}$$

Quando esse critério é atingido, o algoritmo encerra sua execução, retornando a melhor solução encontrada durante o processo. O valor de T_f deve ser suficientemente baixo para que o algoritmo tenha explorado adequadamente o espaço de soluções, mas não tão baixo que leve a uma convergência prematura.

3.4 PROCEDIMENTOS EXPERIMENTAIS

3.4.1 Definição das instâncias de teste

As instâncias de teste foram criadas para representar diferentes cenários de scheduling. Cada instância variou em termos de número de tarefas, tempo de processamento e prazos de entrega. Tanto para o scheduling em máquina única, quanto para o scheduling em máquinas em paralelo, os cenários foram divididos em três categorias principais, conforme apresentado na Tabela 4.

Tabela 4 – Instâncias de teste

Instância	Máquina Simples	Máquinas em Paralelo
Pequena	10 Tarefas	8 Tarefas e 2 Máquinas
Média	50 Tarefas	50 Tarefas e 2 Máquinas
Grande	100 Tarefas	100 Tarefas e 2 Máquinas

Fonte: Autor, 2024

O tamanho das instâncias pequenas foi definido experimentalmente, baseando-se na execução de diversos cenários. O maior cenário executado em máquinas paralelas envolveu 8 máquinas e 2 tarefas. No entanto, ao tentar cenários maiores, o processo levou até dois dias para rodar em um computador com processador intel core i5 de oitava geração, com 24 Gb de RAM e sistema operacional Windows 11, e não encontrou soluções factíveis. O mesmo ocorreu com o cenário de máquina simples. Por conta disso, optou-se por realizar a comparação com o resultado exato apenas para as instâncias pequenas. Já as instâncias médias e grandes foram comparadas exclusivamente entre as meta-heurísticas aplicadas.

3.4.2 Medidas de desempenho

A avaliação do desempenho dos algoritmos foi realizada com base nas seguintes métricas:

- *Tardiness Ponderado*: Soma dos atrasos de todas as tarefas, multiplicada pela multa de atraso de cada tarefa atrasada;
- *Earliness Ponderado*: Soma dos adiantamentos de todas as tarefa, multiplicada pela multa de adiantamento de cada tarefa adiantada;
- *Tempo computacional*: Tempo de execução do algoritmo.

3.4.3 Calibração de parâmetros

Para a análise de desempenho dos algoritmos foram realizados testes com diferentes instâncias e diferentes valores de parâmetros, com o objetivo de identificar

para quais valores de parâmetros os algoritmos encontram melhores soluções em média. Os dados de entrada foram definidos randomicamente, permitindo explorar uma variedade de cenários de teste.

3.4.3.1 Calibração dos parâmetros do Algoritmo Genético

Os parâmetros calibráveis do Algoritmo Genético (AG) para o problema de sequenciamento de tarefas em máquina simples e em máquinas paralelas foram definidos de forma sistemática para garantir o melhor desempenho. Para ambos os casos, foram analisados:

- **Corte:** Percentual da população a ser descartada antes de gerar uma nova população. Por exemplo, um corte de 0,1 significa eliminar os 10% piores indivíduos;
- **Taxa de mutação:** Probabilidade de uma solução sofrer mutação;
- **Máximo de iterações (*max*):** Número máximo de populações geradas antes de parar o algoritmo.

No caso de máquinas simples, a calibração foi realizada separadamente para instâncias pequenas, médias e grandes. Para cada cenário, foram executadas 100 bases de dados distintas, permitindo identificar os valores ideais dos parâmetros. Os resultados estão resumidos na Tabela 5.

Tabela 5 – Calibração do AG para Máquina Simples

Instância	FO	Tempo (s)	Corte	Taxa de Mutação	Max
Pequena	982	0,010	0,1	0,5	330
Média	190.879	1,010	0,7	0,6	350
Grande	830.092	2,736	0,4	0,5	380

Fonte: Autor, 2024

Para máquinas paralelas, a calibração seguiu o mesmo processo, com ajustes específicos para as características de cada instância. A análise foi realizada para instâncias pequenas, médias e grandes, conforme apresentado na Tabela 6.

Tabela 6 – Calibração do AG para Máquinas em Paralelo

Instância	FO	Tempo (s)	Corte	Taxa de Mutação	Max
Pequena	842	0,016	0,1	0,1	280
Média	41.129	1,256	0,7	0,6	340
Grande	179.257	5,322	0,8	0,8	410

Fonte: Autor, 2024

3.4.3.2 Calibração dos parâmetros do GRASP

Os parâmetros calibráveis do *Greedy Randomized Adaptive Search Procedure (GRASP)* foram definidos para garantir o desempenho ideal em máquinas simples e máquinas paralelas. Para ambos os casos, os parâmetros avaliados foram:

- **Tamanho da LRC:** Porcentagem da quantidade de tarefas a serem sequenciadas considerada na construção da solução inicial.
- **Max:** Número máximo de iterações consecutivas em que o algoritmo pode criar novas soluções sem obter uma melhor que a incumbente (critério de parada).

No caso de máquinas simples, a calibração foi realizada separadamente para instâncias pequenas, médias e grandes. Para cada cenário, foram executadas 100 bases de dados distintas, permitindo identificar os valores ideais dos parâmetros. Os valores calibrados estão apresentados na Tabela 7.

Tabela 7 – Calibração do GRASP para Máquina Simples

Instância	FO	Tempo (s)	LRC (% de Tarefas)	Max
Pequena	7.976	0,0158	20	10
Média	316.199	0,0055	30	10
Grande	1.597.102	0,0211	70	10

Fonte: Autor, 2024

Para máquinas paralelas, a calibração seguiu o mesmo processo, com ajustes específicos para cada tipo de instância. Os resultados são apresentados na Tabela 8.

Tabela 8 – Calibração do GRASP para Máquinas em Paralelo

Instância	FO	Tempo (s)	LRC (% de Tarefas)	Max
Pequena	2.108	0,0090	10	10
Média	88.368	4,8201	80	60
Grande	421.651	31,5518	90	80

Fonte: Autor, 2024

3.4.3.3 Calibração dos parâmetros do Simulated Annealing

Os parâmetros calibráveis do Simulated Annealing (SA) para o problema de sequenciamento de tarefas foram ajustados para alcançar o melhor desempenho tanto em máquinas simples quanto em máquinas paralelas. Os parâmetros avaliados foram:

- **Taxa de resfriamento:** Proporção pela qual a temperatura é reduzida a cada iteração.

- **Temperatura de congelamento:** Valor da temperatura que encerra o algoritmo.

No caso de máquinas simples, a calibração foi realizada separadamente para instâncias pequenas, médias e grandes. Os valores encontrados estão apresentados na Tabela 9.

Tabela 9 – Calibração do SA para Máquina Simples

Instância	FO	Tempo (s)	Taxa de Resfriamento	Temp. Congelamento
Pequena	3.972	0,0060	0,87	0,8
Média	305.453	0,0782	0,92	0,4
Grande	1.357.165	0,0899	0,89	0,9

Fonte: Autor, 2024

Para máquinas paralelas, os valores calibrados estão apresentados na Tabela 10.

