Revista Brasileira de Computação Aplicada, Novembro, 2025

DOI: 10.5335/rbca.v17i3.16997
Vol. 17, Nº 3, pp. 64–73
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

Geração eficiente de soluções iniciais de alta qualidade para o problema de minimização de trocas de ferramentas

Efficient generation of high-quality initial solutions for the job sequencing and tool switching problem

Clark Cerqueira Engelhardt Veronez [1] and Leonardo Cabral da Rocha Soares [1]

[1]Instituto Federal do Sudeste de Minas Gerais

*clark.veronez.2023004668@estudante.ifsudestemg.edu.br, †leonardo.soares@ifsudestemg.edu.br

Recebido: 28/04/2025. **Revisado:** 16/11/2025. **Aceito:** 30/11/2025.

Resumo

Como problema de otimização, o *problema de minimização de trocas de ferramentas* retém diversos estudos no campo da pesquisa operacional sobre suas diferentes variações, enfatizado por sua relevância acadêmica e industrial. Embora os métodos atuais para solução deste problema reportem soluções de altíssima qualidade, o tempo computacional requerido tem se mostrado proibitivo frente ao aspecto prático do problema. Assim, neste trabalho, apresenta-se um método para geração de soluções válidas de alta qualidade em baixo tempo computacional, que possam ser utilizadas como soluções iniciais por métodos mais robustos, visando acelerá-los e contribuir para a qualidade final das soluções. A abordagem proposta consiste em uma nova implementação do método de descida randômica em vizinhança variável contendo buscas locais tradicionais e especializadas. Para geração do ponto inicial de exploração do método proposto, foram consideradas cinco heurísticas propostas originalmente para o problema do caixeiro viajante. Os resultados obtidos foram comparados com uma estratégia recente da literatura para geração de soluções iniciais e demonstraram melhoria significativa. Adicionalmente, o método proposto foi comparado ao atual estado da arte para o problema tratado e reportou um *gap* médio de apenas 5,36%, evidenciando a alta qualidade das soluções reportadas para o objetivo proposto.

Palavras-Chave: Descida em vizinhança variável; Heurísticas construtivas; Manufatura flexível; Otimização combinatória; Trocas de ferramentas.

Abstract

As an optimization problem, the *job sequencing and tool switching problem* has been the subject of several studies in operations research on its different variations, emphasizing its academic and industrial relevance. Although current methods approaching this problem yield extremely high-quality solutions, the computational time required has proven prohibitive when considering the practical aspects of the problem. Thus, in this paper, a method is presented for generating valid, high-quality solutions in low computational time, which can be used as initial solutions by more robust methods, aiming to accelerate them and contribute to the final quality of the solutions. The proposed approach consists of a new implementation of the random variable-neighborhood descent method using traditional and tailored local searches. Five traveling salesman problem heuristics were considered to generate the initial exploration point for the proposed method. The results obtained were compared with a recent strategy in the literature for generating initial solutions, which demonstrated significant improvement. Additionally, the proposed method was compared to the current state-of-the-art method for the addressed problem, and an average gap of only 5.36% was reported, evidencing the high quality of the solutions achieved for the proposed objective.

Keywords: Combinatorial optimization; Constructive heuristics; Flexible manufacturing; Tool switching; Random Variable Neighborhood Descent.

1 Introdução

Inicialmente conceituados para usinagem, os Sistemas de Manufatura Flexíveis (SMFs) tomaram direcionamento significativo para as indústrias modernas a partir dos anos 1980, atenuando problemas de *trade off* entre produção veloz e qualidade, frente à perspectiva fordista de manufatura e ascendendo campos de estudo tecnológicos influenciados pelos problemas advindos deste novo formato.

