Sequenciamento de tarefas com tempos de preparação dependentes da sequência em máquinas paralelas idênticas utilizando algoritmo genético

Alisson Michel Sganzerla[1], Guilherme Lopes Weis[1], Eugênio Antônio Zanini[1]
[1]*Universidade Federal de Santa Maria, Santa Maria, Brazil*
alisson.sganzerla@gmail.com, glweis11@gmail.com.br, genio.zanini@gmail.com

Keywords: Makespan, Genetic Algorithm, Job Scheduling, Metaheuristic.

Abstract: Este estudo comparou três estratégias para gerar a população inicial de um algoritmo genético aplicado ao problema de sequenciamento de tarefas em máquinas paralelas idênticas, tendo em conta o tempo de preparação dependente da sequência. Todas elas geravam uma população inicial aleatória, porém na segunda estratégia incluía-se um indivíduo gerado por um método construtivo guloso e na terceira, um indivíduo gerado com a solução encontrada na literatura. A codificação dos indivíduos foi realizada através da técnica *List Scheduling*(LS), agrupando os índices das tarefas alocadas nas máquinas em ordem crescente em uma única lista. Os cromossomos foram gerados utilizando a técnica de corte *one-point crossover*, dividindo o material genético em dois segmentos e cruzando-os entre os pais para gerar novos indivíduos. Além disso, foram utilizadas técnicas de reparação de genes para garantir a viabilidade dos indivíduos gerados. A decodificação do cromossomo gera um novo indivíduo, distribuindo as tarefas na sequência apresentada com uma heurística gulosa conforme a regra *Longest Process Time*(LPT), na máquina menos carregada naquele instante e na melhor posição dentro da máquina. O processo de mutação dos indivíduos utilizou uma estratégia de vizinhança de busca local composta por *troca* e *inserção* em um pequeno percentual da população. O algoritmo genético foi executado 100 vezes por 10 gerações para todas as instâncias, em cada uma das três estratégias propostas. Os resultados obtidos mostraram que o valor médio para o Cmax foi apenas 3,31% acima dos melhores resultados relatados na literatura para as duas primeiras abordagens, e em 4 das 30 instâncias testadas, foram encontrados resultados melhores do que os relatados na literatura existente. No Tópico 4, comparamos os resultados obtidos com o valor médio e mínimo de cada abordagem, com um método construtivo guloso, uma busca local e o resultado da literatura existente.

This study compared three strategies for generating the initial population of a genetic algorithm applied to the problem of task sequencing on identical parallel machines, taking into account the sequence-dependent preparation time. All of them generated a random initial population, however in the second strategy an individual generated by a greedy constructive method was included, and in the third, an individual generated with the solution found in the literature. The individuals' coding was carried out using the List Scheduling technique (LS), grouping the indices of the tasks allocated to machines in increasing order in a single list. The chromosomes were generated using the one-point crossover cut technique, dividing the genetic material into two segments and crossing them between parents to generate new individuals. In addition, gene repair techniques were used to ensure the viability of the generated individuals. Chromosome decoding generates a new individual, distributing tasks in the sequence presented with a greedy heuristic according to the Longest Processing Time (LPT) rule, on the least loaded machine at that time and in the best position within the machine. The mutation process of individuals used a local search neighborhood strategy composed of swap and insertion in a small percentage of the population. The genetic algorithm was run 100 times for 10 generations for all instances, in each of the three proposed strategies. The results showed that the average value for Cmax was only 3.31% above the best results reported in the literature for the first two approaches, and in 4 of the 30 tested instances, results were found to be better than those reported in existing literature. In Topic 4, we compared the results obtained with the average and minimum value of each approach, with a greedy constructive method, a local search, and the result of existing literature.

1 INTRODUÇÃO

A estrutura organizacional utilizada para a tomada de decisões em um ambiente empresarial é geralmente dividida em três níveis: estratégico, tático e operacional. O nível estratégico está relacionado aos objetivos de longo prazo, o nível tático está relacionado aos objetivos de médio prazo e o nível operacional está relacionado à entrega de resultados de curto prazo, de acordo com Slack et al. (2008). Este estudo se concentra na tomada de decisão no nível operacional, que requer respostas precisas e eficientes em curtos períodos de tempo. Devido à sua complexidade, essas decisões frequentemente necessitam de suporte computacional. Encontrar uma solução ótima para problemas complexos pode, em alguns casos, exigir um tempo de processamento significativo através do emprego de um método exato. Como alternativa, utiliza-se técnicas heurísticas, que podem fornecer resultados de alta qualidade em tempos de processamento aceitáveis, dependendo das instâncias do problema.

2 REVISÃO TEÓRICA

A Teoria da complexidade computacional, introduzida por Hartmanis and Stearns (1965) em *On the Computational Complexity of Algorithms*, conceitua uma medida de complexidade como uma função do tempo computacional usando uma máquina de *Turing*. O sequenciamento de tarefas em máquinas paralelas podem ser classificados como problemas de complexidade NP-difícil dependendo do modelo específico e das restrições, o que significa que, em teoria, eles podem ser resolvidos em tempo polinomial com um algoritmo exato. No entanto, atualmente não há um método exato conhecido capaz de resolver o problema de maneira eficiente. Em vez disso, são utilizadas técnicas heurísticas e metaheurísticas para encontrar soluções aproximadas em tempos computacionais razoáveis. Algoritmos desse tipo geralmente produzem bons resultados dentro de um tempo de computação aceitável, mas não garantem a otimalidade da solução. Para Pearl (1984), uma heurística é um conjunto de critérios, métodos ou princípios usados para decidir quais alternativas de ação levam a um resultado mais eficiente.

