São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

Busca Tabu Aplicada ao Sequenciamento de Tarefas com o Tempo Dependente de Sequência em Sistemas de Manufatura Flexível

Gabriel Carvalho Domingos da Conceição
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil
gabriel.conceicao@aluno.ufop.edu.br

Marco Antonio Moreira de Carvalho
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, Ouro Preto, Minas Gerais, 35400-000, Brasil
mamc@ufop.edu.br

RESUMO

O problema de minimização de trocas de ferramentas (SSP) com o tempo dependente de sequência é composto por um conjunto de tarefas, como torneamento, retificação ou soldagem, que exigem um conjunto limitado de ferramentas, como lixas, furadeiras e parafusadoras, em uma máquina com capacidade limitada para armazená-las. O objetivo é encontrar uma sequência de tarefas que minimize o tempo total gasto com as trocas de ferramentas, que é dependente da sequência. O SSP pode ser dividido em dois subproblemas: o problema de sequenciamento das tarefas (SP) e o problema de alocação das ferramentas (TP). Neste estudo, foi aplicada a estratégia de busca tabu para tratar o SP, enquanto o TP foi abordado por meio de um método exato e uma busca local para lidar com o tempo dependente de sequência. Os resultados demonstraram a competitividade do método proposto para todas as instâncias disponíveis, com melhorias de até 42,61% quando comparado ao método estado da arte.

PALAVRAS CHAVE. Tempo dependente de sequência, Busca Tabu, troca de ferramentas. Área Principal: POI, ADG&GP, MH

ABSTRACT

The minimization of tool switches problem (SSP) with sequence-dependent setup time consists of a set of tasks, such as turning, grinding, or welding, that require a limited set of tools, such as sandpaper, drills, and screwdrivers, on a machine with limited capacity to store them. The objective is to find a sequence of tasks that minimizes the total time spent on tool switching, which is sequence-dependent. The SSP can be divided into two subproblems: the job sequencing problem (SP) and the tool allocation problem (TP). In this study, the tabu search strategy was applied to address the SP, while the TP was tackled using an exact method and a local search to handle the sequence-dependent setup time. The results demonstrated the competitiveness of the proposed method for all available instances, with improvements of up to 42.61% when compared to the state of the art method.

KEYWORDS. Sequence-dependent time, Tabu Search, tool switching.

Main Area: OPI, AD&GP, MH

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

1. Introdução

Um sistema de manufatura flexível (*flexible manufacturing system*, FMS) é um sistema composto por uma rede de máquinas interconectadas e controladas por um sistema automatizado. Essas máquinas são projetadas para serem altamente flexíveis e capazes de lidar com uma ampla variedade de produtos e processos de fabricação. Em um FMS, são utilizadas máquinas programáveis capazes de executar diversas tarefas, como usinagem, fresamento e torneamento. Tais máquinas são equipadas com um compartimento chamado *magazine*, em que as ferramentas necessárias, como lixadeiras, parafusadoras e prensas são instaladas.

Idealmente, todas as ferramentas necessárias para processar todas as tarefas estariam disponíveis no *magazine*. No entanto, devido às limitações de espaço, é comum que não seja possível armazenar todas as ferramentas simultaneamente. Isso leva à necessidade de realizar trocas de ferramentas durante o processo de fabricação, o que resulta em interrupções na produção. Para minimizar essas interrupções e otimizar o tempo de produção, é essencial planejar um sequenciamento adequado das tarefas. O objetivo é reduzir ao máximo o tempo gasto com as trocas de ferramentas, maximizando assim a eficiência do FMS.

O problema de determinação de um sequenciamento eficiente das tarefas a fim de minimizar estas interrupções é conhecido como o problema de minimização de trocas de ferramentas (ou *job sequencing and tool switching problem*, SSP). Professor [1988] conceitua uma versão uniforme do SSP, em que considera-se um sistema composto por uma única máquina e ferramentas de tamanho igual ocupando apenas um compartimento no *magazine*. Além disso, as trocas de ferramentas não ocorrem simultaneamente, os tempos de trocas são constantes e não são considerados desgastes ou quebras das ferramentas. No entanto, o SSP pode assumir diferentes variações, dependendo das características do sistema de manufatura em questão. Neste estudo, será tratado o cenário em que os tempos de trocas entre pares de ferramentas não são constantes e dependem do par específico de ferramentas a serem trocadas.

Tang e Denardo [1988] decompõem o SSP em dois subproblemas, o problema de sequenciamento das tarefas (SP) e o problema de alocação das ferramentas (TP). O SP é classificado como *NP*-Difícil, o que significa que, atualmente, determinar uma solução ótima é computacionalmente complexo. Por outro lado, o TP pode ser solucionado por um algoritmo determinístico em tempo polinomial, como o algoritmo *keep tool needed soonest* (KTNS) de complexidade $O(mn)$, em que m corresponde à quantidade de ferramentas necessárias para concluir as n tarefas, ou por meio do algoritmo *greedy pipe construction algorithm* (GPCA) proposto em Cherniavskii e Goldengorin [2022], de complexidade $O(Cn)$, em que C corresponde à capacidade do *magazine*.