Tabela 10 – Calibração do SA para Máquinas em Paralelo

Instância	FO	Tempo (s)	Taxa de Resfriamento	Temp. Congelamento
Pequena	1.564	0,0157	0,90	0,5
Média	106.939	0,0473	0,85	0,4
Grande	490.389	0,2454	0,91	0,6

Fonte: Autor, 2024

4 RESULTADOS

Nesta seção são apresentados os resultados obtidos com aplicação dos métodos desenvolvidos neste trabalho: PLIM, Algoritmo Genético, GRASP e Simulated Annealing na solução do problema de sequenciamento de tarefas em máquinas simples e em máquinas em paralelo. O desempenho de cada abordagem foi analisado em termos de função objetivo e tempo computacional.

4.1 FERRAMENTAS UTILIZADAS

Na realização deste trabalho foram utilizadas diversas ferramentas para aprimorar o armazenamento, processamento e interpretação dos dados, bem como para organizar a metodologia de pesquisa. O Excel foi utilizado para armazenar e organizar os dados de entrada, fornecendo uma estrutura clara e facilmente acessível para o gerenciamento inicial das informações do problema de sequenciamento. A linguagem de programação utilizada foi Python, desempenhando um papel central na manipulação dos dados, na resolução do problema de scheduling e na interpretação dos resultados, tanto de forma numérica quanto gráfica.

Os experimentos foram conduzidos em uma máquina com processador Intel Core i5 de oitava geração, 24GB de RAM, utilizando o sistema operacional Windows 11.

4.2 ANÁLISE DE CONVERGÊNCIA DOS MÉTODOS PROPOSTOS

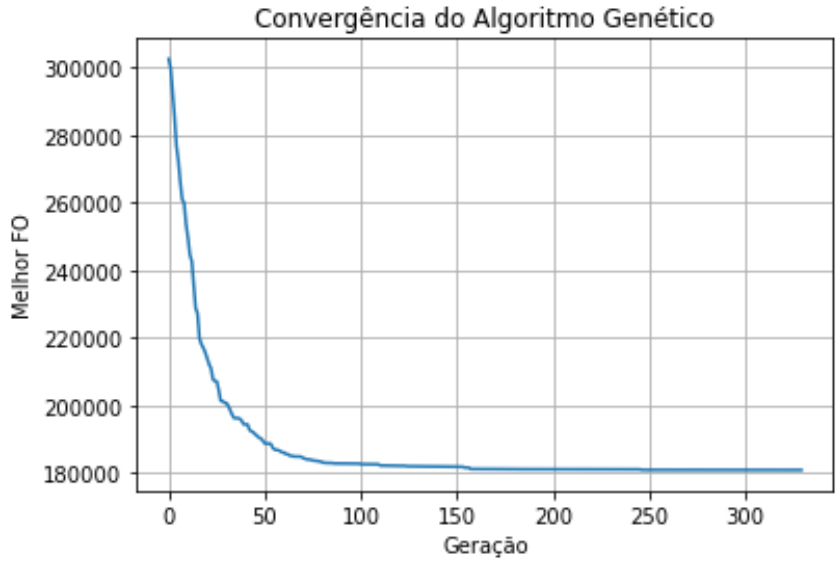
A análise de convergência dos métodos propostos é essencial para compreender o desempenho das abordagens implementadas ao longo das iterações. Esta seção apresenta os gráficos de convergência correspondentes aos métodos Algoritmo Genético, GRASP e Simulated Annealing, demonstrando a evolução das soluções em relação à função objetivo. A análise visa destacar a capacidade de cada método em explorar e intensificar o espaço de busca, garantindo a obtenção de soluções de alta qualidade de forma eficiente.

4.2.1 Convergência do Algoritmo Genético

O gráfico de convergência do Algoritmo Genético (Figura 21) ilustra o comportamento da função objetivo ao longo das gerações em uma instância média em máquina simples, permitindo analisar a evolução das soluções e a eficiência do método em encontrar resultados factíveis. No eixo horizontal, são representadas as gerações, enquanto no eixo vertical está o valor da função objetivo correspondente à melhor

solução de cada geração. Observa-se neste gráfico que o algoritmo explora bem o espaço de soluções até a metade das gerações, melhorando pouco após. Este foi o comportamento mais observado nas instâncias rodadas, o que sugere que o operador de mutação do algoritmo poderia ser alterado com o objetivo de melhorar a busca em outras regiões do espaço de soluções.

Figura 21 – Convergência do Algoritmo Genético - instância média - máquina simples

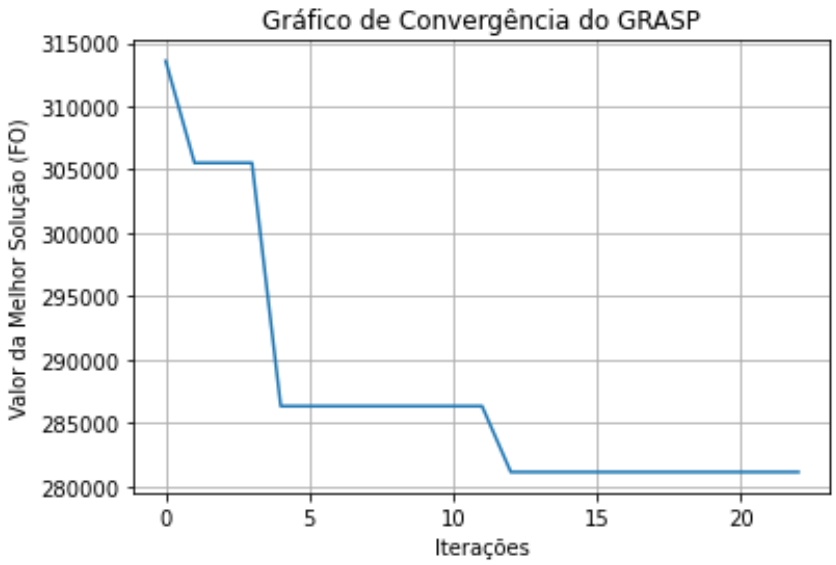


Fonte: Autor, 2024

4.2.2 Convergência do GRASP

O gráfico de convergência do GRASP (Greedy Randomized Adaptive Search Procedure) (Figura 22) ilustra a evolução das soluções ao longo das iterações em uma instância média em máquina simples, destacando como o método combina aleatoriedade e otimização local para refinar os resultados. Nas primeiras iterações, o algoritmo explora diferentes combinações iniciais geradas pela heurística construtiva, levando a variações significativas nos valores das soluções. À medida que as iterações avançam, as soluções tendem a se concentrar em valores mais próximos do ótimo, refletindo a eficácia do refinamento local. O comportamento do gráfico evidencia o equilíbrio entre a diversidade das soluções iniciais e a capacidade do procedimento de busca local de melhorar a qualidade dessas soluções.