Um SMF pode ser usualmente caracterizado por um conjunto de máquinas-ferramenta (ou máquinas flexíveis), computacionalmente controlado e interligado por um sistema automático de manuseio de materiais, visando automatizar o processo produtivo de larga escala sem demandar linhas adicionais de produção (Silveira, 2019). Desta forma, diferentes tarefas podem ser processadas por uma mesma máquina-ferramenta, desde que as ferramentas necessárias para o processamento destas sejam previamente alocadas à máquina. Cada máquina-ferramenta possui um *magazine* responsável por conter as ferramentas dispostas para utilização durante o processamento de uma determinada tarefa (Mancio and Sellitto, 2017). Entretanto, o número total de ferramentas disponíveis, em geral, é superior a capacidade do *magazine*.

Ao se analisar o gerenciamento de um SMF, nota-se a intrínseca necessidade de realizar periódicas trocas de ferramentas no *magazine* das máquinas visando o processamento sequencial de um conjunto de tarefas. Dado o alto custo que estas trocas de ferramentas acarretam aos SMFs, cerca de 25% a 30% dos custos fixos e variáveis (Beezão et al., 2017), esta observação impele diversos estudos da literatura a se interessarem pelos problemas que buscam minimizar o número total de trocas de ferramentas em uma cadeia produtiva de um SMF.

Neste contexto, Tang (1986) introduz o problema de minimização de trocas de ferramentas (ou *job sequencing and tool switching problem*, SSP). Um problema $\mathcal{NP}$-Difícil (Crama et al., 1994), ou seja, para o qual não se conhece algoritmo determinístico polinomial para sua solução, que consiste em definir uma ordem para o processamento de um conjunto de tarefas por uma máquina flexível e o seu posterior *plano de trocas de ferramentas*, visando minimizar o número de trocas de ferramentas necessárias.

Desde seu trabalho seminal, o SSP tem sido amplamente estudado na literatura (Calmels, 2019). Entre as principais abordagens ao problema, destacam-se soluções baseadas em meta-heurísticas. De forma geral, estes métodos partem de uma solução inicial (ou uma população de soluções iniciais) e empregam componentes de intensificação e diversificação com o intuito de aprimorar a qualidade das soluções obtidas. Por conseguinte, a geração de boas soluções iniciais é de extrema importância para a progressão do estado da arte deste problema.

Diante disto, neste trabalho, propõe-se um método de descida randômica em vizinhança variável (*random variable neighborhood descent*, RVND), introduzido por Subramanian et al. (2010), para geração de soluções que possam ser utilizadas como soluções iniciais por outros métodos. O RVND foi escolhido por permitir a combinação de múltiplas estruturas de vizinhanças, em ordem aleatória, em um tempo computacional aceitável, o que garante diversidade e velocidade na construção das soluções iniciais. Por não possuir estratégias de diversificação que permitam escapar de ótimos locais não globais, não é esperado que o RVND reporte soluções competitivas em relação ao estado da arte, entretanto, a utilização destas soluções como ponto inicial por método baseados em trajetória, ou para geração de indivíduos utilizados por meta-heurísticas populacionais, pode contribuir para diminuir o tempo computacional demandado por estes métodos ou para melhorar os resultados alcançados.

O método proposto é avaliado em uma extensa campanha experimental. Os resultados obtidos são comparados com uma heurística proposta para geração de soluções iniciais por Soares (2023) e apresentam um avanço significativo. Adicionalmente, para confirmar a qualidade das soluções reportadas pelo método proposto, os resultados são comparados aos reportados pelo atual estado da arte para o problema.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2, apresenta-se a descrição formal do problema. A revisão da literatura sobre o SSP é exposta na Seção 3. Na Seção 4, os métodos propostos são detalhados. A Seção 5, apresenta os experimentos computacionais realizados e os resultados alcançados. Por fim, na Seção 6, apresenta-se as conclusões e os trabalhos futuros.