Uma instância para o SSP consiste em um conjunto de tarefas $J = \{1, 2, \ldots, n\}$, um conjunto de ferramentas $T = \{1, 2, \ldots, m\}$ e um *magazine* com capacidade C. Cada tarefa $i \in J$ requer um conjunto específico de ferramentas T_i. Portanto, uma solução completa para o problema é uma permutação dos elementos de J e a descrição dos pares de ferramentas a serem trocadas. Para ilustrar o SSP, considera-se uma instância com as seguintes características: 5 tarefas a serem processadas, um conjunto de 10 ferramentas disponíveis e um *magazine* com capacidade para armazenar até 5 ferramentas. A relação entre as tarefas e as ferramentas pode ser representada por meio de uma matriz binária, em que o valor 1 indica a dependência da tarefa em relação à ferramenta correspondente, e o valor 0 indica que a tarefa não depende daquela ferramenta específica. Além disso, é possível definir uma matriz de custos associada às trocas de ferramentas, incluindo os custos de configuração inicial da máquina flexível.

A Tabela 1 apresenta a matriz binária que representa a relação de dependência entre as tarefas e as ferramentas. A Tabela 2 exibe os tempos de trocas, em que a primeira linha indica os

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

tempos de configuração inicial e as demais linhas representam os tempos de troca entre as ferramentas i e j, representadas pela linha e coluna da matriz, respectivamente. Esse exemplo ilustra as principais características do SSP e fornece uma base para análise e proposição de soluções específicas para o problema, levando em consideração as restrições e objetivos particulares de um ambiente de manufatura flexível.

Tabela 1: Relação de tarefas e ferramentas.

	Tarefas				
Ferramentas	0	1	2	3	4
[1]	0	0	0	0	1
[2]	1	1	0	0	1
[3]	0	0	1	0	1
[4]	0	1	0	0	0
[5]	1	1	0	0	0
[6]	0	1	0	0	0
[7]	1	0	1	0	0
[8]	0	0	1	0	0
[9]	1	0	0	1	0
[10]	1	0	0	1	0

Tabela 2: Matriz de tempos de troca.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	1	2	3	4	5	6	7	8	9	10
1	0	5	2	1	4	5	1	4	2	2
2	4	0	5	4	5	2	2	1	5	4
3	2	2	0	3	4	2	4	5	3	1
4	4	3	1	0	4	3	2	3	4	3
5	4	4	2	1	0	3	3	5	3	2
6	2	2	4	4	3	0	4	4	1	2
7	2	2	5	4	2	1	0	4	5	5
8	1	3	5	1	5	2	3	0	1	3
9	1	1	5	1	1	2	2	3	0	4
10	1	5	1	5	4	3	5	4	5	0

Ao analisar a instância do SSP apresentada, é possível calcular o tempo das trocas de ferramentas ao ter a sequência de tarefas. Como exemplo ilustrativo, a Tabela 3 mostra os estados do *magazine* durante o processamento de cada tarefa. Considerando a sequência de tarefas J = [1, 4, 2, 3, 0], i a iteração atual e M_i o estado do *magazine* na iteração atual, podemos observar os passos de execução do KTNS:

1. A primeira tarefa, J_1, requer as ferramentas T_1 = [2, 4, 5, 6] e o *magazine* possui um compartimento vazio. Portanto, é inserida também uma ferramenta da próxima tarefa, resultando em M_1 = [1, 2, 4, 5, 6]. O tempo de configuração inicial é de 18 unidades de tempo.

2. A segunda tarefa, J_2, requer as ferramentas T_2 = [1, 2, 3]. É necessário realizar uma troca de ferramentas. Nesse caso, ocorre a troca da ferramenta 6 pela ferramenta 3, resultando em M_2 = [1, 2, 3, 4, 5]. O tempo total é atualizado para 22 unidades de tempo.

3. A terceira tarefa, J_3, requer as ferramentas T_3 = [3, 7, 8]. É necessário realizar duas trocas de ferramentas. As trocas ocorrem entre as ferramentas 1 e 7, e entre as ferramentas 4 e 8. Após as trocas, o *magazine* fica com o estado M_3 = [2, 3, 5, 7, 8] e o tempo total é atualizado para 26 unidades de tempo.

4. A quarta tarefa, J_4, requer as ferramentas T_4 = [9, 10]. É necessário realizar duas trocas de ferramentas. As trocas ocorrem entre as ferramentas 8 e 9, e entre as ferramentas 3 e 10. Após as trocas, o *magazine* fica com o estado M_4 = [2, 5, 7, 9, 10] e o tempo total é atualizado para 28 unidades de tempo.

5. Por fim, a quinta tarefa, J_5, requer exatamente as mesmas ferramentas presentes no estado atual do *magazine* M_4. Portanto, não são necessárias trocas de ferramentas. O *magazine* permanece como M_5 = [2, 5, 7, 9, 10] e o tempo total é de 28 unidades de tempo.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

Tabela 3: Estados do *magazine*.

