Implementando uma heurística construtiva e melhorando-a com três métodos de busca local para o problema de sequenciamento de tarefas com tempos de preparação dependentes da sequência em máquinas paralelas idênticas

Implementing a constructive heuristic and improving it with three local search methods for the task sequencing problem with sequence-dependent setup times on identical parallel machines

Alisson Michel Sganzerla
Universidade Federal de Santa Maria, Brasil
E-mail: alisson.sganzerla@gmail.com

Guilherme Lopes Weis
Universidade Federal de Santa Maria, Brasil
E-mail: glweis11@gmail.com.br

Eugênio Antônio Zanini
Universidade Federal de Santa Maria, Brasil
E-mail: gênio.zanini@gmail.com

Resumo:
O presente trabalho é proposto como requisito parcial da disciplina Heurísticas e Meta-Heurísticas. No presente estudo de caso, foi implementada uma heurística construtiva, obtendo-se com ela uma solução inicial factível para o problema de sequenciamento de tarefas em máquinas paralelas idênticas, sendo os tempos de preparação dependentes da sequência. Com essa solução inicial, foram realizados testes de melhoria utilizando-se três buscas locais diferentes, cada uma com uma composição de vizinhança diferente. Cada busca local utiliza uma ou mais vizinhanças do tipo: *swap, insertion* ou *generalized insertion.* No tópico resultados apresentamos um comparativo sobre o desempenho das buscas locais implementadas, um sobre o desempenho das vizinhanças quando aplicadas de forma isolada e uma comparação com os resultados da literatura (Muller, 2022).

Palavras-chave: Heurística; Metaheurística; Sequenciamento de tarefas. Busca Local

Abstract:
The present paper is proposed as a partial requirement of the discipline Heuristics and Meta-Heuristics. In the present case study, a constructive heuristic was implemented, obtaining with it an initial feasible solution for the task sequencing problem in identical parallel machines, with the setup times being sequence dependent. With this initial solution, improvement tests were performed using three different local searches, each with a different neighborhood composition. Each local search uses one or more neighborhoods of the type: swap, insertion or generalized insertion. In the results topic, we present a comparison on the performance of implemented local searches, one on the performance of neighborhoods when applied in isolation and a comparison with the results of the literature (Muller, 2022).

Keywords: Heuristics; Metaheuristics; Jobs Scheduling, Local Search

1. Introdução

A estrutura organizacional para tomada de decisões em ambientes corporativos pode ser dividida em três níveis: estratégico, tático e operacional. Sob a óptica dos prazos de entrega de resultados, o nível estratégico está relacionado com os resultados de longo prazo, o nível tático com resultados a médio prazo e o nível operacional com a entrega de resultados a curto prazo (Slack *et al*, 2008). O presente estudo abrange tomadas de decisão em nível operacional, o qual demanda respostas de qualidade em um curto período de tempo. Tais decisões, devido a sua complexidade, necessitam muitas vezes de suporte computacional. A busca de uma solução ótima para problemas complexos, pode demandar muito tempo computacional se empregado métodos exatos para tal. Nesses casos, alternativamente, podem ser utilizadas técnicas heurísticas e metaheurísticas, que poderão fornecer resultados de qualidade em um tempo computacional aceitável.

2. Revisão Teórica

A teoria da complexidade computacional, foi introduzida por Hartmanis e Stearns (1965) no artigo *"On the Computational Complexity of Algorithms"* que introduziu o conceito de "medida de complexidade" em função do tempo de computação utilizando máquinas de Turing. Nos estudos seguintes, os problemas foram classificados em determinísticos polinomiais, agrupados na classe P e problemas não determinísticos polinomiais, agrupados na classe NP.

Para tratar problemas complexos, que demandam alto tempo computacional quando empregado métodos exatos, as heurísticas podem ser uma ótima alternativa. Heurísticas são algoritmos que, baseados no conhecimento do problema, geram bons resultados em tempos computacionais aceitáveis, ainda que não garantam a otimalidade. Para Pearl (1984), heurísticas são um conjunto de critérios, métodos ou princípios para decidir qual alternativa de ações obtêm-se um resultado mais efetivo.

As heurísticas podem ser divididas em várias classes, entre as quais as construtivas e as de melhoria utilizadas no presente trabalho. Os métodos construtivos apresentam soluções iniciais factíveis para um problema qualquer, que dependendo da abordagem empregada podem ser de melhor ou pior qualidade. Já os métodos de melhoria, exigem uma solução inicial factível e a partir dela procuram oportunidades de melhorias através de técnicas como por exemplo a busca local. As soluções apresentadas após a etapa de melhoria podem ficar precocemente presas em ótimos locais não encontrando outras regiões de melhora. Para oportunizar regiões mais atrativas, utilizam-se métodos metaheurísticos. Eles proporcionam às heurísticas maior flexibilidade em aceitar soluções piores dentro de uma determinada regra, possibilitando buscas fora da região em que o modelo está preso. 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 (Sorensen, Sevaux, Glover, 2018).