Figura 22 – Convergência do GRASP - instância média - máquina simples



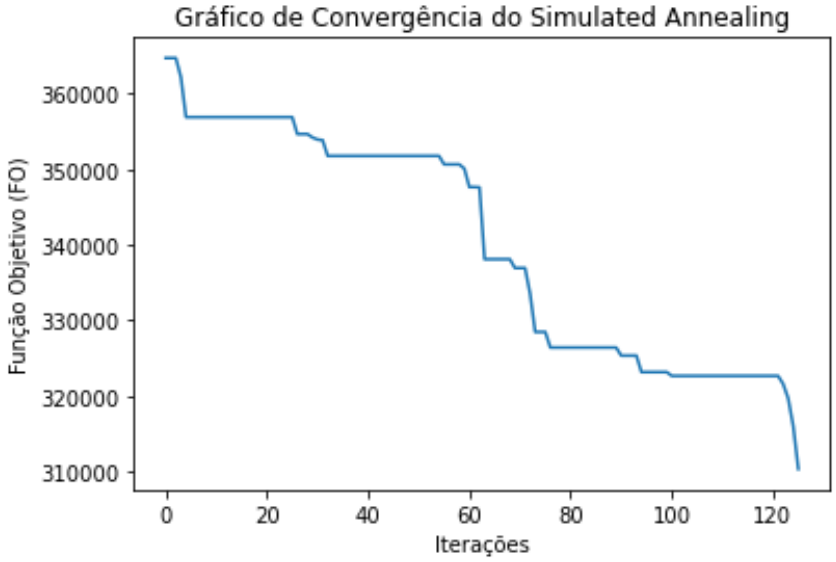
Fonte: Autor, 2024

4.2.3 Convergência do Simulated Annealing

O gráfico de convergência do Simulated Annealing (Figura 23) representa a variação da função objetivo ao longo das iterações para uma instância média em máquina simples, evidenciando o comportamento do algoritmo na busca por soluções de qualidade. No eixo horizontal, encontram-se as iterações realizadas, enquanto no eixo vertical está o valor da função objetivo correspondente à melhor solução encontrada até aquele ponto.

Esse gráfico é essencial para compreender como o método balanceia a intensificação e a diversificação do espaço de busca, aproveitando a aceitação controlada de soluções subótimas em altas temperaturas e a concentração em soluções de alta qualidade em temperaturas mais baixas. A análise da curva de convergência permite avaliar a eficiência do algoritmo em alcançar a estabilidade das soluções e evitar a estagnação em ótimos locais.

Figura 23 – Convergência do Simulated Annealing - instância média - máquina simples



Fonte: Autor, 2024

4.2.4 Comparação de resultados obtidos com as Meta-heurísticas

Para as instâncias pequenas, médias e grandes, tanto de máquina simples quanto de máquinas em paralelo, foram aplicadas as meta-heurísticas Algoritmo Genético (AG), GRASP e Simulated Annealing (SA), adicionalmente, para as instâncias pequenas foi possível aplicar o método PLIM. Encontram-se no Apêndice B as tabelas com os resultados de funções objetivo para cada instância, enquanto no Apêndice C as tabelas com os tempos computacionais.

4.2.4.1 Comparação para máquina simples

Para o caso de instâncias pequenas e máquina simples o PLIM obteve os melhores resultados de Função Objetivo (FO) em todas as instâncias, o que é esperado para um método exato, que encontra a solução ótima. No entanto, o tempo de execução do PLIM foi significativamente maior em comparação com as meta-heurísticas.

As meta-heurísticas, em especial o GRASP, demonstraram grande eficiência em termos de tempo de execução, com tempos médios de execução bem inferiores ao do PLIM. O GRASP foi, em média, 99,6% mais rápido que o PLIM nas instâncias pequenas, o que demonstra sua capacidade de fornecer soluções rápidas, embora não ótimas em todos os casos. O Algoritmo Genético (AG) também apresentou bons resultados, com tempos em média 70,16% mais rápidos que o PLIM, mas a qualidade das soluções foi ligeiramente inferior quando comparado ao GRASP. O Simulated Annealing (SA), por sua vez, teve o pior desempenho em relação à qualidade de FO, apresentando em média resultados 240% piores quando comparado ao PLIM.

Apresenta-se na Tabela 11 o desempenho das meta-heurísticas em relação ao PLIM, comparados com tempo computacional e qualidade de solução.

Tabela 11 – Comparação de resultados em relação ao PLIM - máquina simples

Métrica	AG	GRASP	SA
FO	38,4% Pior	150,5% Pior	240% Pior
Tempo computacional	70,16% Melhor	99,6% Melhor	99,13% Melhor

Fonte: Autor, 2024

No caso das instâncias médias o PLIM não foi aplicável, devido ao tempo computacional extremamente alto. Nesse sentido, apenas as meta-heurísticas foram aplicadas para esse cenário. O Algoritmo Genético alcançou os melhores valores médios de função objetivo, o GRASP teve um desempenho 67% pior quando comparado a função objetivo do AG, enquanto o Simulated Annealing teve o desempenho 82,25% pior, realizando a mesma comparação, conforme é apresentado na Tabela 12.

Tabela 12 – Comparação de resultados em relação ao AG - máquina simples

Métrica	GRASP	SA
FO	67% Pior	82,25% Pior
Tempo computacional	98,51% Melhor	95,87% Melhor

Fonte: Autor, 2024

Nas instâncias grandes o PLIM se mostrou inviável devido ao tempo computacional extremamente elevado, o que era esperado, dado que a busca exata se torna inviável para grandes instâncias. As meta-heurísticas, especialmente o Algoritmo Genético, dominaram as instâncias, fornecendo boas soluções em tempos muito mais rápidos. O Algoritmo Genético alcançou os melhores valores médios de função objetivo, o GRASP teve um desempenho 38,82% pior quando comparado a função objetivo do AG, enquanto o Simulated Annealing teve o desempenho 75,13% pior, realizando a mesma comparação, conforme é apresentado na Tabela 16.

Tabela 13 – Comparação de resultados em relação ao AG - máquina simples

Métrica	GRASP	SA
FO	38,82% Pior	75,13% Pior
Tempo computacional	98,56% Melhor	96,86% Melhor

Fonte: Autor, 2024

Os resultados demonstram que, para as instâncias pequenas, o PLIM foi o

método que encontrou a solução ótima, mas com um custo computacional muito alto. A meta-heurística Algoritmo Genético apresentou os melhores valores de função objetivo, quando comparada com os demais métodos aproximativos. Em contrapartida, as meta-heurísticas GRASP e SA foram muito mais rápidas, com o GRASP se destacando em termos de eficiência computacional, mas com uma qualidade de solução ligeiramente inferior em algumas instâncias. O Simulated Annealing (SA), devido à sua natureza estocástica, apresentou variações nos resultados, sendo a meta-heurística com os piores valores médios em termos de função objetivo.

Para as instâncias médias e grandes, o PLIM não foi uma opção viável, e as meta-heurísticas tiveram um desempenho muito superior em termos de tempo de execução, com o GRASP sendo, em média, o mais rápido, seguido pelo SA e o AG.