J	—	1	2	3	4	5
T	[1]	1	1	0	0	0
	[2]	1	1	1	1	1
	[3]	0	1	1	0	0
	[4]	1	1	0	0	0
	[5]	1	1	1	1	1
	[6]	1	0	0	0	0
	[7]	0	0	1	1	1
	[8]	0	0	1	0	0
	[9]	0	0	0	1	1
	[10]	0	0	0	1	1

Para calcular o custo total das trocas necessárias, é utilizada uma variável binária x_{ijk}. Essa variável é definida da seguinte maneira:

$$x_{ijk} = \begin{cases} 1, & \text{se ocorrer a troca da ferramenta } i \text{ por } j \text{ durante o processamento da tarefa } k \\ 0, & \text{caso contrário} \end{cases}$$

Considerando n como o número de tarefas e uma matriz de tempo de trocas representada pela Tabela 2, a função de avaliação, expressa pela Equação 1, tem como objetivo minimizar o tempo gasto com as trocas de ferramentas necessárias para processar todas as tarefas.

$$\min F = \sum_{k=1}^{n} \sum_{i \in T} \sum_{j \in T} x_{ijk} \times \text{custo}_i^j \tag{1}$$

Este estudo aborda o SSP com tempo dependente de sequência utilizando diferentes de estratégias. Para o SP, é aplicada a técnica de busca tabu, visando otimizar o processo de sequenciamento das tarefas. Já o TP é abordado por meio de um método exato e uma busca local, com o objetivo de encontrar os melhores pares de ferramentas para troca. São consideradas instâncias *benchmark* da literatura e os resultados obtidos são comparados com o atual estado da arte.

Este trabalho está organizado como a seguir. Na Seção 2 é apresentada a revisão da literatura sobre o SSP com tempo dependente de sequência. A Seção 3 descreve os métodos propostos. Os experimentos computacionais são apresentados e discutidos na Seção 4. Por fim, a Seção 5, apresenta as conclusões a respeito do trabalho e aponta futuras direções de pesquisa.

2. Revisão da literatura

A literatura sobre o SSP engloba uma variedade de estudos que exploram diversas variações desse problema. O SSP uniforme considerando uma única máquina, com tamanhos uniformes de ferramentas e tempos de configuração independentes e iguais para todas as sequências, foi introduzido por Tang e Denardo [1988]. Neste estudo foi proposta a política KTNS para resolver o TP. Posteriormente, Crama [1997] provou que o SSP é um problema *NP*-difícil, o que estimulou o desenvolvimento de métodos aproximados para sua resolução. Uma revisão abrangente da literatura sobre o SSP e suas variações foi realizada por Calmels [2019], que engloba desde a primeira definição do problema até as variações existentes até o ano de 2018. Essa revisão fornece uma visão

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

geral completa e atualizada do campo, abordando diferentes abordagens e estratégias propostas para lidar com o SSP.

Privault e Finke [1995] foram os pioneiros a trabalharem no SSP com o tempo dependente de sequência. Reconhecendo a natureza *NP*-difícil desse problema, foi adotada uma abordagem heurística visando encontrar soluções próximas à ótima dentro de um tempo razoável. O estudo apresentou dois grupos principais de heurísticas para lidar com o SSP. O primeiro grupo de heurísticas consiste em abordagens que constroem um sequenciamento inicial e realizam trocas utilizando o KTNS. Algumas dessas abordagens incluem o método da inserção mais distante *(farthest insertion)*, super tarefa *(super task)* e melhor inserção *(best insertion)*.

O segundo grupo de heurísticas difere do primeiro pelo fato de construir o sequenciamento inicial e efetuar as trocas sem o auxílio de outro método. Esse grupo inclui o método do próximo melhor *(next best)*, a nova abordagem denominada gerenciamento de ferramentas em tempo real *(online tool management)* e o algoritmo de particionamento *(partitioning algorithm)*. Ao final, é realizada uma ampla comparação de desempenho entre os métodos mencionados. O algoritmo de particionamento se destaca em todos os testes em termos de qualidade das soluções, enquanto o super tarefa se destaca por convergir para a solução final de forma mais rápida.

Rifai et al. [2022], atual estado da arte para o SSP com tempo dependente de sequência, segue a estratégia proposta por Tang e Denardo [1988], decompondo o SSP em 2 sub-problemas. Para lidar com o TP, é utilizado o método KTNS combinado com a meta-heurística *simulated annealing* (SA). Essa abordagem visa encontrar a melhor alocação de ferramentas para cada tarefa, levando em consideração restrições de tempo e minimizando os custos envolvidos. No que diz respeito ao SP, são considerado três métodos diferentes. O primeiro é o *adaptive large neighborhood search* (ALNS), uma técnica que busca explorar e diversificar o espaço de soluções através de movimentos de vizinhança adaptativos. O segundo é um algoritmo genético (*genetic algorithm*, GA), que utiliza princípios de seleção natural e evolução para encontrar soluções promissoras. Por fim, o SA também é aplicado ao SP, proporcionando uma exploração mais ampla do espaço de soluções e permitindo escapar de ótimos locais.