2 Descrição do problema

Introduzido formalmente por Tang (1986), o SSP ocorre em um SMF contendo uma única máquina flexível e consiste em determinar uma ordem para o processamento de um conjunto $T = \{1, \ldots, n\}$ de tarefas, bem como definir as ferramentas que deverão ser inseridas ou removidas na máquina antes do início do processamento de cada tarefa, visando minimizar a quantidade total de trocas de ferramentas. A máquina flexível possui um conjunto $F = \{1, \ldots, l\}$ de ferramentas que podem ser utilizadas. Cada tarefa i demanda um subconjunto F_i ($F_i \in F$) de ferramentas para ser processada. Durante o processamento de uma determinada tarefa, as ferramentas em uso são armazenadas em um compartimento denominado *magazine*. A capacidade C do *magazine* é suficiente para conter todas as ferramentas necessárias para o processamento de uma tarefa isolada, ou seja, $C \geq |F_i|$. Entretanto, em geral, o conjunto total de ferramentas disponíveis é superior a C, logo, considerando-se o processamento sequencial das tarefas, faz-se necessário efetuar trocas de ferramentas no *magazine*. Embora o tempo para trocar as ferramentas dentro do *magazine* seja desprezível, as trocas de ferramentas entre a área de armazenamento e o *magazine* são custosas, assim, faz-se necessário minimizá-las com o objetivo de aumentar a eficiência da máquina flexível.

São numerosas as variações do SSP (Calmels, 2019), dado o contexto industrial ao qual está inserido, todavia, este trabalho se preocupa em abordar soluções para sua versão padrão, também conhecido como SSP *uniforme*, que considera algumas pressuposições para o plano de trocas de ferramentas (Bard, 1988; Crama et al., 1994): (*i*) cada ferramenta ocupa uma posição única no *magazine*; (*ii*) apenas uma ferramenta deve ser trocada por vez; (*iii*) o tempo para remover ou inserir uma ferramenta é constante, independente de qual seja; (*iv*) tem-se conhecimento do

conjunto e subconjunto de ferramentas com antecedência; (*v*) não são consideradas trocas por avarias ou desgastes; e, (*vi*) uma ferramenta só precisa ser removida se for substituída por outra.

Uma instância para o SSP pode ser representada através de uma matriz binária W, tendo cada linha da matriz como uma tarefa e cada coluna como uma ferramenta. Para toda ferramenta j necessária ao processamento da tarefa i, toma-se $w_{ij} = 1$, caso contrário, $w_{ij} = 0$. A Tabela 1 apresenta a matriz binária para uma instância arbitrária do SSP em que considera-se $n = 5$, $F = \{1, \ldots, 9\}$ e $C = 7$.

Tabela 1: Instância arbitrária do SSP

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

A solução para uma determinada instância é obtida através da permutação π das colunas do *plano de trocas de ferramentas*. O plano de trocas de ferramentas pode ser representado por uma matriz binária $Q^{\pi} = \{q_{ij}\}$ sendo $i \in T$ e $j \in F$. As colunas do plano de trocas de ferramentas representam o estado do *magazine* durante o processamento de uma determinada tarefa, assim, um elemento $q_{ij} = 1$ indica a presença da ferramenta no *magazine*, enquanto $q_{ij} = 0$, o contrário. A ordem das colunas é definida de acordo com a permutação π e indica a ordem de processamento das tarefas.