4.2.4.2 Comparação para máquinas em paralelo

Para o caso de instâncias pequenas e máquinas em paralelo o PLIM obteve os melhores resultados de Função Objetivo (FO) em todas as instâncias, o que é esperado para um método exato, que encontra a solução ótima. No entanto, o tempo de execução do PLIM foi significativamente maior em comparação com as meta-heurísticas.

As meta-heurísticas, em especial o GRASP, demonstraram grande eficiência em termos de tempo de execução, com tempos médios de execução bem inferiores ao do PLIM. O GRASP foi, em média, 99,9% mais rápido que o PLIM nas instâncias pequenas, o que demonstra sua capacidade de fornecer soluções rápidas, embora não ótimas em todos os casos. O Algoritmo Genético (AG) também apresentou bons resultados, com tempos em média 99,98% mais rápidos que o PLIM, mas a qualidade das soluções foi ligeiramente inferior quando comparado ao GRASP. O Simulated Annealing (SA), por sua vez, teve o pior desempenho em relação à qualidade de FO, apresentando em média resultados 230% piores quando comparado ao PLIM. Apresenta-se na Tabela 14 o desempenho das meta-heurísticas em relação ao PLIM, comparados com tempo computacional e qualidade de solução.

Tabela 14 – Comparação de resultados em relação ao PLIM - máquinas em paralelo

Métrica	AG	GRASP	SA
FO	0,076% Pior	164,7% Pior	230,37% Pior
Tempo computacional	99,98% Melhor	99,99% Melhor	99,99% Melhor

Fonte: Autor, 2024

No caso das instâncias médias o PLIM não foi aplicável, devido ao tempo computacional extremamente alto. Nesse sentido, apenas as meta-heurísticas foram aplicadas para esse cenário. O Algoritmo Genético alcançou os melhores valores

médios de função objetivo, o GRASP teve um desempenho 171,10% pior quando comparado a função objetivo do AG, enquanto o Simulated Annealing teve o desempenho 247,27% pior, realizando a mesma comparação, conforme é apresentado na Tabela 15.

Tabela 15 – Comparação de resultados em relação ao AG - máquinas em paralelo

Métrica	GRASP	SA
FO	171,10% Pior	247,27% Pior
Tempo computacional	70,33% Melhor	95,57% Melhor

Fonte: Autor, 2024

Nas instâncias grandes o PLIM se mostrou inviável devido ao tempo computacional extremamente elevado, o que era esperado, dado que a busca exata se torna inviável para grandes instâncias. As meta-heurísticas, especialmente o Algoritmo Genético, dominaram as instâncias, fornecendo boas soluções em tempos muito mais rápidos. O Algoritmo Genético alcançou os melhores valores médios de função objetivo, o GRASP teve um desempenho 142,69% pior quando comparado a função objetivo do AG, enquanto o Simulated Annealing teve o desempenho 177,81% pior, realizando a mesma comparação, conforme é apresentado na Tabela 16.

Tabela 16 – Comparação de resultados em relação ao AG - máquinas em paralelo

Métrica	GRASP	SA
FO	142,69% Pior	177,81% Pior
Tempo computacional	24,75% Melhor	96,62% Melhor

Fonte: Autor, 2024

Os resultados mostram que, para as instâncias pequenas, o PLIM foi o método que encontrou a solução ótima, mas com um tempo de processamento muito alto. A meta-heurística Algoritmo Genético apresentou os melhores valores de função objetivo, quando comparada com os demais métodos aproximativos. Em contrapartida, as meta-heurísticas GRASP e SA foram muito mais rápidas, com o SA se destacando em termos de eficiência computacional, mas com uma qualidade de solução inferior.

Para as instâncias médias e grandes, o PLIM não foi uma opção viável, e as meta-heurísticas apresentaram um desempenho muito superior em termos de tempo de execução, com o SA sendo, em média, o mais rápido, seguido pelo GRASP e o AG.

Em termos gerais, o PLIM apresentou o melhor desempenho em termos de qualidade da solução (FO), mas o tempo de execução foi um grande limitante, especialmente para instâncias maiores. As meta-heurísticas AG, GRASP e SA

forneceram soluções razoáveis em tempos muito menores, com o GRASP e o SA se destacando pelas suas velocidades, especialmente para instâncias grandes e médias. No entanto, a qualidade da solução para instâncias pequenas foi consistentemente melhor com o PLIM, embora o custo computacional tenha sido alto.

Esses resultados indicam que, para problemas de sequenciamento de tarefas, o PLIM é adequado para instâncias pequenas, mas não escalável para instâncias maiores. As meta-heurísticas são a melhor escolha quando é necessário equilibrar entre qualidade e eficiência computacional.

4.3 CONSIDERAÇÕES FINAIS SOBRE OS RESULTADOS

Os resultados obtidos demonstram que as meta-heurísticas Algoritmo Genético, GRASP e Simulated Annealing são eficientes para resolver o problema de sequenciamento de tarefas, proporcionando soluções de boa qualidade com tempos de execução muito menores que o método de PLIM. Embora a PLIM seja capaz de encontrar a solução ótima, o tempo de computacional de execução do algoritmo torna-se inviável sua aplicação para instâncias médias e grandes. As meta-heurísticas são, portanto, uma alternativa eficaz quando se busca um bom compromisso entre qualidade da solução e tempo de computação.

5 CONCLUSÕES

Neste trabalho, foi abordado o problema de sequenciamento de tarefas, um desafio central na área de Planejamento e Controle da Produção (PPC). Trata-se de um problema classificado como NP-difícil, dado seu alto nível de complexidade computacional, especialmente à medida que o tamanho das instâncias cresce. Essa característica torna essencial o desenvolvimento e a aplicação de métodos eficientes para encontrar soluções de qualidade em tempos viáveis, atendendo às demandas de cenários reais.

O objetivo foi investigar a aplicação de meta-heurísticas Algoritmo Genético, GRASP e Simulated Annealing no problema de sequenciamento de tarefas. O foco foi minimizar as penalidades por adiantamento e por atraso e avaliar os tempos computacionais dos métodos mencionados.

A partir das simulações realizadas, observou-se que as meta-heurísticas foram eficazes na obtenção de soluções factíveis em tempos reduzidos, quando comparadas ao modelo exato. O Algoritmo Genético apresentou uma solução superior na maioria dos casos, enquanto o GRASP foi mais eficiente em termos de tempo computacional. A abordagem exata, embora mais precisa, demandou um tempo computacional significativamente maior para as instâncias pequenas, e não encontrou soluções factíveis para os demais tamanhos de instâncias.

O estudo contribui para a área de otimização de problemas de sequenciamento, ao demonstrar a aplicabilidade de diferentes meta-heurísticas no contexto de tarefas com prazos e múltiplas restrições.

O modelo de Programação Linear Inteira Mista (PLIM) foi implementado e aplicado a instâncias pequenas do problema. Os resultados obtidos mostram que o PLIM é capaz de encontrar soluções ótimas para essas instâncias em tempos computacionalmente viáveis, servindo como referência para a validação das soluções geradas pelos algoritmos meta-heurísticos.