As heurísticas podem ser classificadas em diferentes categorias, como as construtivas e as de busca, ambas são utilizadas neste estudo. Os métodos construtivos geram soluções iniciais factíveis para um problema específico, cuja qualidade varia de acordo com a abordagem utilizada. Já os métodos de busca procuram melhorar essas soluções iniciais, explorando mais o espaço de busca. Porém, as soluções obtidas após a etapa de melhoria podem ficar limitadas a uma região com ótimos locais, impedindo o acesso a soluções melhores fora dela. Para ampliar a busca por soluções em áreas não exploradas, utilizam-se métodos metaheurísticos, como o algoritmo genético. De acordo com Sörensen et al. (2018), metaheurísticas podem ser definidas como um conjunto consistente de ideias, conceitos e operadores que podem ser usados para projetar algoritmos de otimização heurística, elas são genéricas e independentes do problema, podendo ser aplicadas a uma ampla variedade de problemas de otimização.

O presente estudo se concentra no problema de sequenciamento de tarefas distribuídas em máquinas paralelas, sendo definido *makespan* ou *Cmax* como a métrica de desempenho para avaliar a eficiência das soluções. O objetivo é minimizar o tempo total de conclusão das tarefas. Graham (1966) desenvolveu um método conhecido como *LS (List Scheduling)*, onde uma lista de tarefas não ordenadas é distribuída nas máquinas de acordo com a sua carga. Esse método foi posteriormente melhorado pelo próprio Graham 1969 com a adição de uma ordenação prévia não crescente, denominada *LPT (Longest Processing Time)*, na qual a tarefa com o maior tempo de processamento é selecionada e alocada na máquina com menor carga. A partir desse estudo inicial, variações do algoritmo LS são frequentemente encontradas em adaptações para problemas em máquinas paralelas uniformes e não relacionadas. No presente trabalho, utilizou-se tanto o conceito de LS quanto o de LPT para a codificação de cada indivíduo e para implementação do método construtivo guloso, respectivamente.

O Algoritmo Genético Holland (1975) é baseado em processos evolutivos biológicos, onde populações de indivíduos representando soluções candidatas a um problema são submetidas a operações de seleção, reprodução e mutação para gerar novas gerações com características melhoradas. Na seleção, um dos métodos usados é o da roleta, que seleciona indivíduos para a reprodução baseado em sua aptidão, garantindo que os indivíduos mais aptos tenham maior probabilidade de serem escolhidos. Na reprodução, os cromossomos dos pais são divididos em pontos aleatórios, dependendo da técnica usada, como a *one-point crossover*, este método é conhecido por ser eficiente em manter a diversidade genética e promover a exploração do espaço de busca, cruzando os dois segmentos dos pais nos filhos.

3 METODOLOGIA

Foram utilizadas 30 instâncias de dados geradas aleatoriamente, que representam dois tipos distintos de problemas: instâncias estruturadas e não estruturadas. Essas instâncias foram escolhidas para representar uma variedade de dificuldades e foram utilizadas para avaliar a eficiência e a eficácia do algoritmo proposto. A configuração de cada grupo de instâncias, incluindo o número de tarefas e máquinas, pode ser observada na Tabela 1.

Tabela 1: Categoria e Configuração das Instâncias

	Struct	Instances	M	N
1	Y	5	2	10
2	Y	5	2	100
3	Y	5	5	100
4	N	5	2	10
5	N	5	2	100
6	N	5	5	100

Fonte: Autores (2023)

A Seção 3.1 apresenta todas as fórmulas utilizadas ao longo do trabalho. A Seção 3.2 descreve o algoritmo construtivo guloso que foi implementado para gerar soluções viáveis para cada instância do problema. A seção 3.3 apresenta três técnicas de busca local, incluindo variações de três vizinhanças, que foram testadas: troca, inserção e inserção generalizada, com diferentes configurações em relação à quantidade e ordem das operações. A seção 3.4 detalha a construção do algoritmo genético, incorporando o método guloso e a busca local mais eficiente. A seção 3.5 explica como foram realizados os testes comparativos entre todas as técnicas implementadas.