Para aprimorar ainda mais os métodos desenvolvidos, Rifai et al. [2022] gerou 10 conjuntos de instâncias com maior complexidade. Essas instâncias apresentam um número significativamente maior de tarefas, ferramentas e capacidade do *magazine*, permitindo uma avaliação mais precisa e abrangente dos métodos propostos. Essa expansão do conjunto de instâncias contribui para uma avaliação mais robusta e aprofundada das abordagens adotadas. Posteriormente, na Seção 4, essas instâncias serão utilizadas nos experimentos computacionais deste estudo, possibilitando uma comparação dos resultados gerados com os resultados apresentados por Rifai et al. [2022].

3. Métodos propostos

Nas próximas seções são apresentadas a busca tabu para abordar o SP e duas abordagens distintas para tratar o problema do TP: uma busca local e um método exato.

3.1. Busca tabu aplicada ao problema de sequenciamento das tarefas

O algoritmo busca tabu, apresentado por Glover [1986] é uma técnica de busca heurística que ajuda a solucionar problemas de otimização combinatória. Utiliza-se uma lista tabu para armazenar soluções recentemente visitadas, evitando retornar a essas soluções durante a busca. Isso promove a diversificação da busca, permitindo explorar soluções além dos ótimos locais. O algoritmo realiza perturbações nas soluções atuais para gerar soluções vizinhas, que são avaliadas por uma função objetivo. Com critérios de intensificação e diversificação, o algoritmo busca melhorar iterativamente a solução atual, explorando movimentos promissores e evitando a estagnação em

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

Algoritmo 1 Busca Tabu.

```
1:  s ← solucaoAleatoria;
2:  buscaLocal(s);
3:  i ← 0;
4:  enquanto i < maxIteracoes faça
5:      s' ← perturbarSolucao(s);
6:      s' ← buscaLocal(s');
7:      se s' ⩽ s então
8:          se s' ∉ listaTabu então
9:              s ← s';
10:             listaTabu ← listaTabu ∪ s';
11:         fim se
12:     fim se
13:     i ← i + 1
14: fim enquanto
```

ótimos locais. O tamanho da lista tabu é um parâmetro importante a ser definido, pois influencia a eficiência e a diversificação da busca. O Algoritmo 1 apresenta o pseudocódigo da busca tabu.

Conforme apresentado no Algoritmo 1, a busca tabu começa com uma solução inicial gerada aleatoriamente e realiza uma busca local para melhorar sua qualidade. Em seguida, entra em um laço que itera por um número fixo de vezes. Dentro desse laço uma solução perturbada é gerada e novamente passa por uma busca local. Se a solução perturbada for melhor ou igual à solução atual e não estiver na lista tabu (uma lista de soluções recentes proibidas), a solução atual é atualizada e adicionada a lista tabu. Esse processo permite explorar diferentes soluções, evitando ficar preso em ótimos locais.

A definição dos parâmetros da busca tabu é apresentada na Seção 4.1. A seguir, são apresentadas as estratégias de perturbação e busca local consideradas durante o desenvolvimento da busca tabu aplicada ao SSP.

3.1.1. *Swap*

O movimento da vizinhança *swap*, ilustrado na Figura 1 consiste em selecionar um par de tarefas e efetuar a troca. Neste exemplo j_2 e j_5, e trocar suas posições no sequenciamento.

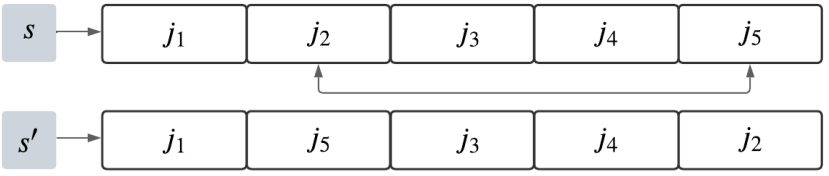


Figura 1: *Swap*.

É importante ressaltar que o tamanho da vizinhança *swap* é $O(n^2)$, em que n representa a quantidade de tarefas. Dentre as vizinhanças consideradas neste estudo, a *swap* é a mais rápida, levando em conta sua complexidade assintótica.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

3.1.2. *2-opt*

O movimento da vizinhança *2-opt* consiste em selecionar um intervalo na solução proposta e inverter a ordem dos elementos contidos nesse intervalo, conforme ilustrado na Figura 2, da tarefa j_2 a j_4. Em seguida, é verificado se essa inversão resultou em uma melhora na função objetivo. O tamanho desse intervalo pode variar de 2 até a quantidade total de elementos presentes na solução, representando as tarefas.