O Algoritmo Genético foi desenvolvido com base nos princípios de seleção, cruzamento e mutação propostos por Holland (1992). Após implementação, o algoritmo mostrou-se eficaz na busca por soluções de alta qualidade, destacando-se pela sua capacidade de diversificar a busca e explorar o espaço de soluções.

O método GRASP foi implementado utilizando uma construção gulosa adaptativa e uma fase de busca local para refinar as soluções. A abordagem mostrou-se robusta e eficiente, especialmente para instâncias de tamanho médio, onde apresentou boa convergência para soluções de alta qualidade em tempos reduzidos.

O Simulated Annealing foi desenvolvido inspirado no modelo de resfriamento

físico de Kirkpatrick (1983), implementando uma estratégia de aceitação probabilística de soluções para evitar ótimos locais. A meta-heurística apresentou um desempenho inferior quando comparada com as demais, principalmente quando comparado com a qualidade da solução

Os métodos desenvolvidos foram comparados quanto à qualidade das soluções e ao tempo computacional. A análise revelou que, embora o Algoritmo Genético tenha apresentado uma leve superioridade em qualidade de solução, o GRASP destacou-se pelo equilíbrio entre qualidade e tempo computacional..

Uma limitação importante deste estudo foi a utilização de dados aleatórios, que, embora representativos, não refletem completamente a complexidade dos dados do mundo real. Além disso, a abordagem exata, embora forneça soluções ótimas, não é escalável para grandes instâncias do problema devido ao seu alto custo computacional.

Futuras pesquisas poderiam focar na adaptação das meta-heurísticas para incluir novas variáveis, como diferentes tipos de restrições ou a consideração de custos variáveis ao longo do tempo e quantidades maiores de máquinas. Além disso, a análise de diferentes estratégias de resfriamento no Simulated Annealing e a combinação de algoritmos híbridos poderiam melhorar ainda mais os resultados obtidos. Uma abordagem interessante seria também testar as meta-heurísticas com dados reais provenientes de empresas de produção. Outrossim, estudos futuros poderiam aplicar diferentes regras heurísticas para o sequenciamento de tarefas, como o EDD explicado no referencial teórico.

Em síntese, pode-se dizer que, para os dados considerados, os resultados obtidos sugerem que as meta-heurísticas podem ser uma solução viável e eficaz para problemas de sequenciamento de tarefas em ambientes de produção, onde a flexibilidade e a adaptabilidade são essenciais. A aplicação desses métodos permite uma melhoria significativa no desempenho operacional, contribuindo para a redução de custos e aumento da eficiência.

REFERÊNCIAS

ABDEL-BASSET, M.; ABDEL-FATAH, L.; SANGAIAH, A. K. Metaheuristic algorithms: A comprehensive review. In: **Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications**. [S.l.: s.n.], 2018. p. 185–231.

ANDRIAN, F. R. Planejamento como ferramenta essencial na gestão e liderança. **Revista Científica da FAEX**, v. 12, n. 1, p. 260–281, jan/abr 2023.

ARENALES, M. et al. **Pesquisa operacional para cursos de engenharia**. 1. ed. Rio de Janeiro, RJ: Editora Campus, 2011.

BAKER, K. R.; TRIETSCH, D. **Principles of Sequencing and Scheduling**. [S.l.]: John Wiley & Sons, 2009.

BARBOSA, L. B. et al. Modelo para sequenciamento de tarefas em máquinas paralelas heterogêneas de injeção plástica com prazo de entrega. In: **ENEGEP 2021 - Encontro Nacional de Engenharia de Produção**. Online: [s.n.], 2021.

BIERWIRTH, C.; MEISEL, F. A survey of berth allocation and quay crane scheduling problems in container terminals. **European Journal of Operational Research**, v. 202, n. 3, p. 615–627, 2010.

BOUKEDROUN, M. et al. A hybrid genetic algorithm for stochastic job-shop scheduling problems. **RAIRO - Operations Research**, v. 57, n. 4, p. 1617–1645, jun 2023.

BRANKE, J. et al. Automated design of production scheduling heuristics: A review. **IEEE Transactions on Evolutionary Computation**, v. 20, n. 1, p. 110–124, 2015.

CAPES. **Quem somos**. Brasília, DF, 2020. Disponível em: https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/index.php/sobre/quem-somos.html. Acesso em: 16 jun. 2024.

CHER, R. **A gerência da pequena e média empresa**. São Paulo: Maltese, 1990.

CORRÊA, H. L.; GIANESI, I. G. N.; CAON, M. **Planejamento, programação e controle da produção: MRP II/ERP : conceitos, uso e implantação: base para SAP, oracle applications e outros softwares integrados de gestão**. 5. ed.. ed. São Paulo: Atlas, 2007. 411 p. ISBN 9788522448531.

FANJUL-PEYRO, L.; RUIZ, R.; PEREA, F. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. **Comput. Oper. Res.**, v. 101, p. 173–182, 2019.

FEO, T. A.; RESENDE, M. G. C.; SMITH, S. H. A greedy randomized adaptive search procedure for maximum independent set. **Operations Research**, INFORMS, v. 42, n. 5, p. 860–878, 1994. ISSN 0030364X, 15265463. Disponível em: http://www.jstor.org/stable/171545.

GLOVER, F.; KOCHENBERGER, G. A. **Handbook of Metaheuristics**. 1. ed. New York: Kluwer Academic Publishers, 2003.

GOLDBARG, M. C.; GOLDBARG, E. G.; LUNA, H. P. L. **Otimização combinatória e meta-heurísticas: algoritmos e aplicações**. 1. ed. Rio de Janeiro: Elsevier, 2016. ISBN 978-85-352-7812-5.

HOLLAND, J. H. Genetic algorithms. **Scientific American**, Scientific American, a division of Nature America, Inc., v. 267, n. 1, p. 66–73, 1992. ISSN 00368733, 19467087. Disponível em: http://www.jstor.org/stable/24939139.

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. ISSN 00368075, 10959203. Disponível em: http://www.jstor.org/stable/1690046.

LEUNG, J. Y.-T. **Handbook of Scheduling: Algorithms, Models, and Performance Analysis**. [S.l.]: CRC Press, 2004.

MINTZBERG, H.; AHLSTRAND, B.; LAMPEL, J. **Safari de estratégia: um roteiro pela selva do planejamento estratégico**. Porto Alegre: Bookman, 2000.

PAGE, M. J. et al. The prisma 2020 statement: An updated guideline for reporting systematic reviews. **PLoS Medicine**, v. 18, n. 3, p. e1003583, 2021. Published simultaneously in BMJ, Int J Surg, J Clin Epidemiol, and Syst Rev.

PINEDO, M. **Scheduling**: theory, algorithms, and systems. 5. ed. New York, NY, USA: Springer, 2016.

