UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

GABRIEL CARVALHO DOMINGOS DA CONCEIÇÃO
Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

BUSCA TABU APLICADA AO SEQUENCIAMENTO DE TAREFAS COM O TEMPO DEPENDENTE DE SEQUÊNCIA EM SISTEMAS DE MANUFATURA FLEXÍVEL

Ouro Preto, MG

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

GABRIEL CARVALHO DOMINGOS DA CONCEIÇÃO

BUSCA TABU APLICADA AO SEQUENCIAMENTO DE TAREFAS COM O TEMPO DEPENDENTE DE SEQUÊNCIA EM SISTEMAS DE MANUFATURA FLEXÍVEL

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

Ouro Preto, MG

CARVALHO, Gabriel Carvalho.

Busca Tabu Aplicada ao Sequenciamento de Tarefas com o Tempo Dependente de Sequência em Sistemas de Manufatura Flexível/ Gabriel Carvalho Domingos da Conceição. –, -

23 p. 1 :il. (colors; grafs; tabs).

Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

Monografia – Universidade Federal de Ouro Preto,
Instituto de Ciências Exatas e Biológicas, Departamento de Computação, .

1. Palavra-chave 1. 2. Palavra-chave 2. 2. Palavra-chave 3. 3. Palavra-chave 4. 4. Palavra-chave 5. I. Prof. Dr. Marco Antonio Moreira de Carvalho. II. Universidade Federal de Ouro Preto. III. Busca Tabu Aplicada ao Sequenciamento de Tarefas com o Tempo Dependente de Sequência em Sistemas de Manufatura Flexível

Gabriel Carvalho Domingos da Conceição

BUSCA TABU APLICADA AO SEQUENCIAMENTO DE TAREFAS COM O TEMPO DEPENDENTE DE SEQUÊNCIA EM SISTEMAS DE MANUFATURA FLEXÍVEL

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau em Bacharel em Ciência da Computação.

Aprovada em Ouro Preto, 8 de fevereiro de 2024.

Prof. Dr. Marco Antonio Moreira de Carvalho
Universidade Federal de Ouro Preto - UFOP
Orientador

Prof. Dr. Leonardo Cabral da Rocha Soares
Instituto Federal do Sudeste de Minas Gerais - IFSEMG
Examinador

Prof. Ms. André Luís Barroso Almeida
Instituto Federal de Minas Gerais - IFMG
Examinador

Resumo

O problema de minimização de trocas de ferramentas (*job sequencing and tool switching problem*, 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, sejam instaladas 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 (*sequencing problem*, SP) e o problema de alocação das ferramentas (*tooling problem*, 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 dos métodos propostos para todas as instâncias disponíveis, com melhorias de até 44,43% quando comparado ao método estado da arte.

Palavras-chave: Tempo dependente de sequência. Busca Tabu. Troca de ferramentas.

Abstract

The *job sequencing and tool switching problem* (SSP) with sequence-dependent setup times consists of a set of jobs, such as turning, grinding, or welding, that require a limited set of tools, such as sandpaper, drills, and screwdrivers, to be installed on a machine with limited capacity to store them. The goal is to find a sequence of jobs that minimizes the total time spent on tool changes, which is dependent on the sequence. SSP can be divided into two subproblems: the job sequencing problem (*sequencing problem*, SP) and the tooling problem (*tooling problem*, TP). In this study, a tabu search strategy was applied to address the SP, while the TP was tackled using an exact method and a local search to handle sequence-dependent setup times. The results demonstrated the competitiveness of the proposed methods for all available instances, with improvements of up to 44.43% compared to the state-of-the-art method.

Keywords: Sequence-dependent time. Tabu Search. Tool switching.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ALNS	*adaptive large neighborhood search*
BT	*busca tabu*
FMS	*flexible manufacturing system*
GCA	*greedy pipe construction algorithm*
GA	*genetic algorithm*
KMPS	*keep machine needed soonest*
LAP	*linear assignment problem*
SA	*simulated annealing*
SP	*job sequencing problem*
SSP	*job sequencing and tool switching problem*
TP	*tool switching problem*

Lista de Símbolos

σ	desvio padrão
μ	média geral
T	tempo de execução

Sumário

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). BARD (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 (*sequencing problem*, SP) e o problema de alocação das ferramentas (*tooling problem*, 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 algoritmos determinísticos 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*.

Este estudo aborda o SSP com tempo dependente de sequência utilizando diferentes

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.

1.1 Motivação

O SSP é classificado como *NP-Difícil*, o que significa que não se conhece algoritmos conhecidos que possam resolver instâncias complexas desse problema de forma ótima em tempo determinístico polinomial. Portanto, é necessário adotar uma abordagem heurística ou metaheurística para obter soluções aproximadas em tempo razoável, pois a solução exata demandaria um tempo inviável para problemas de tamanho real. Nesse contexto, optou-se pelo uso da metaheurística busca tabu.