3.1 Fórmulas

Na presente seção, todas as fórmulas descritas ao longo do trabalho foram agrupadas com o objetivo de proporcionar uma melhor organização e facilidade de consulta ao leitor.

$$f(i) = \frac{\frac{1}{mak(i)}}{\sum_{j=1}^{p}(\frac{1}{mak(i)})} \tag{1}$$

$$f(pop) = 5 + (\frac{n}{m}) * 0.5 \tag{2}$$

$$f(mut) = 1 + (pop * 0.1) \tag{3}$$

$$f(gap) = \frac{C(Lit) - C(GA)}{C(GA} . - 100 \tag{4}$$

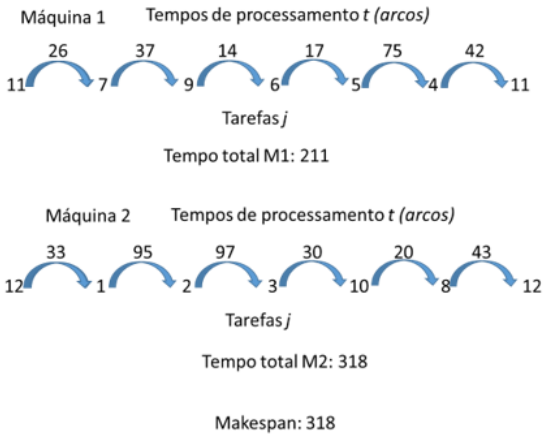
3.2 Heurística construtiva gulosa

O método construtivo guloso é implementado com o objetivo de maximizar a utilização das máquinas e minimizar o tempo total de processamento. Ele segue dois passos principais:

- Ordenação inicial das tarefas: utilizando a heurística LPT, as tarefas são ordenadas de acordo com o seu tempo de processamento em ordem não crescente.
- Alocação das máquinas: seguindo o algoritmo LS, as tarefas ordenadas no passo anterior são distribuídas nas máquinas. Na escolha da posição da tarefa dentro da máquina, consideram-se as tarefas já alocadas nela, buscando-se a melhor posição dentro das máquinas.

Na Figura 1, apresentamos uma solução inicial obtida pelo método construtivo, para uma instância com duas máquinas e dez tarefas. Ela ilustra o sequenciamento das tarefas, os tempos de processamento de cada tarefa e o tempo total de processamento das máquinas. Os nós representam as tarefas distribuídas e as setas indicam o fluxo de execução dessas tarefas. A junção de dois nós indica o tempo de processamento da tarefa, que é a soma dos tempos de preparação da máquina e de execução da tarefa.

Figura 1: Sequenciamento das tarefas em máquinas

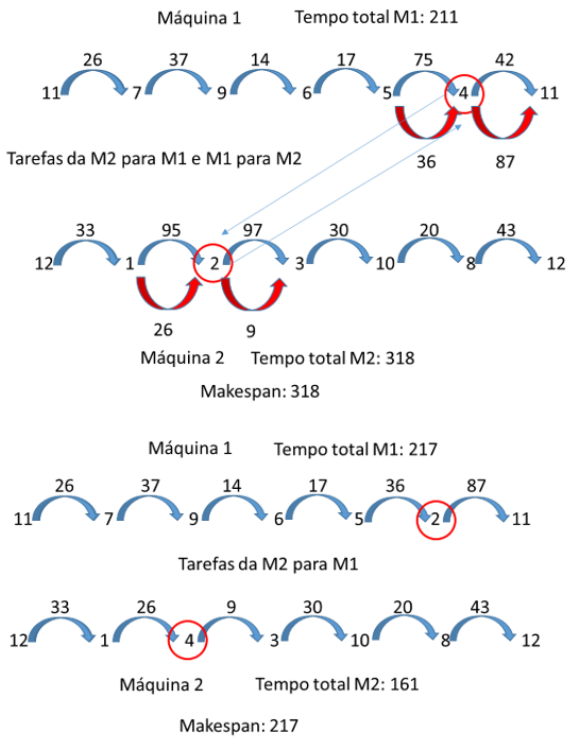


Fonte: Autores (2023)

3.3 Configuração de vizinhanças da busca local

Durante a etapa de melhoramento da solução inicial, busca-se encontrar uma nova combinação de tarefas que reduza o tempo total de processamento da máquina mais carregada, utilizando técnicas de vizinhança do tipo inter e intra-rota. A vizinhança inter-rota envolve a manipulação de tarefas entre duas ou mais máquinas, enquanto a vizinhança intra-rota se concentra em realizar mudanças apenas na própria máquina. Neste trabalho, as vizinhanças testadas foram troca (*swap*), inserção (*insertion*) e inserção generalizada (*generalized insertion*). Elas estão detalhadas abaixo:

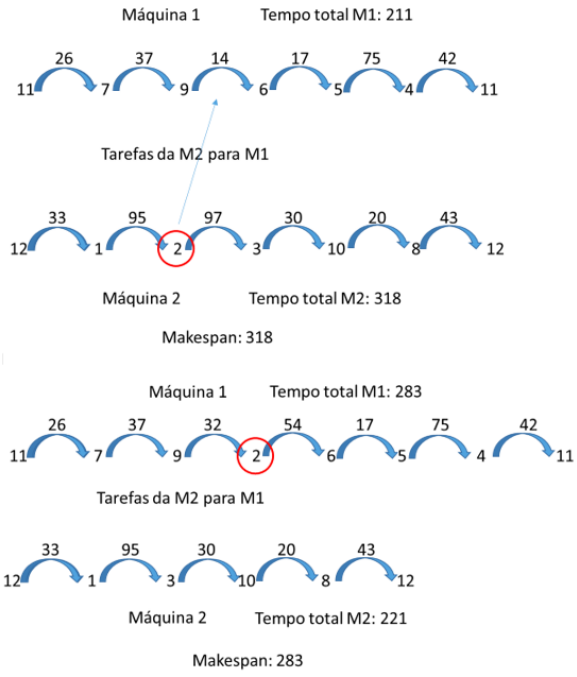
Figura 2: Vizinhança *swap*



Fonte: Autores (2023)