A Tabela 2 apresenta um plano de trocas de ferramentas para a instância introduzida na Tabela 1. A ordem para o processamento das tarefas foi definida arbitrariamente como $\pi = \{1, 2, 3, 4, 5\}$. Sublinhados denotam ferramentas inseridas ou removidas imediatamente antes do processamento da tarefa corrente. A primeira tarefa a ser processada (tarefa 1) requer apenas quatro ferramentas. Dada a capacidade do *magazine* ($C = 7$), ferramentas adicionais, necessárias para o processamento das próximas tarefas são carregadas. Assim, antes do início do processamento da tarefa 1, o *magazine* da máquina foi carregado com as ferramentas $\{2, 3, 4, 5, 6, 7, 9\}$, sendo $\{4, 9\}$ necessários para o processamento da tarefa 2 e $\{7\}$ para o da tarefa 3, gerando sete trocas de ferramentas. Caso não houvesse tamanho suficiente no *magazine* para conter as ferramentas $\{4, 9\}$ durante o processamento da primeira tarefa, antes que a tarefa 2 pudesse ser processada, tais ferramentas deveriam ser inseridas no *magazine*. Antes do processamento da tarefa 3, fez-se necessário inserir a ferramenta $\{1\}$, o que gerou a exigência de desalocar uma das ferramentas presentes no *magazine*. Entre as ferramentas candidatas a remoção (ferramentas que não são requeridas pela tarefa atual), seleciona-se aquelas que serão mais tardiamente necessárias, de acordo com a política *Keep tool needed soonest, KTNS*, introduzida por Tang and Denardo (1988). Neste exemplo, a ferramenta $\{6\}$ foi selecionada e removida, resultando em uma troca de ferramenta antes do processamento da tarefa 3. Para o processamento sequencial da tarefa 4, todas as ferramentas necessárias já estavam previamente carregadas. Por fim, antes do processamento da tarefa 5, as ferramentas $\{6, 8\}$ foram inseridas no *magazine*, enquanto as ferramentas $\{3, 4\}$ foram removidas, resultando em mais duas trocas de ferramentas. Ao final, para o processamento de todas as tarefas na ordem exemplificada, foram necessárias 10 trocas de ferramentas.

Tabela 2: Exemplo de plano de trocas de ferramentas.

Ferramentas	Tarefas				
	1	2	3	4	5
1	0	0	$\underline{1}$	1	1
2	$\underline{1}$	1	1	1	1
3	$\underline{1}$	1	1	1	$\underline{0}$
4	$\underline{1}$	1	1	1	$\underline{0}$
5	$\underline{1}$	1	1	1	1
6	$\underline{1}$	1	$\underline{0}$	0	$\underline{1}$
7	$\underline{1}$	1	1	1	1
8	0	0	0	0	$\underline{1}$
9	$\underline{1}$	1	1	1	1

O plano de trocas de ferramentas pode ser avaliado pela Eq. (1), proposta por Crama et al. (1994). Esta equação calcula o número de inversões de zero para um na matriz binária. Para que o carregamento inicial de ferramentas seja contabilizado, faz-se necessário inserir uma coluna artificial à matriz, em que todos os elementos são iguais à 0, ou seja, $R^{\pi}_{t0} = 0$.

$$Z_{SSP}(R^{\pi}) = \sum_{t \in T} \sum_{f \in F} r^{\pi}_{tf}(1 - r^{\pi}_{tf-1}) \tag{1}$$

Tendo por objetivo minimizar o número total de trocas de ferramentas, a função objetivo do SSP pode ser escrita conforme Eq. (2), sendo Π o conjunto de todas as possíveis permutações para o processamento das tarefas.

$$\min_{\pi \in \Pi} \{Z^{\pi}_{SSP}(R)\} \tag{2}$$

3 Revisão da Literatura

A exposição incoativa realizada pelo trabalho Tang and Denardo (1988), a qual formalizou as bases para o SSP e introduziu a política KTNS, tem, desde a década de 1980, impulsionado novos trabalhos interessados no SSP e suas diversas variações (Soares and Carvalho, 2020; Calmels, 2022; Cura, 2023; Rifai et al., 2022). Uma revisão detalhada da literatura destes estudos pode ser obtida a partir do trabalho de Calmels (2019). Segue-se, a partir dos parágrafos seguintes, um panorama das principais contribuições ao SSP em sua versão padrão (também nomeado na literatura como *SSP Uniform*), cronologicamente organizado.