O SSP de tempo dependente de sequência possui ampla aplicação no ambiente fabril. Um exemplo prático desse problema é apresentado por Smith, Johnson e Brown (2018), que descreve a necessidade de pintar veículos com diferentes cores em uma linha de produção automotiva, levando em consideração as demandas e preferências dos clientes. Cada cor requer um conjunto específico de ferramentas, como pistolas de pintura, bicos e tanques de tinta. A troca de cor requer a limpeza e preparação das ferramentas, o que consome tempo e pode interromper temporariamente o processo de pintura. O objetivo é encontrar a sequência ideal de veículos a serem pintados, de forma a minimizar o tempo total de troca de ferramentas e maximizar a eficiência da linha de produção.

1.2 Objetivos

De maneira geral, o objetivo deste trabalho é elaborar uma heurística consistente para ser utilizada que permita a obtenção rápida de soluções próximas da solução ótima. São objetivos específicos:

- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o SSP com tempo dependente de sequência;
- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre a busca tabu;
- Avaliar o método implementado considerando dados reais e também com problemas teste publicamente disponíveis, realizando uma análise crítica considerando outros métodos da literatura;
- Publicar trabalhos em periódicos e eventos nacionais;

1.3 Organização do Trabalho

A organização deste trabalho segue a seguinte estrutura: no Capítulo 2, é apresentada uma revisão da literatura sobre o SSP com tempo dependente de sequência. O Capítulo 3 descreve os métodos propostos, enquanto o Capítulo 4 apresenta as estratégias adotadas para abordar o problema. Os experimentos computacionais são discutidos no Capítulo 5. Por fim, o Capítulo 6 apresenta as conclusões a respeito do trabalho e aponta futuras direções de pesquisa.

2 Revisão Bibliográfica

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 et al. (1994) 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, incluindo as variações existentes até o ano de 2018. Essa revisão fornece uma visão geral completa e atualizada do campo, apresentando 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 com o tempo dependente de sequência. 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 job)* 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, Mara e Norcahyo (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 dois 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 *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, Mara e Norcahyo (2022) conduziram um estudo em que criaram 10 conjuntos de instâncias com maior complexidade, com o objetivo de avaliar e explorar a capacidade dos seus métodos. Essas instâncias apresentam uma quantidade significativamente ampliada de tarefas, ferramentas e capacidade no magazine, totalizando até 70 tarefas, 90 ferramentas e uma capacidade de 60 *slots* no *magazine*. Essa expansão possibilita 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, no Capítulo 5, essas instâncias serão utilizadas nos experimentos computacionais deste estudo, possibilitando uma comparação dos resultados gerados com os resultados apresentados por Rifai, Mara e Norcahyo (2022).

3 Fundamentação Teórica

A seguir é apresentada uma definição formal do com o tempo dependente de sequência. Em seguida, são apresentadas as estratégias para abordar tanto o SP quanto o TP. Para o SP, é utilizada a abordagem da busca tabu, já para o TP é utilizada uma modelagem baseada no problema de atribuição linear (*linear assignment problem*, LAP).

3.1 SSP com tempo dependente de sequência

O SSP com o tempo dependente de sequência pode ser definido formalmente como 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 3.1 apresenta a matriz binária que representa a relação de dependência entre as tarefas e as ferramentas. A Tabela 3.2 exibe os tempos de trocas, em que a primeira linha indica os 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 com o tempo dependente de sequência 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.

Ao analisar a instância do SSP com o tempo dependente de sequência apresentada, é possível calcular o tempo das trocas de ferramentas ao ter a sequência de tarefas. Como exemplo ilustrativo, a Tabela 3.3 mostra os estados do *magazine* durante o processamento de cada tarefa. Considerando a sequência de tarefas J' = [1, 4, 2, 3, 5], 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,

Tabela 3.1 – Relação de tarefas e ferramentas.

	Tarefas				
Ferramentas	1	4	2	3	5
[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 3.2 – 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

resultando em M_1 = [1, 2, 4, 5, 6]. O tempos 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 tempos 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 de ferramentas. As trocas ocorrem entre as ferramentas 8 e 9, e entre as ferramentas 3 e 10. Após as trocas, 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.

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 3.2, a função de avaliação, expressa pela Equação 3.1, tem como objetivo minimizar o tempo gasto com as trocas de ferramentas necessárias para processar todas as tarefas.

Tabela 3.3 – 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

$$\min F = \sum_{k=1}^{n} \sum_{i \in T} \sum_{j \in T} x_{ijk} \times \text{custo}_i^j \tag{3.1}$$

A Equação 3.1 consiste em três somatórios aninhados. O somatório mais externo percorre cada *job* k, representando cada uma das n tarefas a serem realizadas. Os dois somatórios internos percorrem o conjunto de ferramentas disponíveis T, em que i representa uma ferramenta a ser removida e j representa uma ferramenta a ser instalada.