O presente estudo, trata do sequenciamento de tarefas (*scheduling)* distribuídas em máquinas paralelas, sendo definido *makespan* (C_{max}) o maior tempo de processamento da máquina mais carregada. Com o objetivo de resolver problemas de sequenciamento de tarefas, Graham (1966) desenvolveu um método, onde uma lista de tarefas não ordenada é distribuída nas máquinas, de acordo com a sua carga. Esse método foi denominado LS (*List Scheduling*), 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 maior tempo de processamento é selecionada e alocada na máquina com menor carga. A partir desse estudo inicial variações de seu algoritmo são encontradas com certa frequência em adaptações para os problemas em máquinas paralelas uniformes e máquinas paralelas não relacionadas (Muller et al, 2002).

2. Metodologia

Foram disponibilizadas 30 instâncias, divididas em duas categorias denominadas: estruturadas e não estruturadas. Tanto dados estruturados quanto não estruturados foram gerados de forma aleatória, porém no primeiro grupo respeita-se a desigualdade do triângulo, já no segundo isso não é garantido.

Tabela 1. Organização, instâncias, máquinas, tarefas e tempos de preparo (*setup*)

Organização/Arquivo/Instância	Máquinas (m)	Tarefas (n)	Matriz (tempos de preparo)
Estruturados/5 arquivos/1-5	2	10	12x12
Estruturados/5 arquivos/1-5	2	100	102X102
Estruturados/5 arquivos/1-5	5	100	105x105
Não Estruturados/5 arquivos/1-5	2	10	12x12
Não Estruturados/5 arquivos/1-5	2	100	102X102
Não Estruturados/5 arquivos/1-5	5	100	105x105

Fonte: Autores (2022)

Abordagem utilizada para tratamento do problema

Em cada uma das 30 instâncias apresentadas, aplicou-se um método construtivo para obter uma solução inicial e com ela tentou-se uma melhora utilizando-se 3 diferentes métodos de busca local. Cada um desses métodos era composto de uma combinação de uma ou mais vizinhanças.

A solução factível do método construtivo foi obtida seguindo a heurística LPT e LS de Graham. O esquema abaixo apresenta o algoritmo construtivo em etapas:

Etapa 1 - Ordenação inicial das tarefas

Utilizando a heurística LPT, ordena-se as tarefas de acordo com o maior tempo de processamento;

Etapa 2 – Alocação das máquinas

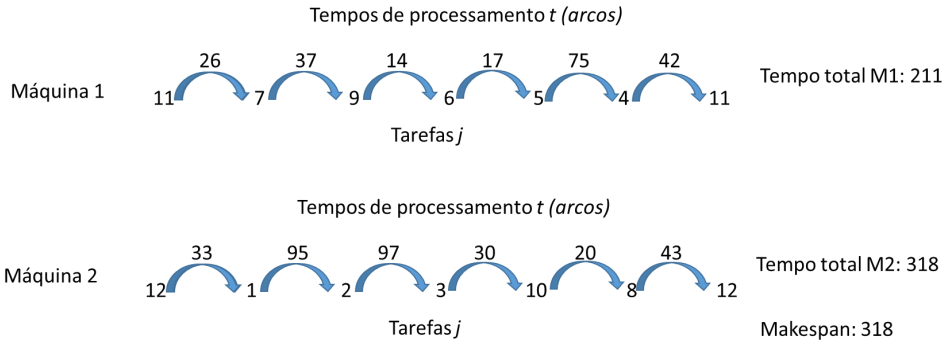
Seguindo o algoritmo LS, são distribuídas as tarefas ordenadas no passo anterior, escolhendo-se inicialmente a máquina de menor carga no momento da distribuição.

Em seguida escolhe-se a melhor posição para essa tarefa dentro da máquina, considerando as tarefas que já estavam alocadas nela.

Ao final dessa fase obtêm-se uma solução factível.

Na Figura 1, apresentamos o sequenciamento das tarefas, os tempos totais de processamento para cada máquina e o makespan. Nela está ilustrada uma solução inicial, obtida pelo método construtivo a partir de uma instância do tipo 2x10, ou seja duas máquinas e com dez tarefas. Os nós onde a seta aponta são as tarefas distribuídas, e a junção de dois nós indica o tempo total de execução dessa tarefa, já somados tempo de preparação da máquina com tempo de execução da tarefa.

Figura 1. Sequenciamento das tarefas para cada máquina, com seus respectivos tempos de processamento e makespan