RAHMATIKA, D.; WIBOWO, F. T.; RIYADI, A. Job shop scheduling problem: Literature review. **ResearchGate**, 2020. Accessed: 2024-06-17. Disponível em: https://www.researchgate.net/publication/343826343_job_shop_scheduling_problemliterature_review.

ROBBINS, S. P.; COULTER, M. **Administração**. 14. ed. Rio de Janeiro: Prentice-Hall do Brasil, 1988.

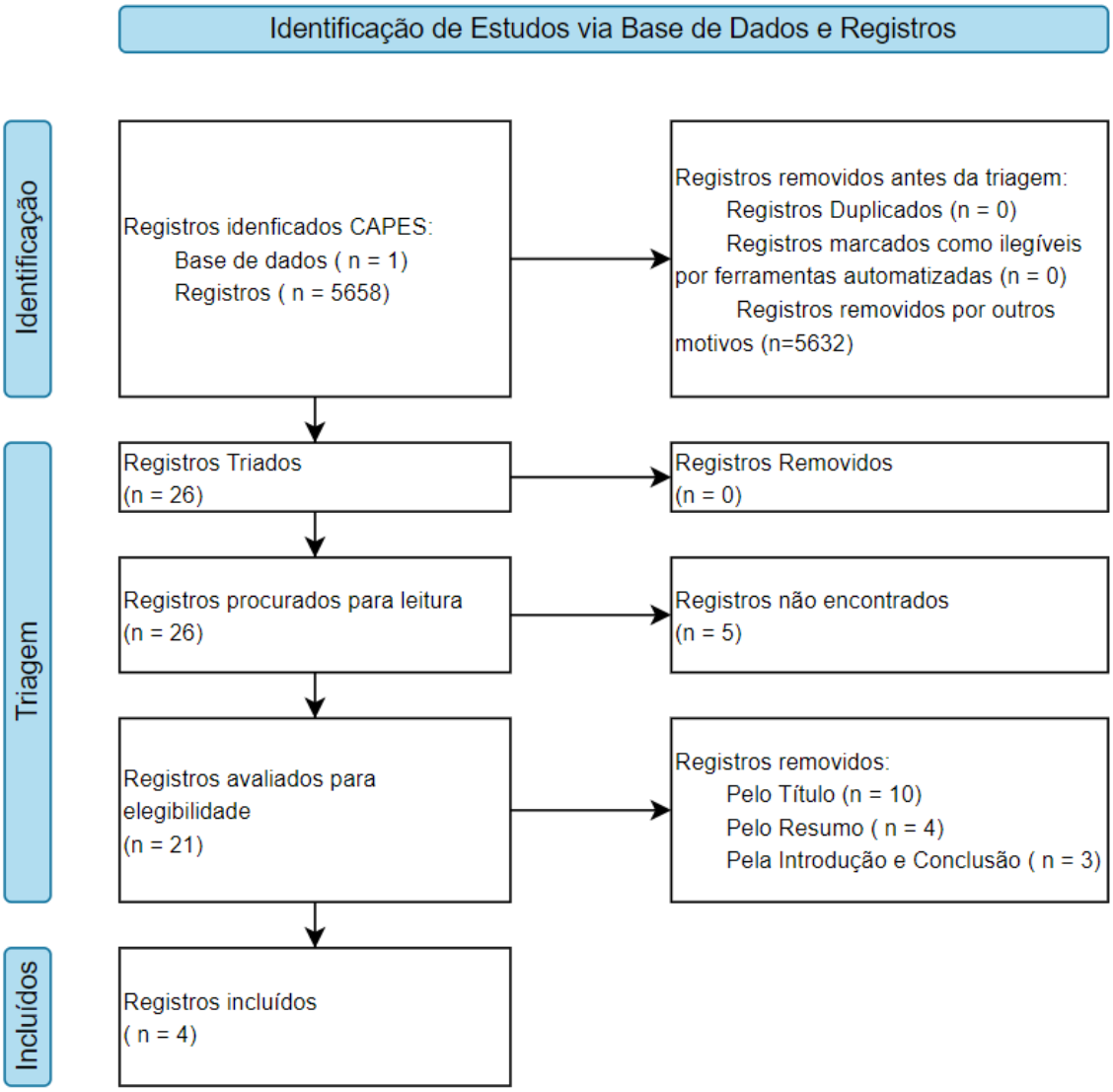
SÖRENSEN, K.; GLOVER, F. Metaheuristics. In: **Encyclopedia of Operations Research and Management Science**. [S.l.: s.n.], 2013. v. 62, p. 960–970.

TAHA, H. A.; ZHANG, C. **Pesquisa Operacional**. 6. ed.. ed. São Paulo: Pearson, 2014. ISBN 9788576051507.

TAILLARD, Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, p. 278–285, 1993.

WU, C.-C. et al. A two-agent one-machine multitasking scheduling problem solving by exact and metaheuristics. **Complex and Intelligent Systems**, v. 8, n. 1, p. 199–212, fev 2022.

APÊNDICE A - APLICAÇÃO DA METODOLOGIA PRISMA



APÊNDICE B - TABELAS DE RESULTADOS DA APLICAÇÕES DAS META-HEURÍSTICAS - VALORES DE FO'S

Tabela 17 – FO para Instâncias Pequenas - Máquina Simples

Instância	Exato	AG	GRASP	SA
1	2559	3591	5282	10805
2	3376	4140	8433	12601
3	1639	2027	3820	3891
4	2503	4215	7973	13180
5	3185	3971	7148	9523
6	2363	2832	11191	7652
7	2001	3224	4972	8025
8	2469	4278	5415	8880
9	2460	2986	8260	11978
10	2565	3937	5165	6126
11	1058	2140	5055	6844
12	2543	3017	6267	7222
13	3766	5064	10019	11953
14	1768	2261	3109	7808
15	2813	5112	8461	8528
16	2263	3136	3790	10493
17	1614	2064	5235	7101
18	7074	9552	9942	13517
19	2831	3894	10863	12613
20	4854	5416	10132	8711
21	4085	5511	8227	10877
22	4040	5147	6884	6300
23	1206	1511	4347	8009
24	2487	2744	7467	6373
25	1793	2091	3840	5734
26	4347	5460	9597	15900
27	2179	4314	4972	6768
28	1892	3567	5224	10331
29	1101	1274	4941	4080
30	686	1611	3173	8553
Médias	2650	3669	6640	9012

Fonte: Autor, 2024

Tabela 18 – FO para Instâncias Médias - Máquina Simples

Instância	AG	GRASP	SA
1	166860	280408	371768
2	170924	312599	324903
3	234770	375878	402303
4	191095	295064	287186
5	198037	291042	333027
6	180958	277066	337698
7	192815	331269	351253
8	139943	268565	265771
9	198087	347502	380424
10	139573	265329	277434
11	209459	326404	359642
12	141296	242234	259621
13	204116	353533	382702
14	226214	340757	364176
15	161605	286553	344523
16	160760	308446	301345
17	189340	285652	321832
18	166150	276502	321993
19	195590	295923	354373
20	149659	254521	285929
21	167848	294209	312589
22	212189	344133	386078
23	1572341	332733	321350
24	1569412	318456	345342
25	1662439	328352	335672
26	1547397	315643	321930
27	1623149	315014	332544
28	1634459	325345	336798
29	1572341	331900	344572
30	1612323	337672	350431
Médias	181883	303745	331488