3.2 Busca tabu

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 movimentos recentemente realizados, evitando comportamentos cíclicos 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 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 inicia-se com uma solução inicial gerada aleatoriamente e, em seguida, realiza uma busca local para melhorar sua qualidade. Em seguida, entra em um laço que itera um número fixo de vezes. Dentro deste laço, uma solução perturbada é gerada e, novamente, submetida a uma busca local. Se a solução perturbada for melhor ou igual à solução atual e o movimento que levou a essa solução não estiver na lista tabu, a solução em análise é atualizada e os movimentos realizados são adicionados à lista tabu. Esse processo permite a exploração de diferentes soluções, evitando ficar preso em ótimos locais. Na

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 movimento ∉ listaTabu então
9:              s ← s';
10:             adicionarListaTabu(movimento);
11:         fim se
12:     fim se
13:     i ← i + 1
14: fim enquanto
```

Seção 4.1 é descrita a implementação da busca tabu aplicada ao SP, incluindo suas especializações para abordar o problema de forma mais eficiente.

3.3 Problema de atribuição linear aplicado ao problema de trocas de ferramentas

O TP é constituído por dois conjuntos de ferramentas: as que serão retiradas e as que serão instaladas. Essas trocas acontecem em um contexto em que a ordem em que as trocas ocorrem pode influenciar nos tempos individuais de cada operação. Nesse cenário, considerando todas as combinações possíveis de ferramentas, é viável representar o TP como um problema de atribuição linear por meio de um grafo bipartido, conforme ilustrado na Figura 3.1.

MARTELLO e TOTH (2011) descrevem o LAP como um problema de otimização combinatória que envolve a atribuição de tarefas a agentes de forma a minimizar ou maximizar uma determinada função objetivo. Ele é denominado linear porque a função objetivo e as restrições são lineares, ou seja, podem ser representadas por equações lineares. Formalmente o problema pode ser compreendido como um conjunto de tarefas $T = [1, 2, \ldots, m]$ e um conjunto de agentes $A = [1, 2, \ldots, m]$, onde cada tarefa t_i deve ser atribuída a um único agente a_j, e cada agente pode ser responsável por apenas uma tarefa.

O problema consiste em encontrar uma atribuição das tarefas aos agentes que minimize ou maximize uma função objetivo, geralmente representada como uma matriz de custos C, em que cada elemento c_{ij} representa o custo ou benefício de atribuir a tarefa t_i ao agente a_j. A tarefa é encontrar a combinação de atribuições que otimize a função objetivo, respeitando a restrição de que cada tarefa é atribuída a apenas um agente e cada agente pode ser responsável por no máximo uma tarefa.

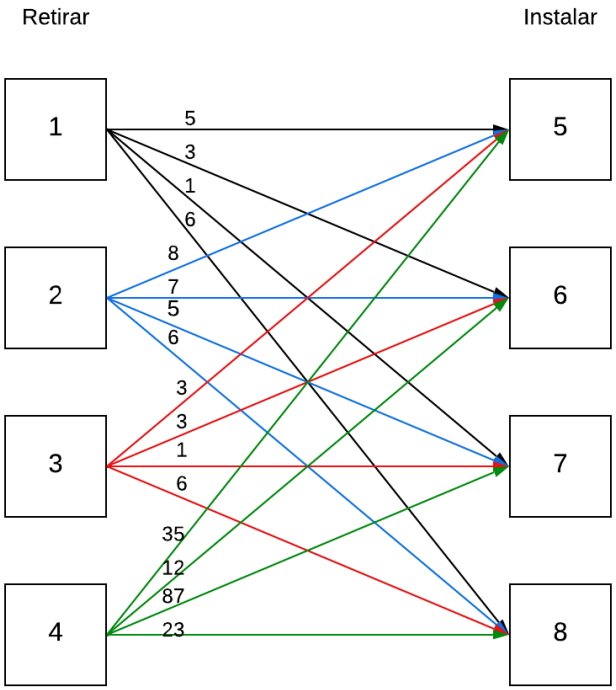

Figura 3.1 – Grafo representando as trocas de ferramentas.

O método húngaro, apresentado por Kuhn (1955), é um algoritmo eficiente para resolver o problema de atribuição linear em tempo determinístico polinomial, com complexidade $O(n^3)$, em que n é o número de tarefas ou agentes no problema. Esse algoritmo garante encontrar a solução ótima para o LAP, respeitando a restrição de atribuir apenas uma tarefa para cada agente.

Outra abordagem é o algoritmo *cost-scaling push-relabel*, inicialmente proposto por Goldberg e Tarjan (1990) e posteriormente apresentado em mais detalhes em Goldberg, Tarjan e Hed (2017). Essa técnica é uma variação do algoritmo *push-relabel*, originalmente proposto por Cherkassky e Goldberg (1995). O *cost-scaling push-relabel* é especialmente útil para resolver problemas de atribuição linear em que os custos das atribuições são relativamente grandes. Ele emprega o conceito de escalonamento de custos para melhorar a convergência do algoritmo, permitindo encontrar a solução ótima de forma mais eficiente. Sua complexidade depende da escala dos custos utilizados no problema. Quando os custos são multiplicados por um fator de escala k antes de iniciar o algoritmo, a complexidade é expressa como $O(n^3 \times \log(k))$, em que n é o número de tarefas ou agentes no problema.