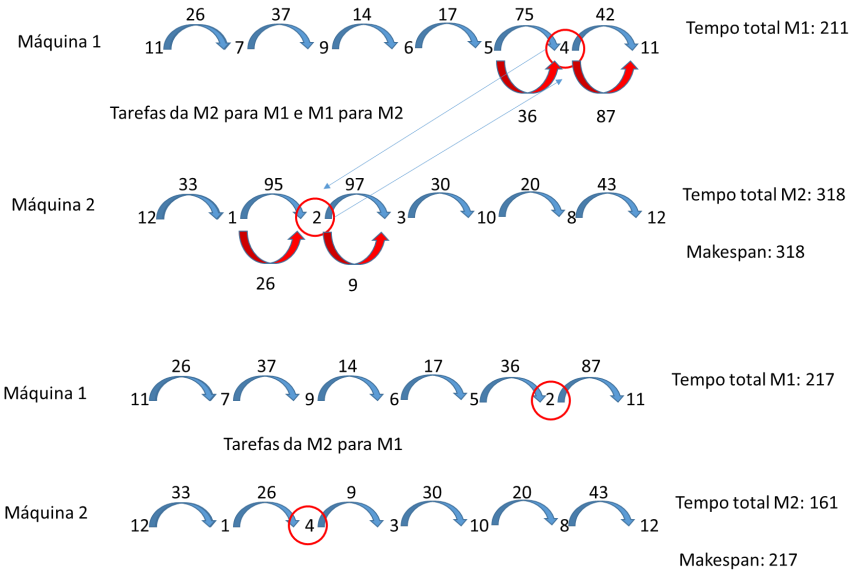


Fonte: Autores (2022)

Durante a etapa de melhoramento da solução inicial, tenta-se encontrar uma nova combinação de tarefas que reduza o tempo total da máquina mais carregada. Para contemplar esse objetivo, vizinhanças inter e intra-rotas foram combinadas na tentativa de desbloquear novas regiões para exploração das buscas. O termo inter-rota indica que a manipulação de tarefas ocorre entre duas ou mais máquinas, já o termo intra-rota a manipulação ocorre apenas em uma máquina. As vizinhanças implementadas no presente trabalho foram: *swap, insertion e generalized insertion.* Elas serão referenciadas respectivamente por conveniência como: *i, ii e iii*, e estão detalhadas abaixo:

- *swap (i)*: método do tipo inter-rota, nele tenta-se realizar a troca de todos os nós na máquina com maior carga com todas as tarefas das outras máquinas, avaliando-se qual a troca mais benéfica diminuirá o tempo total C_{max}. Para ilustrar a execução da vizinhança *swap*, utilizou-se os resultados obtidos na Figura 1. A Figura 2 exibe o melhor movimento encontrado entre todas as possibilidades, uma troca entre as máquinas 1 e 2, entre as tarefas 4 e 2. Essa troca reduz o C_{max} de 318 para 211.

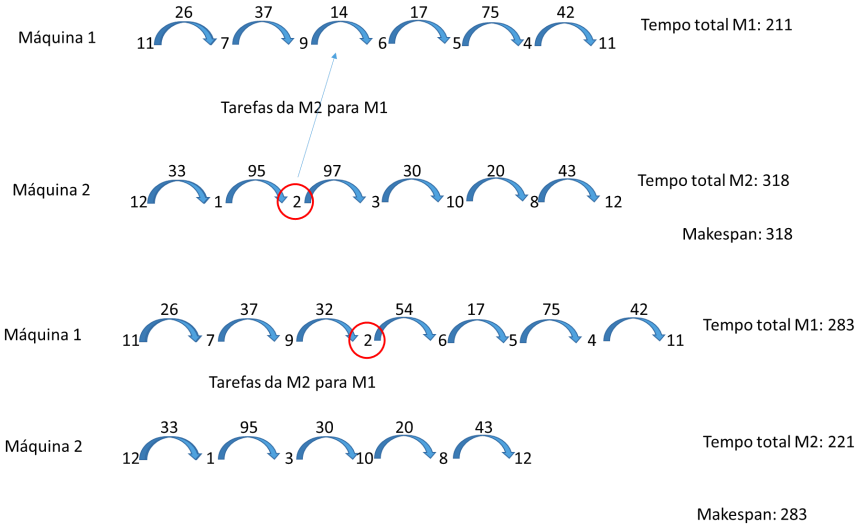
Figura 2. Vizinhança *swap reduzindo o tempo da máquina mais carregada*.



Fonte: Autores (2022)

- *insertion (ii):* método do tipo inter-rota, nele tenta-se remover uma tarefa da máquina mais carregada e repassá-la para qualquer outra, buscando-se a maior diminuição do C_{max}. Novamente para ilustrar a aplicação da vizinhança *insertion*, utilizou-se a solução inicial da Figura 1. A Figura 3 abaixo, exibe o melhor movimento encontrado: a remoção da tarefa 2 na máquina 2, inserindo-a na máquina 1. Essa operação reduz o C_{max} de 318 para 283.

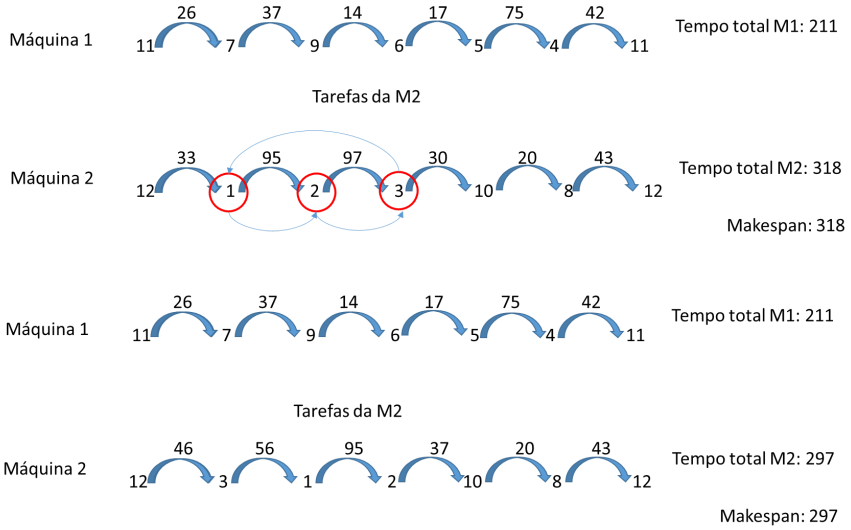
Figura 3. Vizinhança *insertion*.