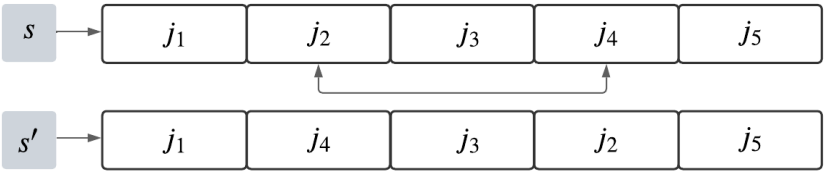


Figura 2: *2-opt*.

Com relação à complexidade, a vizinhança em questão tem um custo mais elevado para explorar a vizinhança em comparação com a *swap*. A complexidade é limitada $O(n^3)$, em que n representa a quantidade de tarefas.

3.1.3. *Insertion*

O movimento da vizinhança *insertion* envolve a escolha de uma tarefa existente no sequenciamento e a sua posterior inserção em uma posição selecionada aleatoriamente. Como ilustra a Figura 3, ao aplicar um movimento *insertion*, foi realizada a operação de inserção da tarefa j_1 na posição 3.

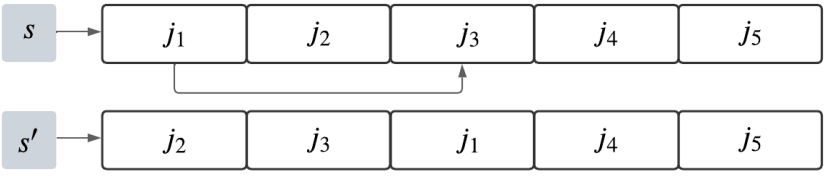


Figura 3: *Insertion*.

A vizinhança em questão possui uma complexidade de tempo limitada por $O(n^3)$, em que n é o número de tarefas. Isso significa que, em termos assintóticos, ela tem uma complexidade semelhante ao *2-opt*. No entanto, em comparação com o *swap*, essa estratégia tem um custo maior.

3.1.4. *1-block grouping*

O movimento da vizinhança *1-block grouping* utiliza a matriz binária que representa a relação entre tarefas e ferramentas como base para seus movimentos. Seu objetivo é agrupar os blocos contíguos de 1s na matriz, ou seja, as tarefas que utilizam a mesma ferramenta, através da troca de colunas como pode ser observado nas Tabelas 4 e 5. Essa troca de colunas na matriz resulta na reordenação da sequência de processamento das tarefas, otimizando a utilização das ferramentas.

Por exemplo, na Tabela 4, é possível observar que na linha 2 existem dois blocos de 1s separados por um bloco de 0s. A aplicação da operação *1-block grouping* agrupa esses dois blocos de 1s, trocando a tarefa j_2 com a j_4, como mostrado na Tabela 5. Com essa nova ordem das tarefas, é possível notar que a ferramenta 2 permanecerá no *magazine* durante o processamento das tarefas

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

Tabela 4: Relação de tarefas e ferramentas anterior ao *1-block grouping*.

Tarefas	—	0	1	2	3	4
Ferramentas	[1]	0	0	0	0	1
	[2]	**1**	**1**	0	0	**1**
	[3]	0	0	1	0	1
	[4]	0	1	0	0	0
	[5]	1	1	0	0	0
	[6]	0	1	0	0	0
	[7]	1	0	1	0	0
	[8]	0	0	1	0	0
	[9]	1	0	0	1	0
	[10]	1	0	0	1	0

Tabela 5: Relação de tarefas e ferramentas posterior ao *1-block grouping*.

Tarefas	—	0	1	2	3	4
Ferramentas	[1]	0	0	1	0	0
	[2]	**1**	**1**	**1**	0	0
	[3]	0	0	1	0	1
	[4]	0	1	0	0	0
	[5]	1	1	0	0	0
	[6]	0	1	0	0	0
	[7]	1	0	0	0	1
	[8]	0	0	0	0	1
	[9]	1	0	0	1	0
	[10]	1	0	0	1	0

j_0, j_1 e j_2. A complexidade desta busca local é $O(n^2m)$, em que n é a quantidade de tarefas e m é a quantidade de ferramentas.

3.2. Problema de alocação das ferramentas

Neste subproblema específico, há a necessidade de realizar trocas entre ferramentas a serem retiradas e ferramentas a serem instaladas, sendo que cada troca possui um custo associado. Para encontrar a combinação com o menor custo possível, o problema é modelado como um problema de atribuição linear em grafos. Esse problema é classificado como *P*, o que significa que pode ser resolvido de forma exata em tempo determinístico polinomial. No entanto, apesar dessa propriedade, o estudo realizado por Rifai et al. [2022] optou por utilizar meta-heurísticas para abordar esse problema.

Essa escolha pode ser atribuída ao fato de que meta-heurísticas têm a capacidade de encontrar soluções de boa qualidade em um tempo mais razoável. Com o objetivo de realizar uma comparação e análise mais abrangentes, este estudo realizou abordagens separadas utilizando um método exato, que é capaz de encontrar a solução ótima, e uma busca local na tentativa de melhorar a solução mais rapidamente.