Ambos os algoritmos, o método húngaro e o *cost-scaling push-relabel*, são fundamentais e amplamente utilizados para resolver problemas de atribuição linear, cada um apresentando suas características específicas e eficiência distintas. Enquanto o método húngaro é mais adequado para problemas com custos menores, o *cost-scaling push-relabel* é especialmente vantajoso quando os custos são grandes e precisam ser escalonados para obter uma melhor convergência.

4 Desenvolvimento

Nas próximas seções, são apresentadas as estratégias para abordar o SP e o TP. Para o SP, será utilizada a abordagem busca tabu, já para o TP, serão apresentadas duas abordagens distintas. A primeira é uma busca local, que foca na melhoria iterativa da solução inicial alterando os pares de ferramentes a serem trocados e a segunda abordagem é um método exato baseado no algoritmo *cost-scaling push-relabel*.

4.1 Busca tabu aplicada ao problema de sequenciamento das tarefas

Conforme exposto na Seção 3.2, a busca tabu se destaca por sua capacidade de evitar movimentos redundantes e possíveis estagnações em ótimos locais. Na aplicação da busca tabu ao problema SP, a implementação segue uma abordagem semelhante àquela apresentada no Algoritmo 1, com a diferença de que a solução inicial é gerada por meio de uma heurística construtiva apresentada na Subseção 4.1.1.

No entanto, a distinção ocorre nas linhas 1 e 10 do Algoritmo 1. Na linha 1, a diferença surge devido ao fato de a solução inicial ser gerada por meio de uma heurística construtiva detalhada na Subseção 4.1.1. Já na linha 10, em vez de armazenar os movimentos que conduziram a melhorias nas soluções recentemente encontradas, optou-se por armazenar os *hashes* correspondentes a essas soluções. Essa escolha de usar soluções se mostrou necessária pois neste problema é difícil se locomover entre diferentes soluções, ficando preso com facilidade em ótimos locais, por isso a solução foi colocada na *hash*, e não o movimento. Essa decisão, ademais, teve um impacto favorável na otimização da implementação, uma vez que a necessidade de verificar repetidamente a presença da solução em análise na lista tabu é um requisito constante. Utilizar uma tabela *hash* é mais eficiente do que comparar um conjunto completo de soluções em formato de *array*. Isso ocorre porque a pesquisa em uma lista tabu contendo *arrays* teria uma complexidade de $O(n)$, enquanto uma tabela *hash* teria uma complexidade de $O(\log n)$. Ao armazenar *hashes* em vez de soluções completas, o algoritmo consegue verificar rapidamente se a solução atual já foi visitada, evitando soluções repetidas e, assim, melhorando consideravelmente o desempenho da busca tabu.

A definição dos parâmetros da busca tabu é apresentada na Seção 5.1. A seguir, são apresentadas as estratégias de perturbação e busca local consideradas durante o desenvolvimento da busca tabu aplicada ao SSP.

4.1.1 *Heurística Construtiva*

Uma solução inicial é gerada por meio de uma heurística construtiva, em que a escolha da primeira tarefa é orientada pelo menor custo de *setup*. As posições subsequentes são preenchidas considerando as similaridades entre as tarefas, calculadas a partir do custo aproximado das trocas de ferramentas entre elas. A terminologia "aproximado" é empregada, uma vez que alcançar um valor exato demandaria a construção de um grafo bipartido, no qual ambas as tarefas deveriam requerer a mesma quantidade de ferramentas. Como as tarefas não necessariamente demandam o mesmo número de ferramentas, são adicionadas arestas artificiais, cujos pesos correspondem aos custos de *setup* das ferramentas em questão. Supondo que T_i = [1, 2, 3, 4] e T_{i+1} = [5, 6, 7], ferramentas artificiais são acrescentadas ao conjunto de ferramentas da tarefa com menor quantidade de ferramentas necessárias, neste caso, T_{i+1}, conforme ilustrado na Figura 4.1.