Fonte: Autores (2022)

- *generalized insertion* (iii): método do tipo intra-rota, busca-se aqui realizar um rearranjo entre 3 tarefas quaisquer da própria máquina, isso implica que a máquina obrigatoriamente possua no mínimo 3 tarefas. Para exemplificar o novo arranjo, 3 tarefas quaisquer são escolhidas (***J, K, L***) e trocadas de posição, ***J*** troca com ***K, K*** troca com ***L*** e ***L*** troca com ***J*** resultando uma nova configuração (***L, J, K***). A Figura 4 abaixo exibe uma troca entre três tarefas (2, 3, 8) na máquina 2, novamente dados obtidos da solução inicial exibida na Figura 1. Essa nova troca de posição (8, 2, 3) reduziu C_{max} de 318 para 187.

Figura 4. Vizinhança *generalized insertion*



Fonte: Autores (2022)

Os métodos de busca local implementados são citados no presente trabalho como: **BL1, BL2 e BL3**. O Quadro 1 exibe a composição de cada método, com as respectivas estratégias de vizinhança: (i) *swap,* (ii) *insertion,* e (iii) *generalized insertion*.

Quadro 1. Composição de vizinhanças na busca local

Local Search	Neighborhood
BL1	iii, ii, i, ii, i, iii
BL2	i, ii, iii, i, ii, iii
BL3	iii, i, iii, ii, iii, ii, iii, i

Fonte: Autores (2022)

3. Resultados e Discussão

Todos os métodos de busca local e as respectivas vizinhanças foram implementados na 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, com 16GB de memória RAM, rodando em um sistema operacional Windows 11 Pro 22H2, rodando em um sistema de armazenamento do tipo SSD.

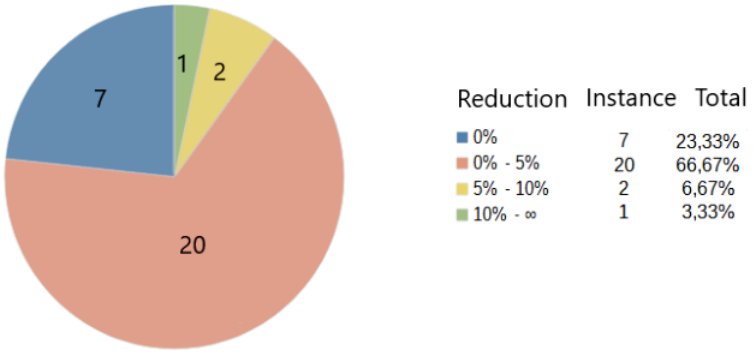
Tabela 2. Solução inicial da heurística construtiva

Instance	CMax(CH)		Instance	CMax(CH)	
001_struc_2_10_01	400	0,00	016_nonstruc_2_10_01	315	0,00
002_struc_2_10_02	513	0,01	017_nonstruc_2_10_02	462	0,00
003_struc_2_10_03	464	0,00	018_nonstruc_2_10_03	388	0,00
004_struc_2_10_04	381	0,00	019_nonstruc_2_10_04	294	0,00
005_struc_2_10_05	441	0,00	020_nonstruc_2_10_05	369	0,00
006_struc_2_100_01	3335	0,02	021_nonstruc_2_100_01	2889	0,02
007_struc_2_100_02	3449	0,02	022_nonstruc_2_100_02	2884	0,02
008_struc_2_100_03	3193	0,02	023_nonstruc_2_100_03	2729	0,02
009_struc_2_100_04	3275	0,02	024_nonstruc_2_100_04	2648	0,02
010_struc_2_100_05	3411	0,02	025_nonstruc_2_100_05	2767	0,02
011_struc_5_100_01	1447	0,02	026_nonstruc_5_100_01	1183	0,02
012_struc_5_100_02	1342	0,02	027_nonstruc_5_100_02	1177	0,02
013_struc_5_100_03	1372	0,02	028_nonstruc_5_100_03	1135	0,02
014_struc_5_100_04	1237	0,02	029_nonstruc_5_100_04	1187	0,02
015_struc_5_100_05	1410	0,02	030_nonstruc_5_100_05	1367	0,02

Fonte: Autores (2022)

Em 23 das 30 instâncias disponíveis, foi possível reduzir o valor do C_{max} com algum dos três métodos de busca local, isso representa mais de 75% das instâncias. Na Figura 5 é possível observar o percentual de melhora das instâncias em relação à solução inicial apresentada na tabela acima: 20 delas obtiveram redução de até 5%, 2 delas apresentaram redução entre 5% e 10% e 1 obteve mais de 10%.