- *Swap*: é uma técnica de vizinhança do tipo inter-rota, que busca trocar tarefas entre máquinas para reduzir o tempo total de processamento da máquina mais carregada. Isso é feito avaliando qual troca seria mais benéfica para alcançar esse objetivo. Para ilustrar a execução dessa vizinhança. A Figura 2 exibe o melhor movimento encontrado entre as possibilidades, que é a troca entre as tarefas 4 e 2 nas máquinas 1 e 2, resultando em uma redução do tempo total de 318 para 217.
- *Insertion*: é uma técnica do tipo inter-rota, que busca a remoção de uma tarefa da máquina mais carregada e repassá-la para outra, com o objetivo de reduzir o tempo total. A Figura 3 ilustra um exemplo dessa técnica, mostrando o melhor movimento encontrado, que é a remoção da tarefa 2 na máquina 2 e inserindo-a na máquina 1, resultando em uma redução do tempo da máquina mais carregada de 318 para 283.
- *Generalized insertion*: é uma técnica do tipo intra-rota, que busca realizar um rearranjo entre 3 tarefas quaisquer dentro da própria máquina, desde que essa máquina possua pelo menos 3 tarefas. A Figura 4 exibe um exemplo dessa técnica, mostrando a troca entre três tarefas (2-3-8) na máquina 2, resultando na configuração (8-2-3) e reduzindo o Cmax de 318 para 297.

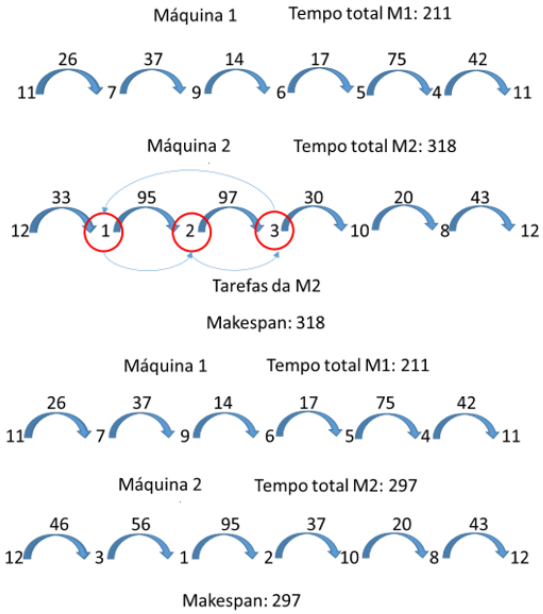
Figura 3: Vizinhança *insertion*



Fonte: Autores (2023)

A Tabela 2 descreve as configurações testadas, incluindo a composição e ordenação das operações de vizinhança e os resultados obtidos. Entre as configurações avaliadas, a configuração *BL2* se destacou como a mais eficiente, conseguindo melhorar a maioria das instâncias em menor tempo comparado às demais configurações.

Figura 4: Vizinhança *generalized insertion*



Fonte: Autores (2023)

Tabela 2: Desempenho e Composição da Busca Local

Name	Neighborhood	Time	Better
BL1	iii, ii, i, ii, i, iii	30.81%	43.33%
BL2	i, ii, iii, i, ii, iii	28.57%	63.33%
BL3	iii, i, iii, ii, iii, ii, iii, i	40.62%	50.00 %

Fonte: Autores (2023)

Em outro teste realizado, as estratégias de vizinhança foram comparadas de forma isolada, com o objetivo de identificar qual delas apresentava maior contribuição para o tempo de processamento. Os resultados mostraram que a estratégia *(iii)* representava mais de 99% do tempo de processamento, conforme apresentado na Tabela 3. Diante desse resultado, foi decidido ajustar a estratégia de mutação do algoritmo genético para utilizar a configuração *BL2*, excluindo a vizinhança *(iii)*, com o objetivo de aumentar a eficiência e performance do algoritmo, mesmo que essa vizinhança apresentasse melhor desempenho em relação à quantidade de instâncias melhoradas.

Tabela 3: Desempenho Vizinhanças Isoladas

Name	Neighborhood	Time	Better
i	i	0.29%	3.33%
ii	ii	0.37%	26.57%
iii	iii	99.34%	46.67%

Fonte: Autores (2023)

3.4 Algoritmo Genético

O Algoritmo Genético é iniciado com uma população inicial viável e um número específico de gerações como parâmetros de entrada. Ele é composto por cinco etapas que se repetem a cada geração, conforme ilustrado no fluxograma da Figura 5. Essas etapas incluem: a atualização da população, a avaliação e a substituição da população, a seleção de pais, a reprodução e a mutação. O tamanho da população inicial para cada grupo de instâncias da Tabela 1 é determinado pela Equação 2. Ele é calculado combinando o número mínimo de indivíduos para qualquer categoria de instância, que no caso é 5, com metade da razão entre a quantidade de tarefas e o número de máquinas. Desta forma, para a configuração das instâncias avaliadas neste trabalho, a população ficou 7, 30 e 15 indivíduos, respectivamente, para as instâncias (2x10), (2x100) e (5x100).