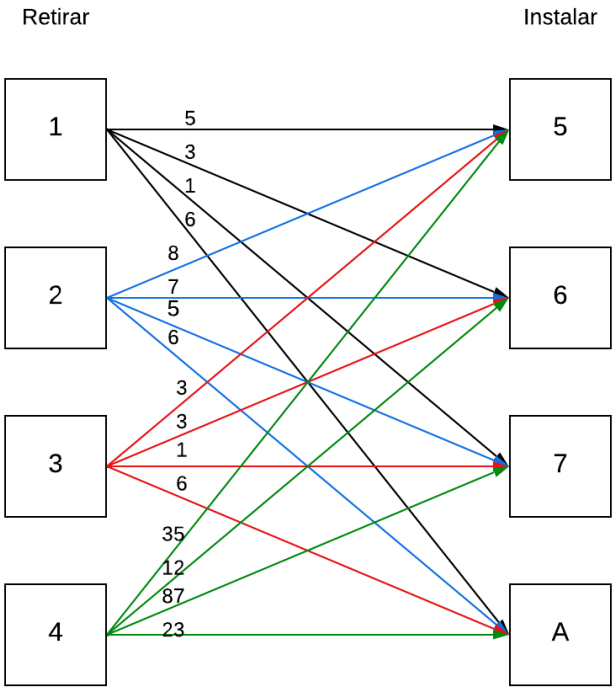

Figura 4.1 – Grafo representando as trocas de ferramentas.

Conforme ilustrado na Figura 4.1, considerando que a diferença cardinal entre os conjuntos era de apenas uma unidade, foi adicionada uma única ferramenta artificial. Os pesos das arestas artificiais conectadas a essa ferramenta representarão os custos de *setup* das ferramentas reais. Com essa configuração, torna-se viável calcular o custo aproximado por meio do LAP.

4.1.2 *Swap*

O movimento da vizinhança *swap*, como mostrado na Figura 4.2, envolve a seleção de um par de tarefas para realizar a troca. A escolha é feita da seguinte maneira: dado j_i, j_{i+1} é

substituída por uma tarefa que compartilhe o máximo de ferramentas com j_i. Neste exemplo, j_2 e j_5 são selecionados, e suas posições no sequenciamento são trocadas.

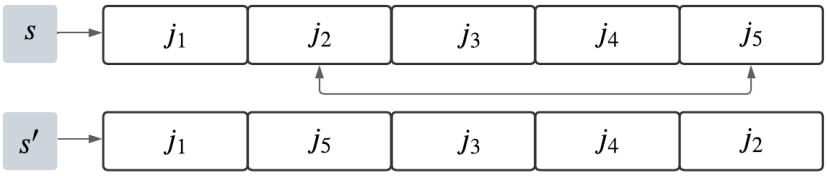

Figura 4.2 – *Swap*.

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.

4.1.3 *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 4.3, 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.

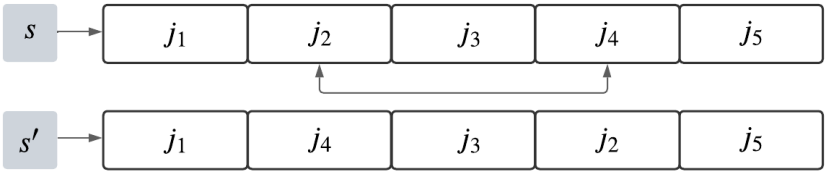

Figura 4.3 – *2-opt*.

Com relação à complexidade, a a vizinhança em questão tem um custo mais elevado para explorar a vizinhança em comparação com a *swap*. A complexidade é limitada $O(n^2)$, em que n representa a quantidade de tarefas.

4.1.4 *Insertion*

O movimento da vizinhança *insertion* envolve a seleção de uma tarefa existente no sequenciamento, seguida pela sua inserção em uma posição determinada por similaridade. A escolha é realizada da seguinte maneira: dado j_i, a tarefa que compartilha o máximo de ferramentas com j_i é inserida em j_{i+1}. Como ilustrado na Figura 4.4, ao aplicar um movimento *insertion*, a operação consiste em inserir a tarefa j_1 na posição 3.

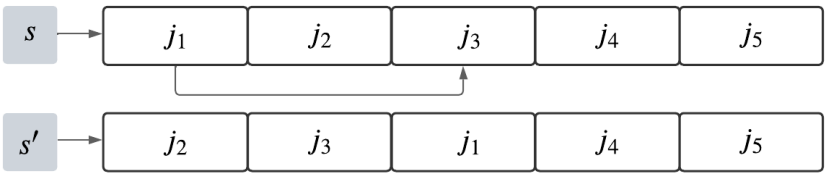

Figura 4.4 – *Insertion*.

A vizinhança em questão possui uma complexidade de tempo limitada por $O(n^2)$, 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.

4.1.5 *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 4.1 e 4.2. Essa troca de colunas na matriz resulta na reordenação da sequência de processamento das tarefas, otimizando a utilização das ferramentas.

Tabela 4.1 – 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 4.2 – Posterior ao *1-block grouping*.

Tarefas	|	0	1	4	3	2
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

Por exemplo, na Tabela 4.1, 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 4.2. Com essa nova ordem das tarefas, é possível notar que a ferramenta 2 permanecerá no *magazine* durante o processamento das tarefas j_0, j_1 e j_2. A complexidade desta busca local é $O(m^2n)$, em que n é a quantidade de tarefas e m é a quantidade de ferramentas.

4.2 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, Mara e Norcahyo (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.

4.2.1 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 $(u, v) \in A$ 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.

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.

4.2.2 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 4.5, baseada na ideia da busca local *swap* mencionada anteriormente e aplicada ao SP.

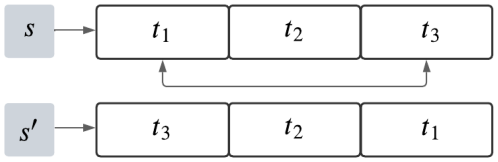

Figura 4.5 – Movimento da busca local.

A Figura 4.5 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 4.6.

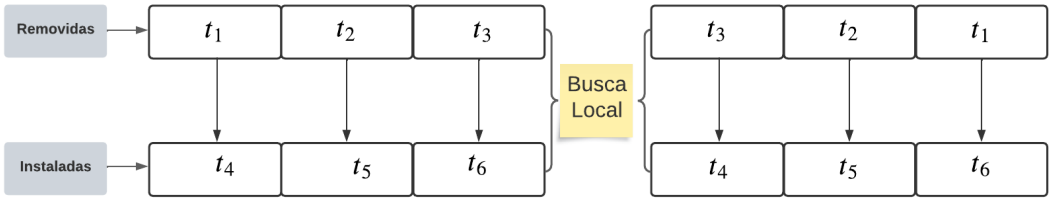

Figura 4.6 – Resultado após a busca local.

A Figura 4.6 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.

5 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, Mara e Norcahyo (2022). Utilizou-se a implementação do algoritmo *cost-scaling push-relabel* disponível no o pacote *open-source or-tools*.

5.1 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 5.1 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 β lista os possíveis percentuais das vizinhança a serem explorados pelas buscas locais, a linha α apresenta as opções de percentual de perturbação de uma solução. Por fim, a linha *Lista Tabu* indica o tamanho máximo da lista tabu.