Figura 5. Percentual de melhora nas instâncias



Fonte: Autores (2022)

Nos testes realizados, conforme o Quadro 2, o grupo de 10 instâncias do tipo 2x100, é responsável em média, por aproximadamente 93% do tempo de processamento dos testes realizados, apresentando ligeira diminuição nas instâncias não estruturadas.

Quadro 2. Tempo de execução por tipo de instância

Instance	Type	BL1	BL2	BL3	Type	BL1	BL2	BL3	Total Time	
2x10		0,00	0,00	0,00		0,00	0,00	0,00	0,01	0,02%
2x100	struct	8,04	7,49	10,36	nonstruct	7,34	6,78	9,67	49,69	93,32%
5x100		0,53	0,45	0,82		0,50	0,48	0,77	3,55	6,67%

Fonte: Autores (2022)

O Quadro 3 mostra o tempo de execução de cada método de busca local. Com ele é possível observar que a busca BL3 é a mais cara utilizando 40% do tempo total dos testes. Vale destacar aqui, que os métodos BL1 e BL2 apresentam um desempenho parecido, com ligeira vantagem para o método BL2.

Quadro 3. Tempo de execução por busca local

Instance	Type	2x10	2x100	5x100	Type	2x10	2x100	5x100	Total Time	
BL1		0,00	8,04	0,53		0,00	7,34	0,50	16,40	30,81%
BL2	struct	0,00	7,49	0,45	nonstruct	0,00	6,78	0,48	15,21	28,57%
BL3		0,00	10,36	0,82		0,00	9,67	0,77	21,63	40,62%

Fonte: Autores (2022)

Apresentamos na Equação 1, o cálculo para obter o percentual de redução no valor do C_{max} apresentado nas Tabelas 3 e 4, sendo *SBL* o valor obtido na busca local e *CMax(CH)* o valor inicial da heurística construtiva.

Equação 1. Equação para mensurar o percentual de redução no valor do C_{max}

$$P(\%) = \frac{SBL - CMax(CH)}{CMax(CH)}.100\%$$

Fonte: Autores (2022)

A Tabela 3 e a Tabela 4 apresentam destacados, os melhores índices de redução de cada busca local em cada instância. As linhas onde não há índice destacado, são as 7 instâncias que não apresentaram melhora. Essas 7 instâncias sem melhora, são do tipo 2x10, sendo 3 nos dados estruturados e 4 nos dados não estruturados.

Tabela 3. Busca local BL1, BL2 e BL3 em instâncias estruturadas

Instance	CMax(CH)		BL1			BL2			BL3		
001_struc_2_10_01	400	0,00	400	0,00	0,00%	400	0,00	0,00%	400	0,00	0,00%
002_struc_2_10_02	513	0,00	472	0,00	**-7,99%**	472	0,00	**-7,99%**	472	0,00	**-7,99%**
003_struc_2_10_03	464	0,00	424	0,00	**-8,62%**	424	0,00	**-8,62%**	424	0,00	**-8,62%**
004_struc_2_10_04	381	0,00	381	0,00	0,00%	381	0,00	0,00%	381	0,00	0,00%
005_struc_2_10_05	441	0,00	441	0,00	0,00%	441	0,00	0,00%	441	0,00	0,00%
006_struc_2_100_01	3335	0,02	3247	9,23	-2,64%	3257	9,11	-2,34%	3214	13,36	**-3,63%**
007_struc_2_100_02	3449	0,02	3406	4,30	**-1,25%**	3406	4,34	**-1,25%**	3406	6,69	**-1,25%**
008_struc_2_100_03	3193	0,02	3152	6,73	-1,28%	3136	6,38	**-1,79%**	3152	9,13	-1,28%
009_struc_2_100_04	3275	0,02	3159	12,71	-3,54%	3159	12,65	-3,54%	3136	14,95	**-4,24%**
010_struc_2_100_05	3411	0,02	3332	8,04	**-2,32%**	3332	7,49	**-2,32%**	3332	10,36	**-2,32%**
011_struc_5_100_01	1447	0,02	1423	0,39	**-1,66%**	1423	0,39	**-1,66%**	1423	0,68	**-1,66%**
012_struc_5_100_02	1342	0,02	1318	0,67	-1,79%	1299	0,67	**-3,20%**	1323	0,96	-1,42%
013_struc_5_100_03	1372	0,02	1331	0,53	-2,99%	1328	0,45	**-3,21%**	1331	0,82	-2,99%
014_struc_5_100_04	1237	0,02	1205	0,61	-2,59%	1202	0,60	**-2,83%**	1205	0,92	-2,59%
015_struc_5_100_05	1410	0,02	1398	0,43	**-0,85%**	1398	0,44	**-0,85%**	1398	0,73	**-0,85%**

Fonte: Autores(2022)