3.4.1 Atualização da População

Nesta etapa os filhos gerados na etapa de reprodução são adicionados à população corrente dos pais, formando assim a população atualizada de indivíduos.

3.4.2 Cálculo da Aptidão

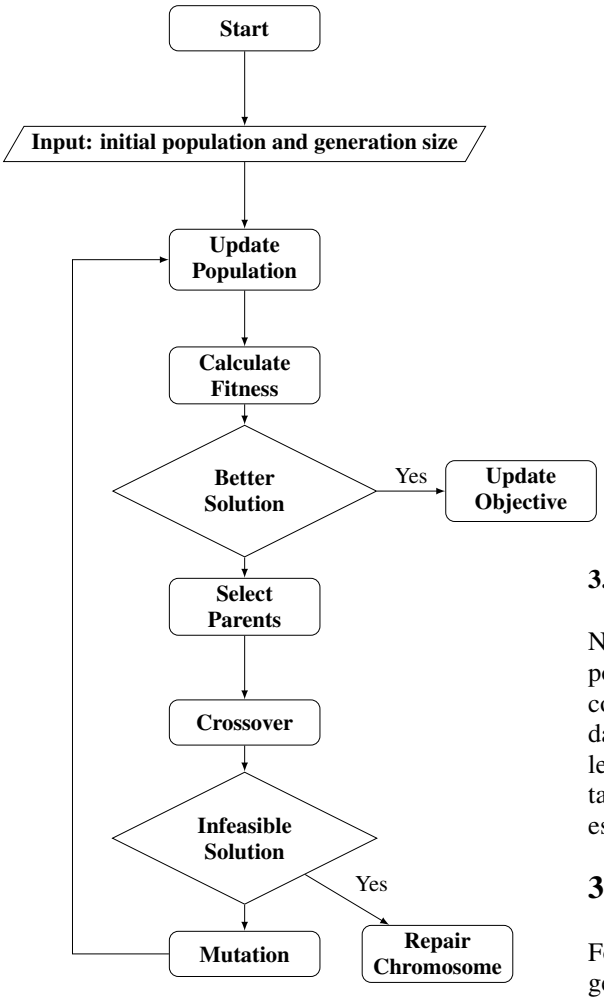
Os indivíduos são avaliados com base no valor de Cmax e aqueles com valores menores são eliminados, mantendo somente os mais aptos. Caso seja encontrado um indivíduo com desempenho superior à função objetivo atual, essa função objetivo é atualizada. Cada indivíduo recebe uma pontuação de aptidão proporcionalmente inversa à sua contribuição para a soma de todos os valores de Cmax da população, conforme a Equação (1).

3.4.3 Seleção de Pais

É utilizado um método de seleção por roleta para selecionar indivíduos com uma taxa de probabilidade proporcional ao seu valor de aptidão. Isso tende a favorecer indivíduos com melhores desempenhos, aumentando as chances de que esses indivíduos sejam escolhidos como pais para a próxima geração, aumentando assim a chance de que as características desejadas sejam passadas para a próxima geração.

Nesta etapa, os indivíduos são codificados como a sequência de tarefas alocadas em cada máquina, conforme ilustrado na Figura 6, em ordem crescente (LS). O método *one point crossover* é utilizado para combinar os cromossomos dos pais, cortando-os em um ponto aleatório único, excluindo-se as duas extremidades. Os segmentos dos pais são recombina-

Figura 5: Fluxograma Algoritmo Genético



Fonte: Autores (2023)

dos para gerar dois novos indivíduos, os filhos. Se esses filhos tiverem tarefas repetidas, uma técnica de reparação é aplicada para corrigir o cromossomo e torná-lo viável. Uma heurística gulosa é aplicada para decodificar cada cromossomo e gerar um novo indivíduo, distribuindo as tarefas na ordem em que o cromossomo é gerado e alocando-as na máquina com menor carga, na melhor posição possível, como apresentado na Seção 3.2.

Figura 6: Representação dos Cromossomos

Fonte: Autores (2023)

3.4.4 Mutação

Nesta etapa, a fim de introduzir variação genética na população, aplicou-se o método de busca local BL2, como apresentado na Figura 2, mas sem a utilização da vizinhança *(iii)*. A quantidade de indivíduos selecionados para sofrer mutação é proporcional ao tamanho da população conforme a Equação 3, já a escolha dos indivíduos é feita de forma aleatória.

3.5 Testando População Inicial

Foram conduzidos 300 testes com o algoritmo genético, utilizando três abordagens distintas: (a) aleatória, (b) gulosa e (c) baseada na literatura, cada uma das quais foi executada 100 vezes em cada uma das 30 instâncias disponíveis, evoluindo-as 10 gerações. As três abordagens são detalhadas abaixo:

- Aleatória: indivíduos da população inicial foram gerados de forma aleatória.
- Gulosa: os indivíduos da população inicial foram gerados aleatoriamente, exceto um, gerado pelo método construtivo guloso.
- Literatura: os indivíduos da população inicial foram gerados aleatoriamente, exceto um, obtido de acordo com a configuração apresentada pela literatura *(BKS-Best Known Solution)* Muller (1993).