Tabela 5.1 – 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, o número máximo de iterações da busca tabu foi definido como 100. O código e os

resultados estão disponíveis *online* no repositório[1], acompanhados por um validador de soluções desenvolvido pelo aluno Pedro Damasceno.

5.2 Comparação com o estado da arte

Denotando a busca tabu desenvolvida neste estudo como *BT* e o método proposto por Rifai, Mara e Norcahyo (2022) como *ALNS*, o valor de $gap(\%)$ foi calculado usando a fórmula $100 \times \frac{\text{BT-ALNS}}{\text{ALNS}}$. Assim como Rifai, Mara e Norcahyo (2022), que limitou suas execuções a 10 minutos, o mesmo limite de tempo foi adotado neste estudo. A Tabela 5.2 apresenta a comparação entre a *math-heuristic* BT + exato e os resultados obtidos por Rifai, Mara e Norcahyo (2022). Na referida tabela, a coluna *ALNS* representa o melhor resultado encontrado por Rifai, Mara e Norcahyo (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.

Tabela 5.2 – 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	581,00	**613,75**	0,05	-30,89	574,00	**613,68**	0,03	-30,90
J50C30	653,50	682,83	469,00	**545,00**	0,08	-20,19	461,00	**532,05**	0,07	-22,08
J50C35	496,00	536,03	425,00	**465,50**	0,08	-13,16	417,00	**464,65**	0,06	-13,32
J50C40	438,75	459,33	383,00	**441,03**	0,11	-3,98	380,00	**421,65**	0,08	-8,2
J60C30	1.300,75	1.334,23	797,00	**839,68**	0,04	-37,07	797,00	**838,80**	0,03	-37,13
J60C40	791,75	824,25	601,00	**665,60**	0,10	-19,25	598,00	**656,75**	0,07	-20,32
J60C50	542,00	569,08	467,00	**529,30**	0,09	-6,99	457,00	**521,40**	0,07	-8,38
J70C40	1.944,25	2.007,55	1.131,00	**1.170,53**	0,03	-41,69	1.128,00	**1.159,20**	0,02	-42,26
J70C50	1.209,75	1.247,38	829,00	**894,53**	0,06	-28,29	827,00	**877,05**	0,05	-29,69
J70C60	809,00	839,25	673,00	**718,10**	0,06	-14,44	670,00	**715,75**	0,06	-14,72

Analisando a Tabela 5.2, conclui-se que a *math-heuristic* BT + exato foi eficiente, entregando resultados competitivos dentro do tempo limite de 10 minutos estipulados por Rifai, Mara e Norcahyo (2022). Foi observado um *gap* mínimo de -8,2% em uma instância menos complexa e um *gap* máximo de -42,26% 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 -3,98% em uma instância menos complexa e de -41,69% na instância mais difícil.

[1] <https://gabao_06@bitbucket.org/gabao_06/ssp.git>

Tabela 5.3 – Resultados obtidos sem limite de tempo com a busca tabu e o método exato.

	Rifai, Mara e Norcahyo (2022)		*BT + exato*				
Grupo	*ALNS*	μ	*BT*	μ	*T*	σ	**gap**(%)
J50C25	850,50	888,08	552	**572,3**	5.109,16	0,01	-35,56
J50C30	653,50	682,83	445	**492,8**	2.990,91	0,01	-27,83
J50C35	496,00	536,03	403	**426,55**	2.402,13	0,02	-20,42
J50C40	438,75	459,33	351	**386,68**	1.928,28	0,02	-15,82
J60C30	1.300,75	1.334,23	777	**801,08**	8.873,25	0,01	-39,96
J60C40	791,75	824,25	570	**594,28**	4.761,66	0,01	-27,9
J60C50	542,00	569,08	430	**466,2**	2.897,57	0,02	-18,08
J70C40	1.944,25	2.007,55	1.091	**1.115,53**	19.788,56	0,01	-44,43
J70C50	1.209,75	1.247,38	782	**813,08**	10.228,53	0,01	-34,82
J70C60	809,00	839,25	630	**644,13**	5.410,93	0,01	-23,25

A Tabela 5.3 exibe os resultados obtidos ao término da execução do método, sem restrições de tempo, com o objetivo de avaliar sua convergência. Observou-se um *gap* mínimo de -20,42% em uma instância menos complexa e um máximo de -44,43% em uma instância mais desafiadora. Ao analisar a convergência do método, torna-se evidente que foram alcançadas soluções competitivas dentro do limite de tempo estabelecido. As melhorias ao longo da execução, no entanto, totalizam apenas 2,17% no conjunto de instâncias mais complexas, ao longo de aproximadamente 5 horas e 30 minutos.

A Tabela 5.4 apresenta a comparação entre BT + busca local e os resultados obtidos por Rifai, Mara e Norcahyo (2022), em que a coluna *ALNS* representa o melhor resultado encontrado por Rifai, Mara e Norcahyo (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 5.4 – Resultados obtidos com a busca tabu e a busca local.