Tabela 4. Busca local BL1, BL2 e BL3 em instâncias não estruturadas

Instance	CMax(CH)		BL1			BL2			BL3		
016_nonstruc_2_10_01	315	0,00	315	0,00	0,00%	315	0,00	0,00%	315	0,00	0,00%
017_nonstruc_2_10_02	462	0,00	462	0,00	0,00%	462	0,00	0,00%	462	0,00	0,00%
018_nonstruc_2_10_03	388	0,00	388	0,00	0,00%	388	0,00	0,00%	388	0,00	0,00%
019_nonstruc_2_10_04	294	0,00	264	0,00	**-10,20%**	264	0,00	**-10,20%**	264	0,00	**-10,20%**
020_nonstruc_2_10_05	369	0,00	369	0,00	0,00%	369	0,00	0,00%	369	0,00	0,00%
021_nonstruc_2_100_01	2889	0,02	2845	8,65	-1,52%	2822	10,02	**-2,32%**	2845	11,00	-1,52%
022_nonstruc_2_100_02	2884	0,02	2828	7,34	**-1,94%**	2833	6,78	-1,77%	2828	9,67	**-1,94%**
023_nonstruc_2_100_03	2729	0,02	2673	6,53	**-2,05%**	2683	6,03	-1,69%	2673	9,02	**-2,05%**
024_nonstruc_2_100_04	2648	0,02	2595	8,04	**-2,00%**	2595	8,10	**-2,00%**	2595	10,39	**-2,00%**
025_nonstruc_2_100_05	2767	0,02	2734	5,53	**-1,19%**	2734	5,52	**-1,19%**	2734	7,89	**-1,19%**
026_nonstruc_5_100_01	1183	0,02	1174	0,50	-0,76%	1159	0,50	**-2,03%**	1174	0,80	-0,76%
027_nonstruc_5_100_02	1177	0,02	1163	0,43	**-1,19%**	1163	0,45	**-1,19%**	1163	0,72	**-1,19%**
028_nonstruc_5_100_03	1135	0,02	1127	0,50	**-0,70%**	1127	0,48	**-0,70%**	1127	0,77	**-0,70%**
029_nonstruc_5_100_04	1187	0,02	1174	0,47	-1,10%	1163	0,41	**-2,02%**	1171	0,72	-1,35%
030_nonstruc_5_100_05	1367	0,02	1345	0,68	-1,61%	1342	0,51	**-1,83%**	1345	0,86	-1,61%

Fonte: Autores(2022)

A busca local que mais vezes obteve a melhor solução, isoladamente ou empatada com outro método, foi a BL2. Ela apresentou o melhor desempenho em 19 instâncias (63% do total). O Quadro 4, exibe essa informação e também o desempenho dos outros métodos. Importante destacar, que BL2, além de ter o melhor desempenho na quantidade de instâncias otimizadas, também é o método que apresentou o melhor desempenho no tempo de execução, apresentado no Quadro 3.

Quadro 4. Desempenho do método de busca local

Local Search	struct	nonstruct	Total
BL1	6	7	43,33%
BL2	10	9	63,33%
BL3	8	7	50,00%

Fonte: Autores(2022)

Depois de identificado que o método mais eficiente é o BL2, uma segunda bateria de testes foi realizada para identificar qual vizinhança teve maior impacto dentro dessa busca local. Utilizando o mesmo conjunto de instâncias e a mesma solução inicial, três novas buscas locais rodaram os mesmos testes, mas agora cada uma delas com apenas uma vizinhança, respectivamente: i, ii e iii.

O resultado dessa nova bateria de testes é apresentado na Tabela 5 e Tabela 6. Analisando os dados dessas tabelas, observa-se que os maiores tempos de execução ocorreram novamente no tipo 2x100.

Tabela 5. Vizinhanças *i, ii, iii* em instância estruturadas

Instance	CMax(CH)		i			ii			iii		
001_struc_2_10_01	400	0,00	400	0,00	0,00%	400	0,00	0,00%	400	0,00	0,00%
002_struc_2_10_02	513	0,01	513	0,00	0,00%	472	0,00	**-7,99%**	513	0,00	0,00%
003_struc_2_10_03	464	0,00	464	0,00	0,00%	424	0,00	**-8,62%**	464	0,00	0,00%
004_struc_2_10_04	381	0,00	381	0,00	0,00%	381	0,00	0,00%	381	0,00	0,00%
005_struc_2_10_05	441	0,00	441	0,00	0,00%	441	0,00	0,00%	441	0,00	0,00%
006_struc_2_100_01	3335	0,02	3335	0,00	0,00%	3319	0,03	-0,48%	3289	6,67	**-1,38%**
007_struc_2_100_02	3449	0,02	3449	0,00	0,00%	3449	0,01	0,00%	3411	3,04	**-1,10%**
008_struc_2_100_03	3193	0,02	3188	0,05	-0,16%	3193	0,01	0,00%	3164	5,50	**-0,91%**
009_struc_2_100_04	3275	0,02	3275	0,00	0,00%	3275	0,01	0,00%	3219	10,54	**-1,71%**
010_struc_2_100_05	3411	0,02	3397	0,02	-0,41%	3411	0,01	0,00%	3372	6,84	**-1,14%**
011_struc_5_100_01	1447	0,02	1428	0,01	**-1,31%**	1447	0,01	0,00%	1447	0,21	0,00%
012_struc_5_100_02	1342	0,02	1338	0,01	-0,30%	1333	0,02	**-0,67%**	1336	0,48	-0,45%
013_struc_5_100_03	1372	0,02	1372	0,00	0,00%	1340	0,03	**-2,33%**	1372	0,33	0,00%
014_struc_5_100_04	1237	0,02	1237	0,01	0,00%	1220	0,02	**-1,37%**	1234	0,45	-0,24%
015_struc_5_100_05	1410	0,02	1410	0,02	0,00%	1410	0,01	0,00%	1399	0,26	**-0,78%**