Fonte: Autor, 2024

Tabela 19 – FO para Instâncias Grandes - Máquina Simples

Instância	AG	GRASP	SA
1	1411305	573091	602160
2	1508837	584553	693820
3	1437286	663810	700335
4	1314351	620993	761546
5	1398947	621922	631929
6	1384984	539418	675298
7	1547927	592923	653998
8	1500505	639258	750584
9	1446923	572670	645166
10	1336225	592673	654131
11	1445047	626319	749716
12	1303607	551530	568260
13	1324100	583127	671341
14	1491060	547093	670810
15	1620000	557227	614346
16	1378713	573169	634379
17	1317035	606565	678063
18	1529391	560594	637191
19	1325314	589928	650854
20	1498049	544213	749795
21	1628018	556225	623966
22	1246344	601577	718530
23	1303435	726657	737529
24	1160793	565121	690859
25	1569412	634046	701538
26	1375132	679776	756216
27	1451966	641827	792733
28	1330939	627257	676851
29	1572341	486745	565884
30	1380995	515217	690171
Médias	809628	1123922	1417966

Fonte: Autor, 2024

Tabela 20 – FO para Instâncias Pequenas - Máquina em Paralelo

Instância	Exato	AG	GRASP	SA
1	701	705	2492	3846
2	1012	1015	3157	6907
3	694	698	2649	2757
4	1140	1144	4017	3734
5	1991	1994	3108	3364
6	1248	1249	3466	4908
7	937	941	2429	2965
8	2910	2913	4525	8125
9	1055	1058	3845	5301
10	1227	1231	3047	2264
11	616	616	2294	3903
12	1187	1189	2829	2222
13	1183	1186	3656	3752
14	1985	1986	5553	6836
15	1098	1102	1611	5282
16	918	918	3347	4369
17	1661	1664	2857	3919
18	1458	1462	4471	6075
19	711	714	4290	4244
20	1057	1060	2822	2162
21	1157	1160	4546	3797
22	2171	2174	3718	4382
23	1074	1076	4486	5166
24	1106	1110	2261	4120
25	2860	2865	4070	4269
26	1754	1758	4306	4673
27	791	793	3647	4280
28	833	838	3225	6007
29	718	722	1913	2425
30	1294	1298	3572	1502
Médias	1286	1287	3406	4251

Fonte: Autor, 2024

Tabela 21 – FO para Instâncias Médias - Máquinas em Paralelo

Instância	AG	GRASP	SA
1	64289	157421	186818
2	22005	95077	106785
3	36309	130236	163443
4	56353	138543	163280
5	46105	107988	171998
6	42465	131449	132926
7	41106	113262	149053
8	34172	99591	99480
9	58612	127979	178402
10	59702	121178	179561
11	42905	114162	172178
12	43408	133027	126993
13	42976	122069	133318
14	29823	94406	137445
15	44211	121760	138834
16	51603	121127	172329
17	43898	102524	140228
18	52380	128964	163900
19	33950	97805	122796
20	37164	114080	164045
21	47203	107503	161871
22	38184	129225	164706
23	30891	94111	103877
24	41011	112734	136561
25	42984	112588	134403
26	48192	120248	162986
27	51813	153708	215506
28	35305	103860	176480
29	48472	117421	128173
30	38796	117405	147962
Médias	43542	118048	151211

Fonte: Autor, 2024

Tabela 22 – FO para Instâncias Grandes - Máquinas em Paralelo

Instância	AG	GRASP	SA
1	194456	573091	602160
2	262176	584553	693820
3	278286	663810	700335
4	272587	620993	761546
5	218155	621922	631929
6	216823	539418	675298
7	185627	592923	653998
8	249168	639258	750584
9	246849	572670	645166
10	246669	592673	654131
11	285461	626319	749716
12	211159	551530	568260
13	267811	583127	671341
14	239622	547093	670810
15	204036	557227	614346
16	246091	573169	634379
17	286342	606565	678063
18	224477	560594	637191
19	213662	589928	650854
20	218240	544213	749795
21	225462	556225	623966
22	286171	601577	718530
23	278313	726657	737529
24	252034	565121	690859
25	294522	634046	701538
26	264627	679776	756216
27	284364	641827	792733
28	258079	627257	676851
29	188632	486745	565884
30	224369	515217	690171
Médias	244142	592517	678266

Fonte: Autor, 2024

APÊNDICE C - TABELAS DE RESULTADOS DAS APLICAÇÕES DAS META-HEURÍSTICAS - TEMPOS COMPUTACIONAIS

Tabela 23 – Tempos computacionais para Instâncias Pequenas - Máquina Simples

Instância	Exato	AG	GRASP	SA
1	0,2119	0,0754	0,002	0,0036
2	0,2456	0,0832	0	0,003
3	0,2393	0,0938	0,002	0,002
4	0,2788	0,0803	0,0016	0,0029
5	0,3252	0,0834	0,002	0
6	0,1641	0,0848	0,001	0,002
7	0,2215	0,0936	0,002	0,001
8	0,3906	0,0806	0,002	0,0035
9	0,2308	0,0773	0,001	0,0035
10	0,6077	0,0859	0,0011	0,003
11	0,2998	0,0898	0,0015	0,003
12	0,2237	0,0835	0,002	0,003
13	0,1728	0,0852	0,001	0,0035
14	0,6726	0,0879	0,001	0,002
15	0,2187	0,0853	0,001	0,003
16	0,2213	0,0927	0,0009	0,003
17	0,2245	0,1021	0,001	0,001
18	0,3402	0,0838	0,002	0,002
19	0,2058	0,0936	0,001	0,003
20	0,1616	0,0888	0,001	0,002
21	0,3643	0,0877	0,001	0,0025
22	0,9961	0,0867	0,001	0,004
23	0,2136	0,0814	0,001	0,002
24	0,1562	0,0814	0	0,0027
25	0,1953	0,0873	0,001	0,0015
26	0,1562	0,0908	0,001	0,002
27	0,3444	0,0757	0,002	0,0011
28	0,2137	0,0747	0,001	0,003
29	0,188	0,083	0,001	0,0035
30	0,1533	0,0979	0,001	0,002
Médias	0,2879	0,08591	0,001234	0,002487