	Rifai, Mara e Norcahyo (2022)		*BT + busca local*				
Grupo	*ALNS*	μ	*BT*	μ	*T*	σ	**gap**(%)
J50C25	850,50	888,08	562	**598,18**	23,73	0,01	-32,64
J50C30	653,50	682,83	466	**509,85**	25,33	0,01	-25,33
J50C35	496,00	536,03	409	**433,35**	24,24	0,02	-19,16
J50C40	438,75	459,33	355	**391,9**	15,47	0,02	-14,68
J60C30	1.300,75	1.334,23	818	**839,38**	48,66	0,01	-37,09
J60C40	791,75	824,25	584	**612,35**	40,8	0,01	-25,71
J60C50	542,00	569,08	438	**469,45**	36,55	0,02	-17,51
J70C40	1.944,25	2.007,55	1.149	**1.175,5**	82,06	0,01	-41,45
J70C50	1.209,75	1.247,38	806	**844,15**	71,34	0,01	-32,33
J70C60	809,00	839,25	634	**653,55**	65,38	0,01	-22,13

Ao analisar a Tabela 5.4, constata-se que a abordagem BT + busca local obteve resultados competitivos dentro do prazo estabelecido de 10 minutos, conforme mencionado por Rifai, Mara e Norcahyo (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 14% do tempo limite para todos os grupos de instâncias.

Ao comparar as Tabelas 5.2 e 5.4, percebe-se uma melhoria nos resultados obtidos, alcançando uma diferença de até 42,26% em relação ao conjunto completo de instâncias. A metodologia BT + busca local se destaca ao apresentar uma convergência significativamente mais rápida, atingindo soluções competitivas em um tempo médio de 43,35 segundos, o que representa 7,2% do tempo limite. Além disso, o BT + exato obteve um *gap* de -1,36% em comparação ao BT + busca local no maior grupo de instâncias, mas demandou 10 minutos. É relevante destacar a consistência dos métodos, evidenciada pelo baixo desvio padrão apresentado.

Realizou-se adicionalmente uma análise estatística para comparações de BT + busca local, BT + exato e ALNS. A Tabela 5.5 apresenta o resultado do teste de normalidade *Shapiro-Wilk*, introduzido por Shapiro e Wilk (1965), para todos os métodos. Os valores médios apresentados na Tabela 5.2 foram considerados como dados de entrada.

Tabela 5.5 – Teste de normalidade *Shapiro-Wilk*.

Método	W	p-*value*
ALNS	0,8679	0,0945
BT + busca local	0,8956	0,1970
BT + exato (10min)	0,9216	0,3701
BT + exato	0,9013	0,2265

A Tabela 5.5, confirma a hipótese nula de que os resultados de ALNS, BT + busca local e BT + exato podem ser modelados de acordo com uma distribuição normal, pois todos os métodos apresentam p-*value* superior 0,05. Assim, o método paramétrico *Student's t-test*, introduzido por Student (1908), foi utilizado para comparar os conjuntos de soluções. A seguir, a Tabela 5.6 exibe os resultados do *t-test* entre cada par de métodos.

Tabela 5.6 – Teste de normalidade *Shapiro-Wilk*.

	ALNS	BT + busca local	BT + exato (10min)
BT + exato (10min)	0,0019		
BT + busca local	0,0053		0,0026
BT + exato	0,0020	0,0021	4.93e-07

Analisando a Tabela 5.6, é possível concluir que o *t-test* indicou diferenças estatisticamente significativas entre os resultados de cada método. Observou-se que tanto o BT + exato até o fim da execução quanto o BT + exato registrado em 10 minutos (limite de tempo estipulado) e o BT + busca local demonstraram ser estatisticamente superiores ao método ALNS. Além disso, ao comparar as soluções geradas por BT + exato registrado em 10 minutos de execução com o BT + busca local, os resultados indicam que o segundo método apresenta um desempenho superior. No entanto, ao permitir que o método exato execute até o final, o BT + exato se mostra superior.

6 Conclusão

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 eficientes para abordar esse problema: uma meta-heurística que combina busca tabu com busca local e uma *math-heuristic* de busca tabu hibridizada com um método exato. Além disso, foi introduzida uma heurística construtiva e operadores especializados, contribuindo para aprimorar a eficiência desses métodos. 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 44,43% no maior grupo de instâncias.