Fonte: Autores(2022)

Tabela 6. Vizinhanças *i, ii, iii* em instância não estruturadas

Instance	CMax(CH)		i			ii			iii		
016_nonstruc_2_10_01	315	0,00	315	0,00	0,00%	315	0,00	0,00%	315	0,00	0,00%
017_nonstruc_2_10_02	462	0,00	462	0,00	0,00%	462	0,00	0,00%	462	0,00	0,00%
018_nonstruc_2_10_03	388	0,00	388	0,00	0,00%	388	0,00	0,00%	388	0,00	0,00%
019_nonstruc_2_10_04	294	0,00	294	0,00	0,00%	264	0,00	**-10,20%**	294	0,00	0,00%
020_nonstruc_2_10_05	369	0,00	369	0,00	0,00%	369	0,00	0,00%	369	0,00	0,00%
021_nonstruc_2_100_01	2889	0,02	2889	0,00	0,00%	2876	0,02	-0,45%	2845	7,42	**-1,52%**
022_nonstruc_2_100_02	2884	0,02	2884	0,00	0,00%	2883	0,02	-0,03%	2828	6,13	**-1,94%**
023_nonstruc_2_100_03	2729	0,02	2729	0,02	0,00%	2729	0,01	0,00%	2706	4,18	**-0,84%**
024_nonstruc_2_100_04	2648	0,02	2648	0,02	0,00%	2648	0,01	0,00%	2595	6,83	**-2,00%**
025_nonstruc_2_100_05	2767	0,02	2767	0,02	0,00%	2767	0,01	0,00%	2749	4,24	**-0,65%**
026_nonstruc_5_100_01	1183	0,02	1183	0,02	0,00%	1180	0,01	-0,25%	1174	0,36	**-0,76%**
027_nonstruc_5_100_02	1177	0,02	1177	0,01	0,00%	1177	0,01	0,00%	1167	0,27	**-0,85%**
028_nonstruc_5_100_03	1135	0,02	1135	0,01	0,00%	1133	0,02	-0,18%	1130	0,28	**-0,44%**
029_nonstruc_5_100_04	1187	0,02	1182	0,00	-0,42%	1181	0,02	**-0,51%**	1187	0,24	0,00%
030_nonstruc_5_100_05	1367	0,02	1367	0,01	0,00%	1351	0,02	**-1,17%**	1367	0,32	0,00%

Fonte: Autores (2022)

No Quadro 5, é possível consolidar a análise anterior, sendo verificado que as instâncias do tipo 2x100 comprometem 95,07% do tempo total de processamento dos testes.

Quadro 5. Tempo de execução por tipo de instância

Instance	Type	i	ii	iii	Type	i	ii	iii	Total Time	
2x10		0,00	0,00	0,00		0,00	0,00	0,00	0,00	0,00%
2x100	struct	0,00	0,01	6,67	nonstruct	0,02	0,01	6,13	12,83	95,07%
5x100		0,01	0,02	0,33		0,01	0,02	0,28	0,67	4,93%

Fonte: Autores(2022)

No Quadro 6, mostra-se que a vizinhança "*iii*" é a que demanda maior tempo computacional.

Quadro 6. Tempo de execução por tipo de vizinhança

Instance	Type	2x10	2x100	5x100	Type	2x10	2x100	5x100	Total Time	
i		0,00	0,00	0,01		0,00	0,02	0,01	0,04	0,29%
ii	struct	0,00	0,01	0,02	nonstruct	0,00	0,01	0,02	0,05	0,37%
iii		0,00	6,67	0,33		0,00	6,13	0,28	13,41	99,34%

Fonte: Autores(2022)

Já no Quadro 7, verifica-se que nenhuma busca local que rodou apenas uma vizinhança (*i, ii, ou iii*) reduz o custo do C_{max} mais que 46% das instâncias. Nessa avaliação, o melhor desempenho apresentado foi da vizinhança "*iii*" e a pior foi a "*i*".

Quadro 7. Tempo de execução por tipo de vizinhança

Local Search	struct	nonstruct	Total
i	1	0	3,33%
ii	5	3	26,67%
iii	6	8	46,67%

Fonte: Autores(2022)

Com o objetivo de nortear os resultados obtidos pelo método construtivo, apresenta-se na Tabela 7 uma comparação com a solução inicial encontrada por Muller (2022). Observa-se que em 14 das 30 instâncias, encontrou-se melhores resultados. Sendo o valor médio de diferença entre todas as instâncias de 0,07%.