Fonte: Autor, 2024

Tabela 24 – Tempos computacionais para Instâncias Médias - Máquina Simples

Instância	AG	GRASP	SA
1	1,0893	0,0126	0
2	1,0541	0,0085	0,1063
3	0,845	0,0116	0,003
4	1,0385	0,023	0,0909
5	1,1302	0,026	0,0085
6	0,9413	0,0166	0,0264
7	1,0101	0,0093	0,0633
8	1,2265	0,0196	0,0873
9	0,8349	0,0106	0,0465
10	1,4581	0,0155	0,0587
11	0,7695	0,0082	0,0222
12	0,963	0,0107	0,0135
13	1,0107	0,0116	0,005
14	1,2475	0,0237	0,0882
15	0,7801	0,01	0,0161
16	0,9104	0,0217	0,0045
17	1,2138	0,0169	0,0157
18	1,3987	0,0242	0,001
19	0,9373	0,0095	0,0388
20	0,9469	0,0121	0,0203
21	0,7926	0,0161	0,0819
22	0,8923	0,0095	0,0748
23	0,955	0,011	0,006
24	0,7505	0,0191	0,0915
25	1,1662	0,0125	0,055
26	0,8002	0,0136	0,0283
27	0,9445	0,0226	0,0085
28	1,0762	0,0213	0
29	0,9176	0,0096	0,0835
30	0,9004	0,0095	0,0927
Médias	1,0000	0,01489	0,04128

Fonte: Autor, 2024

Tabela 25 – Tempos computacionais para Instâncias Grandes - Máquina Simples

Instância	AG	GRASP	SA
1	5,1994	0,0429	0,0287
2	3,6792	0,044	0,2581
3	3,6134	0,0384	0,1237
4	3,6034	0,1348	0,0121
5	4,004	0,048	0,018
6	3,9638	0,0749	0
7	3,7702	0,0413	0,0227
8	3,5621	0,1172	0,1053
9	3,6219	0,0386	0,0115
10	3,5531	0,0382	0,2074
11	4,6923	0,0833	0,0227
12	4,6309	0,0806	0,2208
13	4,2951	0,064	0,1796
14	4,48	0,068	0,2465
15	3,516	0,0409	0,1634
16	4,3941	0,0671	0,2052
17	4,9009	0,0479	0,1675
18	3,3384	0,0682	0,0969
19	3,6009	0,0412	0,0656
20	4,1948	0,0354	0,0802
21	3,4405	0,0401	0,1649
22	3,6384	0,0395	0,2337
23	4,5728	0,0628	0,1444
24	4,364	0,0459	0,1627
25	4,2544	0,0701	0,2099
26	3,6028	0,034	0,1438
27	3,4345	0,0432	0,0065
28	4,3347	0,0891	0,1826
29	4,4204	0,0409	0,0756
30	3,8588	0,0489	0,2179
Médias	4,0178	0,05764	0,1259

Fonte: Autor, 2024

Tabela 26 – Tempos computacionais para Instâncias Pequenas - Máquina em Paralelo

Instância	Exato	AG	GRASP	SA
1	215,0905	0,0745	0,0009	0,0025
2	120,2401	0,0734	0,002	0,003
3	118,856	0,0747	0	0,001
4	90,6932	0,0725	0,001	0,002
5	1394,4283	0,0894	0,001	0,001
6	2168,9902	0,0738	0,001	0,0045
7	66,7722	0,0738	0,001	0,003
8	87,0927	0,0776	0,0005	0,003
9	883,7623	0,0828	0	0,001
10	35,1065	0,0738	0,001	0,0035
11	454,1333	0,0755	0,001	0,002
12	19,2096	0,0793	0,001	0,002
13	75,5055	0,0702	0,001	0,0035
14	323,8363	0,0744	0,001	0,0015
15	110,9858	0,07	0,001	0,001
16	300,0855	0,0731	0,001	0,003
17	5000,6993	0,0771	0,001	0,001
18	365,5619	0,0702	0,0009	0,0026
19	1486,7269	0,0776	0,001	0,0025
20	140,168	0,0687	0,001	0,003
21	12,531	0,0688	0,001	0,001
22	48,3462	0,0693	0,001	0,0029
23	106,0508	0,0703	0	0,0035
24	44,6853	0,0783	0	0,001
25	96,7191	0,0699	0,001	0,002
26	829,1624	0,0749	0,001	0,003
27	37,7171	0,0689	0,0015	0,003
28	218,9701	0,0876	0	0,002
29	810,8971	0,0717	0,0035	0,0026
30	496,9231	0,0723	0,0011	0,003
Médias	538,6648	0,07448	0,0009509	0,002361

Fonte: Autor, 2024

Tabela 27 – Tempos computacionais para Instâncias Médias - Máquinas em Paralelo

Instância	AG	GRASP	SA
1	0,9956	0,0275	0,0532
2	1,2625	0,746	0,0826
3	1,4756	0,5213	0,0752
4	0,8716	0,2938	0,0698
5	1,2056	0,6466	0,0383
6	1,2345	0,3231	0,05
7	1,24	0,2463	0,0826
8	0,9526	0,5148	0,0752
9	0,9379	0,543	0,0202
10	0,9652	0,2591	0,077
11	1,0775	0,0771	0,0631
12	1,0439	0,3625	0,0237
13	0,9595	0,2544	0,0206
14	1,5149	0,2272	0,0076
15	0,9938	0,0845	0,0406
16	1,198	0,369	0,0268
17	0,9783	0,3605	0,0305
18	1,5166	0,2379	0,066
19	1,2676	0,296	0,0165
20	1,2459	0,2995	0,0075
21	1,2388	0,1499	0,0792
22	1,1296	0,3605	0,019
23	1,2394	0,1998	0,0823
24	0,8899	0,384	0,0243
25	0,6981	0,1359	0,1045
26	0,7565	0,1944	0,088
27	1,0968	0,6233	0,0391
28	1,1729	0,6788	0,0218
29	1,2925	0,2684	0,0897
30	1,5052	0,3879	0,0298
Médias	1,1318	0,3357	0,05015

Fonte: Autor, 2024

Tabela 28 – Tempos computacionais para Instâncias Grandes - Máquinas em Paralelo

Instância	AG	GRASP	SA
1	5,2815	3,7772	0,2468
2	4,3902	3,9203	0,2296
3	4,9562	1,2244	0,0733
4	5,322	1,9025	0,2697
5	4,7475	2,3812	0,2445
6	5,7119	5,0079	0,0772
7	2,9536	5,6456	0,1097
8	4,8788	4,8165	0,1641
9	5,9188	1,1436	0,3024
10	3,6137	0,6397	0,3206
11	4,569	6,9271	0,0448
12	4,2247	3,9248	0,1383
13	5,4456	4,3201	0,2282
14	4,7222	4,1941	0,0247
15	5,1835	3,2883	0,1382
16	5,542	2,2037	0,2257
17	5,4203	3,2917	0,3152
18	4,5958	5,1422	0,133
19	4,258	5,1952	0,0281
20	3,6691	3,1758	0,0135
21	3,9082	2,2456	0,2189
22	5,5831	2,6627	0,2896
23	4,6425	3,0812	0,0494
24	3,8752	2,5502	0,1183
25	5,9679	2,4441	0,1837
26	7,5497	7,3115	0,0202
27	4,8787	3,6849	0,1057
28	4,1985	2,1488	0,1839
29	2,2353	2,8289	0,1308
30	3,4087	5,503	0,1505
Médias	4,7217	3,5527	0,1592

Fonte: Autor, 2024