A *math-heuristic* demonstrou eficiência, produzindo resultados competitivos no prazo estipulado de 5 minutos e exibindo uma boa convergência, em que já havia alcançado uma solução muito próxima daquela obtida em 10 minutos, atingindo uma diferença mínima de apenas 0,57% no grupo de instâncias mais complexo. Além disso, ao considerar o desvio padrão, observamos que o método é consistente, apresentando uma média de desvio padrão de 0,05. Ao observar a *math-heuristic* sem levar em consideração a restrição de tempo, nota-se uma redução média ainda menor no desvio padrão, atingindo 0,01, e uma melhoria de 2,17% em relação ao *gap* relativo a melhor solução conhecida até então. Entretanto, é importante mencionar que esse desempenho superior demandou aproximadamente 5 horas e 30 minutos em média para a conclusão da execução do método para o grupo de instâncias mais complexo.

Ao analisar o desempenho da busca tabu hibridizada com busca local, busca local, é notável a rápida convergência do método, alcançando soluções finais em apenas 7,2% do tempo limite. Os resultados mantêm uma boa qualidade, apresentando uma diferença inferior de apenas 0,81% em relação a *matheuristic*. Além disso, o método demonstra consistência, evidenciada pelo desvio padrão médio de 0,01.

Considerando as duas configurações disponíveis, pode-se concluir que, na ausência de restrições de tempo, a *math-heuristic* produz soluções de melhor qualidade, embora essa vantagem seja apenas margina. No entanto, quando o tempo de execução é uma restrição, a abordagem hibridizada com busca local demonstra maior eficiência.

Referências

BARD, J. F. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, Taylor & Francis, v. 20, n. 4, p. 382–391, 1988.

CALMELS, D. The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends. *International Journal of Production Research*, Taylor & Francis, v. 57, n. 15-16, p. 5005–5025, 2019.

CHERKASSKY, B. V.; GOLDBERG, A. V. On implementing push-relabel method for the maximum flow problem. In: BALAS, E.; CLAUSEN, J. (Ed.). *Integer Programming and Combinatorial Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. p. 157–171. ISBN 978-3-540-49245-0.

CHERNIAVSKII, M.; GOLDENGORIN, B. *An almost linear time complexity algorithm for the Tool Loading Problem*. 2022.

CRAMA, Y.; KOLEN, A. W. J.; OERLEMANS, A. G.; SPIEKSMA, F. C. R. Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, v. 6, p. 33–54, 1994.

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

GOLDBERG, A. V. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, v. 22, n. 1, p. 1–29, 1997. ISSN 0196-6774.

GOLDBERG, A. V.; TARJAN, R. E. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, v. 15, n. 3, p. 430–466, 1990.

GOLDBERG, A. V.; TARJAN, R. E.; HED, S. a. Minimum-cost flows in unit-capacity networks. *Theory of Computing Systems*, v. 61, n. 4, p. 1433–0490, 2017.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, v. 2, n. 1-2, p. 83–97, 1955.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016.

MARTELLO, S.; TOTH, P. Linear assignment problems. *Surveys in Combinatorial Optimization*, Elsevier, v. 31, p. 259–282, 2011.

PRIVAULT, C.; FINKE, G. Modelling a tool switching problem on a single nc-machine. *Journal of Intelligent Manufacturing*, v. 6, n. 2, p. 87–94, 1995. ISSN 1572-8145.

RIFAI, A. P.; MARA, S. T. W.; NORCAHYO, R. A two-stage heuristic for the sequence-dependent job sequencing and tool switching problem. *Computers & Industrial Engineering*, Elsevier, v. 163, p. 107813, 2022.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, [Oxford University Press, Biometrika Trust], v. 52, n. 3/4, p. 591–611, 1965. ISSN 00063444.

SMITH, J.; JOHNSON, A.; BROWN, M. Optimization of automotive painting process for improved efficiency and quality. *International Journal of Advanced Manufacturing Technology*, v. 18, p. 4291–4305, 2018.

STUDENT. The probable error of a mean. *Biometrika*, [Oxford University Press, Biometrika Trust], v. 6, n. 1, p. 1–25, 1908. ISSN 00063444.

TANG, C. S.; DENARDO, E. V. Models arising from a flexible manufacturing machine, part i: Minimization of the number of tool switches. *Operations Research*, v. 36, n. 5, p. 767–777, 1988.