Conforme mencionado, o trabalho de Tang and Denardo (1988), define as bases do SSP. O problema é modelado e resolvido como uma instância do *Traveling Salesman Problem*

(TSP), ou problema do caixeiro viajante. No mesmo artigo, os autores desenvolvem a política KTNS que permite calcular em tempo determinístico polinomial a solução ótima para o número de trocas de ferramentas para uma dada sequência fixa de tarefas a serem processadas. No mesmo ano, Bard (1988) modelou o SSP como um problema de programação linear não inteira e utilizou um método de *branch-and-bound* para resolver uma relaxação Lagrangeana. Com o intuito de minimizar o tempo de execução ao adquirir boas soluções, o número de iterações foi limitado, reportando-se soluções que representam ótimos locais.

Gray et al. (1993) discutem sobre a elaboração de modelos de decisão para gerenciamento de ferramentas no contexto do SSP. Crama et al. (1994) comprovam que o SSP como pertencente à classe de problemas $\mathcal{NP}$-difícil para qualquer instância com $C \geq 2$. Novamente o problema é abordado como instância do TSP. Embora seja evidente a recorrência desta modelagem na literatura (Ahmadi et al., 2018; Djellab et al., 2000; Hertz et al., 1998; Laporte et al., 2004; Shirazi and Frizelle, 2001), Yanasse and Lamosa (2006) reportam a ineficiência de tal abordagem, dada a não equivalência entre o SSP e o TSP.

Laporte et al. (2004) propõe um novo modelo de programação linear inteira que apresenta melhores valores de relaxação linear ante o modelo de Tang and Denardo (1988). Algoritmos de *branch-and-bound* e *branch-and-cut* utilizando instâncias próprias do trabalho também são apresentados para efeito de comparação e análise. Respectivamente, um demonstrou aptidão para resolver de forma ótima instâncias com até 9 tarefas, enquanto o outro instâncias com até 25 tarefas.

Yanasse et al. (2009) apresentaram um algoritmo enumerativo capaz de superar os resultados obtidos por Laporte et al. (2004). Senne and Yanasse (2009) apresentam três implementações do algoritmo *beam search* baseadas em enumeração, entretanto, os resultados não são comparados a outros métodos da literatura.

Um método híbrido que mescla um algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm*, BRKGA) com um algoritmo de busca de agrupamento (*Clustering Search*, CS) foi apresentado por Chaves et al. (2016). Os experimentos computacionais utilizaram as instâncias propostas por Yanasse et al. (2009) e Crama et al. (1994), reportando melhores resultados para todos os problemas.

A exploração da relação entre o SSP e problema de minimização de blocos de uns consecutivos (*Consecutive block minimization*, CBM) forneceu recursos que contribuíram significativamente para os resultados de Paiva and Carvalho (2017). O trabalho apresenta uma heurística construtiva, uma nova representação do problema em grafos e uma nova implementação da meta-heurística busca local iterada (*iterated local search*, ILS) para o SSP. Os experimentos computacionais consideraram 1670 instâncias (Yanasse et al., 2009; Crama et al., 1994; Catanzaro et al., 2015) e reportaram valores equivalentes ou melhores para todos os problemas.

Apesar de não exposta a métrica para determinar as distâncias utilizadas na abordagem, Ahmadi et al. (2018) modelaram o SSP como uma instância do TSP de segunda ordem (2-TSP) e resolveram o problema utilizando um algoritmo genético *q-learning*. Os autores consideraram apenas um subconjunto das instâncias disponíveis (Catanzaro et al., 2015; Crama et al., 1994) e apresentam melhorias marginais ante o trabalho de Paiva and Carvalho (2017).

da Silva et al. (2021) apresentam uma nova formulação para o SSP adotando um modelo de multi-fluxo. Os experimentos utilizaram um subconjunto das instâncias disponíveis na literatura (Catanzaro et al., 2015; Yanasse et al., 2009). Os resultados obtidos superam os resultados reportados pelos modelos anteriores. No entanto, o modelo ainda não é capaz de resolver os problemas com maiores dimensões dentro do tempo limite de uma hora.