A análise dos resultados dos testes das diferentes abordagens foi realizada utilizando a diferença percentual em relação à solução ideal conhecida (BKS), conforme mostrado na Equação 4. Essa métrica é crucial para avaliar a eficácia do algoritmo em comparação com a solução existente, pois permite

medir a distância entre a solução encontrada pelo algoritmo e a solução ideal da literatura. Os resultados são detalhadamente descritos no tópico 4, onde são apresentadas tabelas comparativas para facilitar a interpretação dos dados, garantindo uma compreensão mais clara e precisa dos resultados obtidos.

4 RESULTADOS

Os resultados das três abordagens experimentadas são apresentados detalhadamente nas Tabelas 6, 7 e 8, ilustrando o desempenho da função objetivo média e mínima das abordagens comparadas ao método construtivo guloso, busca local *BL2* e valor de referência na literatura. O cabeçalho da tabela apresenta as técnicas na ordem: método construtivo *C(Greedy)*, busca local *C(BL2)*, média do algoritmo genético *GA(Mean)*, valor mínimo do algoritmo genético *GA(Min)* e o valor da literatura *C(BKS)*. As quatro últimas colunas mostram o percentual em relação ao valor da literatura. O tempo médio de execução de cada instância é apresentado na Tabela 4.

Tabela 4: Tempo Médio Execução Instância

	M	**N**	**Time(Mean)**
0	2	10	0.03s
1	2	100	5.47s
2	5	100	2.73s

Fonte: Autores (2023)

A Tabela 5 apresenta os resultados médios e mínimos obtidos somadas todas as instâncias, comparando-os com a literatura. É possível ver que o algoritmo genético implementado encontra soluções eficientes mesmo sem indivíduos de alta qualidade na população inicial, como observado nas abordagens 1 e 2. Já na abordagem 3, verifica-se que em algumas instâncias o algoritmo superou os resultados encontrados na literatura, ocorrendo em 4 das 30 instâncias testadas. Esses resultados também foram alcançados nas outras abordagens.

Tabela 5: Média das Abordagens em relação Literatura

Approach	**Greedy**	**BL2**	**GA(Mean)**	**GA(Min)**
1	7.83%	7.06%	3.31%	1.44%
2	7.83%	7.06%	3.31%	1.43%
3	7.83%	7.06%	-0.09%	-0.29%

Fonte: Autores (2023)

Todas as implementações necessárias para o presente estudo foram construídas com a linguagem de programação Python, versão 3.8. Os testes foram aplicados em um computador equipado com processador AMD Ryzen 5 3500X 6-Core 3.59GHz, 16GB de memória RAM e sistema operacional Ubuntu 22.04, rodando em um sistema de armazenamento do tipo SSD.

5 DISCUSSÕES

Como sugestão para trabalhos futuros, propõe-se investigar novas estratégias de busca local do tipo intrarota, que sejam mais eficientes do que a inserção generalizada descartada nos testes anteriores. A etapa de mutação do algoritmo genético, poderia incluir métodos de melhoria, como o uso de heurísticas, para explorar outras regiões não visitadas buscando novas bacias de atratividade. Além disso, sugere-se testar o comportamento do algoritmo, com diferentes métodos de seleção de pais, como o método de torneio em vez do método de roleta atualmente utilizado, e no cruzamento dos cromossomos, o corte poderia ser realizado em mais de um ponto, de acordo com a configuração da instância, para obter uma maior diversidade de soluções e aumentar as chances de encontrar soluções ótimas.

REFERÊNCIA

R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell system technical journal*, 45(9):1563–1581, 1966.

R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

F. M. Muller. Algoritmos heurísticos e exatos para resolução do problema de sequenciamento em processadores paralelos. 1993.

J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.

N. Slack, S. Chambers, R. Johnston, and A. Betts. *Gerenciamento de Operações e de Processos: princípios e práticas de impacto estratégico*. Bookman, 2008.

K. Sörensen, M. Sevaux, and F. Glover. A history of metaheuristics. In *Handbook of heuristics*, pages 791–808. Springer, 2018.