3.2.1. Método exato

Modelando o problema de atribuição linear em grafos bipartidos, o grafo $G = (V, A)$ é definido com V representando o conjunto de vértices, que neste caso são as ferramentas disponíveis para atribuição. O conjunto V é então dividido em dois, V_i representando as ferramentas disponíveis no *magazine* durante o processamento da tarefa i, e V_{i+1}, representando as ferramentas disponíveis no magazine durante o processamento da tarefa $i + 1$.

Os arcos do grafo, representados por A, descrevem as possíveis trocas entre as ferramentas, com os pesos dos arcos refletindo o tempo necessário para realizar cada troca. Cada arco (u, v) em A possui uma capacidade que determina o fluxo máximo permitido através dele. No contexto do problema de atribuição linear, essa capacidade pode ser interpretada como a disponibilidade para realizar a troca entre as ferramentas.

Para resolver esse problema, pode-se utilizar o algoritmo *cost-scaling push-relabel*. Conforme descrito em Goldberg [1997], esse algoritmo possui complexidade de tempo limitada por $O(|V||A|\ \log(|V| \max_d))$, em que $\max_d$ representa o maior valor de um arco do grafo. Ao aplicar o algoritmo *cost-scaling push-relabel* ao grafo G, é possível encontrar uma atribuição ótima das ferramentas, minimizando o custo total das trocas.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

Com base nesse conceito, o método aplicado á solução do TP pode ser classificado como uma *math-heuristic*, pois combina elementos de uma meta-heurística com um método exato. Dessa forma, a *math-heuristic* combina a eficiência de um método exato em termos de qualidade da solução com a capacidade de explorar e diversificar a busca por meio da meta-heurística, resultando em soluções de alta qualidade para o problema de atribuição linear.

3.2.2. Busca local

Alternativamente ao método exato, e para aprimorar o desempenho do método e torná-lo mais competitivo em termos de tempo de execução, é possível utilizar uma abordagem de busca local, como ilustra a Figura 4, baseada na ideia da busca local *swap* mencionada anteriormente e aplicada ao SP.

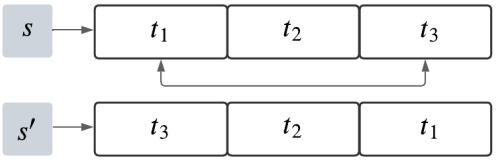


Figura 4: Movimento da busca local.

A Figura 4 ilustra um exemplo de movimento possível na busca local, no qual a ordem de remoção das ferramentas é trocada. Nesse caso, a ferramenta 1 é trocada com a ferramenta 2, resultando em diferentes pares de trocas, conforme mostrado na Figuras 5.

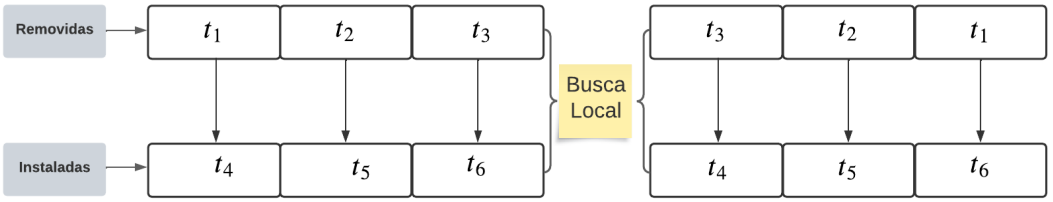


Figura 5: Resultado após a busca local.

A Figura 5 ilustra exemplos de trocas entre pares de ferramentas a serem removidas. No entanto, é importante ressaltar que esses mesmos movimentos também podem ser aplicados às ferramentas a serem instaladas. O que não é permitido é trocar ferramentas de conjuntos diferentes, ou seja, as trocas devem ser restritas dentro de um mesmo conjunto de ferramentas.

4. Experimentos computacionais

O ambiente computacional adotado para a realização dos experimentos consiste em um computador com processador Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, 16GB de memória RAM, e sob o sistema operacional Ubuntu 22.04.1 LTS. O método desenvolvido foi implementado em C++ e compilado utilizando o GCC versão 11.3.0, com as opções de otimização -O3 e march=native. Devido aos componentes de aleatoriedade, o método foi submetido a 10 execuções de forma independente para cada instância analisada. Foram consideradas as 40 instâncias, divididas em 10 subconjuntos, propostas e disponibilizados por Rifai et al. [2022]. Utilizou-se a implementação do algoritmo *cost-scaling push-relabel* disponível no o pacote *open-source or-tools*.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

4.1. Experimentos preliminares

Para definir os parâmetros de cada método proposto neste trabalho foram realizados testes estatísticos utilizando o pacote iRace López-Ibáñez et al. [2016]. A Tabela 6 apresenta as instâncias consideradas, juntamente com os parâmetros a serem definidos e as opções de calibração disponíveis. A opção escolhida para cada parâmetro está destacada em negrito. A linha *Instâncias* lista as instâncias utilizadas para a calibração dos métodos selecionadas aleatoriamente. A linha *Buscas Locais* contém as opções de buscas locais a serem utilizadas, a linha *Perturbação* lista as opções de perturbações e a linha *Iterações* indica o número máximo de iterações da busca tabu. A linha α apresenta as opções de percentual de perturbação de uma solução, a linha β lista os possíveis percentuais das vizinhança a serem explorados pelas buscas locais. Por fim, a linha *Lista Tabu* indica o tamanho máximo da lista tabu.