Mecler et al. (2021) apresentam um algoritmo de pesquisa híbrida (*Hybrid Genetic Search, HGS*) para o SSP. Além das instâncias tradicionais, um novo conjunto contendo 60 problemas de maiores dimensões é apresentado. Os resultados relatados demonstraram melhorias médias significativas para todos os conjuntos de instâncias, comparados aos resultados obtidos pelos métodos de Ahmadi et al. (2018), Chaves et al. (2016) e Paiva and Carvalho (2017).

Soares (2023) propõe uma nova implementação que explora a solução exata do CBM (Soares et al., 2020) como estratégia parcial para a solução do SSP. Apresenta-se uma heurística para geração de soluções iniciais, uma implementação da meta-heurística ILS e uma nova implementação do BRKGA. Os resultados obtidos foram comparados aos reportados por Mecler et al. (2021) e, embora o tempo computacional requerido seja substancialmente menor, os valores de solução não superam o trabalho anterior.

Almeida et al. (2025) apresentam uma implementação do método de revenimento paralelo (*Parallel Tempering*, PT) para abordagem ao SSP. Os experimentos computacionais consideraram apenas as maiores instâncias disponíveis(Mecler et al., 2021). Resultados melhores ou iguais foram reportados para 7 dos 12 subconjuntos das instâncias.

Entre as instâncias disponíveis na literatura, apenas os maiores problemas, propostos por Mecler et al. (2021) permanecem como um desafio frente à alta qualidade dos métodos propostos para solução do SSP. Assim, nos experimentos computacionais reportados na Seção 5, considera-se apenas este conjunto de instâncias. Como não há dominância de um único método sobre este problemas, os melhores valores de solução conhecidos (Mecler et al., 2021; Almeida et al., 2025) são considerados para aferir a qualidade do método proposto.

4 Métodos

Conforme mencionado, a abordagem do SSP como uma instância do TSP tem se mostrado ineficiente na literatura (Yanasse and Lamosa, 2006). Entretanto, esta modelagem é totalmente dependente da métrica utilizada para geração da matriz de distância. Assim, neste trabalho, examina-se o desempenho de heurísticas propostas originalmente para o TSP, aplicadas ao SSP com uma métrica específica, que considera as características intrínsecas ao SSP. Em seguida, a melhor solução reportada por qualquer destes métodos é aprimorada por buscas locais organizadas em um RVND. Nas próximas seções, as heurísticas selecionados e o método proposto são apresentados.

4.1 Heurísticas construtivas

De forma geral, uma heurística construtiva tem por objetivo alcançar uma solução válida pela adição iterativa de novos elementos a uma solução inicialmente vazia. A forma de escolha de cada elemento é definida de acordo com o problema abordado e com a função de avaliação considerada (Souza, 2008). No contexto deste trabalho, a cada iteração uma nova tarefa é adicionada a solução visando um escalonamento completo que minimize o número total de trocas de ferramentas. A seguir, são descritas as cinco heurísticas selecionadas como parte da abordagem para solução do SSP. Todas as heurísticas foram originalmente propostas para o TSP, entretanto, são descritas aqui já no contexto do problema abordado.

A tradicional heurística do *vizinho mais próximo*, a partir de uma dada tarefa inicial, escolhe como próxima tarefa a ser alocada aquela para a qual possuir a menor diferença (distância) considerando-se os requisitos de ferramentas. Todas as tarefas são consideradas como tarefa inicial. Para o cálculo da distância, considera-se o número de trocas de ferramentas obtido pelo KTNS sobre o escalonamento parcial. Esta estratégia resulta em uma distância mais precisa para o SSP pois considera o histórico de ferramentas do *magazine*. Dado que, $C \geq |F_i|$, é possível que ferramentas adicionais sejam previamente carregadas no *magazine*. Caso alguma destas ferramentas seja requerida pela próxima tarefa, isto resultará na diminuição da distância efetiva entre as tarefas. Eventuais empates são resolvidos de forma aleatória. Estas mesmas estratégias para construção da matriz de distâncias e para solução de empates são utilizadas em todas as heurísticas consideradas.