Tabela 6: Resultado GA com soluções iniciais aleatórias

	Struct	Instance	C(Greedy)	C(BL2)	GA(Mean)	GA(Min)	C(BKS)	Greedy	BL2	Mean	Min
1	Y	2_10_01	400	400	372.05	354	354	11.50%	11.50%	4.85%	0.00%
2	Y	2_10_02	513	472	478.09	472	472	7.99%	0.00%	1.27%	0.00%
3	Y	2_10_03	464	424	386.17	379	382	17.67%	9.91%	1.08%	-0.79%
4	Y	2_10_04	381	381	374.15	364	364	4.46%	4.46%	2.71%	0.00%
5	Y	2_10_05	441	441	393.95	388	388	12.02%	12.02%	1.51%	0.00%
6	Y	2_100_01	3335	3257	3231.97	3182	3084	7.53%	5.31%	4.58%	3.08%
7	Y	2_100_02	3449	3406	3416.69	3378	3244	5.94%	4.76%	5.05%	3.97%
8	Y	2_100_03	3193	3136	3094.26	3038	2922	8.49%	6.82%	5.57%	3.82%
9	Y	2_100_04	3275	3159	3176.70	3112	3030	7.48%	4.08%	4.62%	2.63%
10	Y	2_100_05	3411	3332	3266.13	3195	3101	9.09%	6.93%	5.06%	2.94%
11	Y	5_100_01	1447	1423	1407.37	1386	1412	2.42%	0.77%	-0.33%	-1.88%
12	Y	5_100_02	1342	1299	1320.90	1291	1311	2.31%	-0.92%	0.75%	-1.55%
13	Y	5_100_03	1372	2328	1318.74	1297	1259	8.24%	45.92%	4.53%	2.93%
14	Y	5_100_04	1237	1202	1203.45	1180	1228	0.73%	-2.16%	-2.04%	-4.07%
15	Y	5_100_05	1410	1398	1403.22	1388	1374	2.55%	1.72%	2.08%	1.01%
16	N	2_10_01	315	315	258.76	242	242	23.17%	23.17%	6.48%	0.00%
17	N	2_10_02	462	462	429.69	418	418	9.52%	9.52%	2.72%	0.00%
18	N	2_10_03	388	388	347.01	342	342	11.86%	11.86%	1.44%	0.00%
19	N	2_10_04	294	264	259.22	256	256	12.93%	3.03%	1.24%	0.00%
20	N	2_10_05	369	369	339.90	334	334	9.49%	9.49%	1.74%	0.00%
21	N	2_100_01	2889	2822	2842.66	2809	2716	5.99%	3.76%	4.46%	3.31%
22	N	2_100_02	2884	2833	2823.62	2793	2690	6.73%	5.05%	4.73%	3.69%
23	N	2_100_03	2729	2683	2687.85	2649	2573	5.72%	4.10%	4.27%	2.87%
24	N	2_100_04	2648	2595	2567.76	2537	2447	7.59%	5.70%	4.70%	3.55%
25	N	2_100_05	2767	2734	2736.67	2706	2634	4.81%	3.66%	3.75%	2.66%
26	N	5_100_01	1183	1159	1164.72	1147	1110	6.17%	4.23%	4.70%	3.23%
27	N	5_100_02	1177	1163	1168.63	1148	1116	5.18%	4.04%	4.50%	2.79%
28	N	5_100_03	1135	1127	1120.36	1099	1067	5.99%	5.32%	4.76%	2.91%
29	N	5_100_04	1187	1163	1171.58	1153	1117	5.90%	3.96%	4.66%	3.12%
30	N	5_100_05	1367	1342	1346.29	1332	1293	5.41%	3.65%	3.96%	2.93%

Fonte: Autores (2023)

Tabela 7: Resultado GA informando uma solução inicial construtiva

	Struct	Instance	C(Greedy)	C(BL2)	GA(Mean)	GA(Min)	C(BKS)	Greedy	BL2	Mean	Min
1	Y	2_10_01	400	400	371.54	354	354	11.50%	11.50%	4.72%	0.00%
2	Y	2_10_02	513	472	477.74	472	472	7.99%	0.00%	1.20%	0.00%
3	Y	2_10_03	464	424	385.32	379	382	17.67%	9.91%	0.86%	-0.79%
4	Y	2_10_04	381	381	375.19	364	364	4.46%	4.46%	2.98%	0.00%
5	Y	2_10_05	441	441	394.35	388	388	12.02%	12.02%	1.61%	0.00%
6	Y	2_100_01	3335	3257	3231.62	3195	3084	7.53%	5.31%	4.57%	3.47%
7	Y	2_100_02	3449	3406	3416.07	3365	3244	5.94%	4.76%	5.04%	3.60%
8	Y	2_100_03	3193	3136	3093.80	3045	2922	8.49%	6.82%	5.55%	4.04%
9	Y	2_100_04	3275	3159	3178.34	3134	3030	7.48%	4.08%	4.67%	3.32%
10	Y	2_100_05	3411	3332	3265.10	3229	3101	9.09%	6.93%	5.03%	3.96%
11	Y	5_100_01	1447	1423	1407.05	1380	1412	2.42%	0.77%	-0.35%	-2.32%
12	Y	5_100_02	1342	1299	1322.05	1296	1311	2.31%	-0.92%	0.84%	-1.16%
13	Y	5_100_03	1372	2328	1320.87	1298	1259	8.24%	45.92%	4.68%	3.00%
14	Y	5_100_04	1237	1202	1202.41	1173	1228	0.73%	-2.16%	-2.13%	-4.69%
15	Y	5_100_05	1410	1398	1401.42	1382	1374	2.55%	1.72%	1.96%	0.58%
16	N	2_10_01	315	315	258.92	242	242	23.17%	23.17%	6.53%	0.00%
17	N	2_10_02	462	462	429.53	418	418	9.52%	9.52%	2.68%	0.00%
18	N	2_10_03	388	388	347.24	342	342	11.86%	11.86%	1.51%	0.00%
19	N	2_10_04	294	264	259.34	256	256	12.93%	3.03%	1.29%	0.00%
20	N	2_10_05	369	369	339.19	334	334	9.49%	9.49%	1.53%	0.00%
21	N	2_100_01	2889	2822	2845.75	2816	2716	5.99%	3.76%	4.56%	3.55%
22	N	2_100_02	2884	2833	2822.03	2784	2690	6.73%	5.05%	4.68%	3.38%
23	N	2_100_03	2729	2683	2690.21	2666	2573	5.72%	4.10%	4.36%	3.49%
24	N	2_100_04	2648	2595	2569.88	2526	2447	7.59%	5.70%	4.78%	3.13%
25	N	2_100_05	2767	2734	2736.49	2703	2634	4.81%	3.66%	3.75%	2.55%
26	N	5_100_01	1183	1159	1163.98	1140	1110	6.17%	4.23%	4.64%	2.63%
27	N	5_100_02	1177	1163	1169.98	1153	1116	5.18%	4.04%	4.61%	3.21%
28	N	5_100_03	1135	1127	1119.29	1098	1067	5.99%	5.32%	4.67%	2.82%
29	N	5_100_04	1187	1163	1169.94	1147	1117	5.90%	3.96%	4.53%	2.62%
30	N	5_100_05	1367	1342	1347.98	1326	1293	5.41%	3.65%	4.08%	2.49%