Tabela 6: Definição dos parâmetros.

Instâncias	J50C25_4	J60C50_1	J70C40_1	J70C60_4
Buscas Locais	***Swap***	2-opt	*Insertion*	*1-block grouping*
Perturbação	*Swap*	**2-opt**	*Insertion*	*1-block grouping*
Iterações	**100**	1000	10000	100000
β	45 50 55 60 65 70 75 80 85 90 95 **100**			
α	25 30 35 40 **45** 50			
Lista Tabu	50 100 150 **200** 250 300			

Com base nos resultados, concluiu-se que o componente de intensificação para o SP seria a busca local *swap* e o método *2-opt* como componente de perturbação. A busca local explora 100% da vizinhança, enquanto a perturbação altera até 45% da solução para escapar de ótimos locais. A lista tabu tem tamanho máximo igual 200 para garantir a eficiência do método. Por fim, buscando uma boa relação entre qualidade da solução e tempo de execução, o número máximo de iterações da busca tabu foi definido como 100.

4.2. Comparação com o estado da arte

Denotando a busca tabu desenvolvida neste estudo como *BT* e o método proposto por Rifai et al. [2022] como *ALNS*, o valor de $gap(\%)$ foi calculado usando a fórmula $100 \times \frac{\text{BT-ALNS}}{\text{ALNS}}$. Assim como Rifai et al. [2022], que limitou suas execuções a 10 minutos, o mesmo limite de tempo foi adotado neste estudo. A Tabela 7 apresenta a comparação entre a *math-heuristic* BT + exato e os resultados obtidos por Rifai et al. [2022]. Na referida tabela, a coluna *ALNS* representa o melhor resultado encontrado por Rifai et al. [2022] para cada grupo de instância, a coluna *BT* representa o melhor resultado obtido pela busca tabu combinada ao método exato para cada grupo de instâncias, a coluna μ representa a média geral para cada grupo de instâncias e a coluna σ representa o desvio padrão encontrado para cada grupo de instâncias. A fim de analisar a convergência da *math-heuristic*, os resultados obtidos em 50% do tempo limite também são apresentados.

Analisando a Tabela 7, conclui-se que a *math-heuristic* BT + exato foi eficiente, entregando resultados competitivos dentro do tempo limite de 10 minutos estipulados por Rifai et al. [2022]. Foi observado um *gap* mínimo de -9,07% em uma instância menos complexa e um *gap* máximo de -39,87% na instância mais difícil. Ao registrar os resultados obtidos em 50% do tempo limite, notou-se que a *math-heuristic* BT + exato converge para boas soluções bem antes do tempo limite. Foi registrado um *gap* mínimos de -8,97% em uma instância menos complexa e de -38,60% na instância mais difícil.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

Tabela 7: Resultados obtidos com a busca tabu e o método exato.

	Estado da Arte		*5 minutos*				*10 minutos*			
Grupo	*ALNS*	μ	*BT*	μ	σ	$\mathbf{gap}(\%)$	*BT*	μ	σ	$\mathbf{gap}(\%)$
J50C25	850,50	888,08	575,00	**607,15**	0,01	-31,63	560,00	**594,08**	0,01	-33,11
J50C30	653,50	682,83	496,00	**528,10**	0,02	-22,66	471,00	**515,93**	0,02	-24,44
J50C35	496,00	536,03	429,00	**456,28**	0,02	-14,88	421,00	**446,50**	0,02	-16,70
J50C40	438,75	459,33	383,00	**418,13**	0,03	-8,97	368,00	**417,65**	0,06	-9,07
J60C30	1300,75	1334,23	831,00	**859,85**	0,01	-35,55	813,00	**839,95**	0,01	-37,05
J60C40	791,75	824,25	614,00	**646,50**	0,02	-21,57	597,00	**632,50**	0,02	-23,26
J60C50	542,00	569,08	476,00	**514,48**	0,02	-9,59	457,00	**502,08**	0,02	-11,77
J70C40	1944,25	2007,55	1186,00	**1232,63**	0,01	-38,60	1156,00	**1207,15**	0,01	-39,87
J70C50	1209,75	1247,38	864,00	**914,43**	0,01	-26,69	874,00	**894,60**	0,01	-28,28
J70C60	809,00	839,25	680,00	**719,98**	0,02	-14,21	669,00	**700,18**	0,02	-16,57

A Tabela 8 apresenta a comparação entre BT + busca local e os resultados obtidos por Rifai et al. [2022], em que a coluna *ALNS* representa o melhor resultado encontrado por Rifai et al. [2022] para cada grupo de instância, *BT* representa o melhor resultado apresentado pela busca tabu desenvolvida neste estudo para cada grupo de instâncias, μ representa a média geral para para cada grupo de instâncias, σ representa o desvio padrão para cada grupo de instâncias e T o tempo médio de execução para cada grupo de instâncias.