Tabela 7. Comparação das soluções obtidas pelo método construtivo e de melhoria comparados com os resultados obtidos por Muller (2022)

Instance	Initial Solution			Improvement		
	Ref	CMax(CH)	Gap	Ref	BL2	Gap
001_struc_2_10_01	429	400	**-7,25%**	354	400	11,50%
002_struc_2_10_02	472	513	7,99%	472	472	**0,00%**
003_struc_2_10_03	445	464	4,09%	382	424	9,91%
004_struc_2_10_04	414	381	**-8,66%**	364	381	4,46%
005_struc_2_10_05	425	441	3,63%	388	441	12,02%
006_struc_2_100_01	3243	3335	2,76%	3084	3257	5,31%
007_struc_2_100_02	3441	3449	0,23%	3244	3406	4,76%
008_struc_2_100_03	3090	3193	3,23%	2922	3136	6,82%
009_struc_2_100_04	3267	3275	0,24%	3030	3159	4,08%
010_struc_2_100_05	3285	3411	3,69%	3131	3332	6,03%
011_struc_5_100_01	1437	1447	0,69%	1341	1423	5,76%
012_struc_5_100_02	1388	1342	**-3,43%**	1257	1299	3,23%
013_struc_5_100_03	1342	1372	2,19%	1261	1328	5,05%
014_struc_5_100_04	1281	1237	**-3,56%**	1162	1202	3,33%
015_struc_5_100_05	1448	1410	**-2,70%**	1349	1398	3,51%
016_nonstruc_2_10_01	242	315	23,17%	242	315	23,17%
017_nonstruc_2_10_02	471	462	**-1,95%**	418	462	9,52%
018_nonstruc_2_10_03	414	388	**-6,70%**	342	388	11,86%
019_nonstruc_2_10_04	313	294	**-6,46%**	256	264	3,03%
020_nonstruc_2_10_05	386	369	**-4,61%**	334	369	9,49%
021_nonstruc_2_100_01	2856	2889	1,14%	2716	2822	3,76%
022_nonstruc_2_100_02	2861	2884	0,80%	2690	2833	5,05%
023_nonstruc_2_100_03	2692	2729	1,36%	2573	2683	4,10%
024_nonstruc_2_100_04	2593	2648	2,08%	2447	2595	5,70%
025_nonstruc_2_100_05	2734	2767	1,19%	2634	2734	3,66%
026_nonstruc_5_100_01	1199	1183	**-1,35%**	1110	1159	4,23%
027_nonstruc_5_100_02	1234	1177	**-4,84%**	1116	1163	4,04%
028_nonstruc_5_100_03	1163	1135	**-2,47%**	1067	1127	5,32%
029_nonstruc_5_100_04	1196	1187	**-0,76%**	1117	1163	3,96%
030_nonstruc_5_100_05	1388	1367	**-1,54%**	1293	1342	3,65%
			0,07%			6,21%

Fonte: Autores(2022)

Já os resultados obtidos pela busca local mais eficiente entre as três, a BL2, quando comparados com a etapa de pós-otimalidade apresentada por Muller (2022) tiveram um valor médio de diferença entre todas as instâncias de 6,21%, também exibidos na Tabela 7.

4. Considerações Finais

Durante os testes realizados foi possível identificar que os métodos de busca local que utilizaram vizinhanças isoladas não foram capazes de superar os resultados daqueles que usaram elas de forma combinada.

Os dois métodos BL1 e BL2 possuindo a mesma quantidade de vizinhanças, diferindo apenas pela ordem em que foram combinadas, apresentaram resultados diferentes. Observa-se com isso, que as soluções são dependentes do arranjo que as vizinhanças foram dispostas.

Observa-se também, que durante a análise individual das vizinhanças, “iii” teve um desempenho pior quando aplicadas ao tipo de instâncias 2x100.

Em relação aos dados da literatura, identificou-se que em 46,6% das instâncias, os resultados obtidos pelo método construtivo são melhores do que os fornecidos por Muller, e a solução fornecida pela busca BL2, ficou em média 6,21% acima do resultado de pós-otimalidade de Muller.

Referências

Slack, N. et al (2008). *Gerenciamento de Operações e de Processos: princípios e práticas de impacto estratégico.* Porto Alegre: Bookman.

Hartmanis, J., & Stearns, R. E. (1965). On the computational complexity of algorithms. Transactions of the American Mathematical Society, 117, 285-306.

Muller, F. M. (1993). Algoritmos heurísticos e exatos para resolução do problema de sequenciamento em processadores paralelos. Doctoral dissertation, Universidade Estadual de Campinas.

Pearl, J. (1984). Heuristics: Intelligent search strategies for computer problem solving. United States.

Müller, F. M., Dias, O. B., & Araújo, O. C. B. D. (2002). Algoritmo para o problema de sequenciamento em máquinas paralelas não-relacionadas. Production, 12, 6-17.

Martins, J. (2017). *Metodologia da Pesquisa Científica*. Dowbis. https://www.academia.edu/37444609/Metodologia_da_Pesquisa_Cient%C3%ADfica

Lin, S. (1965). Computer solutions of the traveling salesman problem. Bell System Technical Journal, 44(10), 2245-2269.

Lin, S. e Kernighan, B. (1973). An effective heuristic algorithm for the traveling salesman problem, Operations Research, 21, 498-516.

Sörensen, K.; Sevaux, M, Glover, F. A History of Metaheuristics. In: SPRINGER (Ed.). Handbook of heuristics. Springer, 2018. p.1–18.