Fonte: Autores (2023)

Tabela 8: Resultado GA informando uma solução inicial da literatura

	Struct	Instance	C(Greedy)	C(BL2)	GA(Mean)	GA(Min)	C(BKS)	Greedy	BL2	Mean	Min
1	Y	2_10_01	400	400	354.00	354	354	11.50%	11.50%	0.00%	0.00%
2	Y	2_10_02	513	472	472.00	472	472	7.99%	0.00%	0.00%	0.00%
3	Y	2_10_03	464	424	381.55	379	382	17.67%	9.91%	-0.12%	-0.79%
4	Y	2_10_04	381	381	364.00	364	364	4.46%	4.46%	0.00%	0.00%
5	Y	2_10_05	441	441	388.00	388	388	12.02%	12.02%	0.00%	0.00%
6	Y	2_100_01	3335	3257	3084.00	3084	3084	7.53%	5.31%	0.00%	0.00%
7	Y	2_100_02	3449	3406	3244.00	3244	3244	5.94%	4.76%	0.00%	0.00%
8	Y	2_100_03	3193	3136	2922.00	2922	2922	8.49%	6.82%	0.00%	0.00%
9	Y	2_100_04	3275	3159	3030.00	3030	3030	7.48%	4.08%	0.00%	0.00%
10	Y	2_100_05	3411	3332	3101.00	3101	3101	9.09%	6.93%	0.00%	0.00%
11	Y	5_100_01	1447	1423	1405.89	1380	1412	2.42%	0.77%	-0.43%	-2.32%
12	Y	5_100_02	1342	1299	1310.05	1295	1311	2.31%	-0.92%	-0.07%	-1.24%
13	Y	5_100_03	1372	2328	1259.00	1259	1259	8.24%	45.92%	0.00%	0.00%
14	Y	5_100_04	1237	1202	1202.71	1178	1228	0.73%	-2.16%	-2.10%	-4.24%
15	Y	5_100_05	1410	1398	1374.00	1374	1374	2.55%	1.72%	0.00%	0.00%
16	N	2_10_01	315	315	242.00	242	242	23.17%	23.17%	0.00%	0.00%
17	N	2_10_02	462	462	418.00	418	418	9.52%	9.52%	0.00%	0.00%
18	N	2_10_03	388	388	342.00	342	342	11.86%	11.86%	0.00%	0.00%
19	N	2_10_04	294	264	256.00	256	256	12.93%	3.03%	0.00%	0.00%
20	N	2_10_05	369	369	334.00	334	334	9.49%	9.49%	0.00%	0.00%
21	N	2_100_01	2889	2822	2716.00	2716	2716	5.99%	3.76%	0.00%	0.00%
22	N	2_100_02	2884	2833	2690.00	2690	2690	6.73%	5.05%	0.00%	0.00%
23	N	2_100_03	2729	2683	2573.00	2573	2573	5.72%	4.10%	0.00%	0.00%
24	N	2_100_04	2648	2595	2447.00	2447	2447	7.59%	5.70%	0.00%	0.00%
25	N	2_100_05	2767	2734	2634.00	2634	2634	4.81%	3.66%	0.00%	0.00%
26	N	5_100_01	1183	1159	1110.00	1110	1110	6.17%	4.23%	0.00%	0.00%
27	N	5_100_02	1177	1163	1116.00	1116	1116	5.18%	4.04%	0.00%	0.00%
28	N	5_100_03	1135	1127	1067.00	1067	1067	5.99%	5.32%	0.00%	0.00%
29	N	5_100_04	1187	1163	1117.00	1117	1117	5.90%	3.96%	0.00%	0.00%
30	N	5_100_05	1367	1342	1293.00	1293	1293	5.41%	3.65%	0.00%	0.00%

Fonte: Autores (2023)