Tabela 8: Resultados obtidos com a busca tabu e a busca local.

	Rifai et al. [2022]		*BT + busca local*				
Grupo	*ALNS*	μ	*BT*	μ	T	σ	$\mathbf{gap}(\%)$
J50C25	850,50	888,08	554,00	**579,45**	117,48	0,01	-34,75
J50C30	653,50	682,83	459,00	**495,60**	91,77	0,02	-27,42
J50C35	496,00	536,03	397,00	**417,98**	82,68	0,03	-22,02
J50C40	438,75	459,33	344,00	**379,48**	101,71	0,02	-17,38
J60C30	1300,75	1334,23	783,00	**810,33**	223,69	0,01	-39,27
J60C40	791,75	824,25	578,00	**598,08**	174,04	0,01	-27,44
J60C50	542,00	569,08	423,00	**464,40**	160,81	0,03	-18,39
J70C40	1944,25	2007,55	1109,00	**1152,13**	476,19	0,01	-42,61
J70C50	1209,75	1247,38	802,00	**837,10**	333,37	0,01	-32,89
J70C60	809,00	839,25	625,00	**648,79**	304,54	0,02	-22,69

Ao analisar a Tabela 8, constata-se que a abordagem BT + busca local obteve resultados competitivos dentro do prazo estabelecido de 10 minutos, conforme mencionado por Rifai et al. [2022]. Foi observado um *gap* mínimo de -17,38% em uma instância menos complexa, e um máximo de -42,61% em uma instância mais desafiadora. Ao examinar a convergência do método, fica evidente que as soluções finais são alcançadas em menos de 50% do tempo limite nos primeiros sete grupos de instâncias, com as instâncias mais complexas aproximando-se dos 5 minutos.

Ao analisar as Tabelas 7 e 8, podemos observar uma melhoria nos resultados obtidos, chegando a uma diferença de até 42,61% em relação ao conjunto completo de instâncias. O método BT + busca local se mostrou eficiente, convergindo para resultados superiores em um tempo menor quando comparado ao método BT + exato. Além disso, o BT + busca local apresentou um *gap* de -4,56% em relação ao BT + exato no maior grupo de instâncias. É importante destacar que o método também demonstra consistência, apresentando um baixo desvio padrão. Por fim, o método exibe uma boa convergência, alcançando as melhores soluções em torno da metade do tempo limite.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135

5. Conclusões e trabalhos futuros

O problema de minimização de trocas de ferramentas com tempo dependente de sequência em sistemas de manufatura flexível é amplamente reconhecido como um problema *NP*-Difícil e possui diversas aplicações no contexto industrial. Neste estudo, foram propostos dois métodos para abordar esse problema de forma eficiente: uma meta-heurística que combina busca tabu com busca local e uma *math-heuristic* busca tabu hibridizada com um método exato.

Foram conduzidos experimentos intensivos com os métodos propostos, e os resultados obtidos revelaram melhorias significativas em relação às melhores soluções conhecidas, alcançando uma diferença de até 42,61% no maior grupo de instâncias. Adicionalmente, o método converge próximo a 50% do tempo limite. As próximas etapas da pesquisa se concentrarão na exploração de outras ferramentas para a resolução exata do problema TP em um tempo menor, além de buscar a paralelização do algoritmo para lidar com o SSP de forma mais eficiente.

6. Agradecimentos

Esta pesquisa foi apoiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela Fundação de Apoio à Pesquisa do Estado de Minas Gerais – (FAPEMIG) e pela Universidade Federal de Ouro Preto (UFOP).

Referências

Calmels, D. (2019). The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends. *International Journal of Production Research*, 57(15-16):5005–5025.

Cherniavskii, M. e Goldengorin, B. (2022). An almost linear time complexity algorithm for the tool loading problem.

Crama, Y. (1997). Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99(1):136–153. ISSN 0377-2217.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549. ISSN 0305-0548. Applications of Integer Programming.

Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29. ISSN 0196-6774.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Privault, C. e Finke, G. (1995). Modelling a tool switching problem on a single nc-machine. *Journal of Intelligent Manufacturing*, 6(2):87–94. ISSN 1572-8145.

Professor, J. F. B. A. (1988). A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, 20(4):382–391.

Rifai, A. P., Windras Mara, S. T., e Norcahyo, R. (2022). A two-stage heuristic for the sequence-dependent job sequencing and tool switching problem. *Computers & Industrial Engineering*, 163:107813. ISSN 0360-8352.

Tang, C. S. e Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part i: Minimization of the number of tool switches. *Operations Research*, 36(5):767–777.

DOI: 10.59254/sbpo-2023-175135
http://live-proceedings.pantheonsite.io/p/175135