O *algoritmo de Bellmore e Nemhauser* (Bellmore and Nemhauser, 1968) utiliza de iterações para construir sub-rotas (soluções parciais), que seguem o critério de inserção em extremidades. Todas as tarefas são consideradas como possível tarefa inicial e, para cada tarefa ainda não alocada, avalia-se o custo de inserção (distância) em ambas as extremidades da sub-rota, ou seja, no início e no fim do escalonamento parcial. A tarefa com o menor custo é selecionada e adicionada à extremidade correspondente.

A heurística *inserção mais barata* considera uma sub-rota inicial envolvendo três tarefas. A cada iteração, calcula-se o custo da inserção de uma nova tarefa entre todos os pares de tarefas escalonados sequencialmente. A tarefa que resultar no menor custo é adicionada. Para geração da sub-rota inicial, todos possíveis trios de tarefas são considerados.

Inspirada no jogo de mesmo nome, a heurística *dominó* (Ismail, 2019), considera cada tarefa como uma *pedra* do jogo. Uma pedra é aleatoriamente selecionada e colocada sobre a mesa (tarefa inicial do escalonamento). As pedras restantes são aleatoriamente divididas entre dois jogadores. A cada turno (uma iteração do método), um jogador seleciona entre as suas pedras a que possuir maior semelhança (menor distância) para as pedras posicionadas em uma das extremidades da mesa, o que corresponde às extremidades do escalonamento parcial, e a adiciona ao jogo. O processo se repete, alternando-se os turnos entre os jogadores até que todas as pedras sejam posicionadas. Embora semelhante à heurística proposta por Bellmore and Nemhauser (1968), a divisão aleatória das tarefas entre os jogadores e o escalonamento em turnos garante que esta heurística obtenha resultados diferentes.

O método de *atribuição em termos de uns* (Basirzadeh, 2014) divide todos os elementos de uma mesma linha da matriz de distâncias pelo menor elemento desta linha, gerando pelo menos um elemento com valor igual a *um* em cada linha. O valor de cada menor elemento é armazenado e associado à linha correspondente. Em seguida, o mesmo processo é executado para cada coluna. A partir da normalização de linhas e colunas, a atribuição de uns se responsabiliza por descartar o menor número de linhas e colunas necessários para atribuir todos os uns independentes da matriz, e, caso o número mínimo de linhas a serem descartadas seja igual à ordem da matriz, tem-se a solução completa. Caso contrário, repete-se os processos anteriores sobre as linhas e colunas não cobertas. A atribuição ocorre por exclusão mútua, garantindo que um mesmo um não seja atribuído para linhas e colunas diferentes. O critério de prioridade ordena os valores mínimos armazenados durante a fase de normalização para escalonar as tarefas (a tarefa com o menor elemento marcado é inserida primeiro), construindo assim, uma solução completa.

4.2 Descida randômica em vizinhança variável

O método *descida em vizinhança variável* (Mladenović and Hansen, 1997), a partir de uma dada solução inicial, busca obter um ótimo local a partir de mudanças sistemáticas na estrutura de vizinhança explorada. Um conjunto de vizinhanças é abordado sequencialmente. Toda vez que a exploração resultar na melhoria da solução corrente, o processo se reinicia a partir da primeira estrutura de vizinhança considerada. Ao final do processo exploratório, um ótimo local comum a todas as vizinhanças exploradas é retornado.

Para uma maior eficiência, a ordem de exploração das vizinhanças consideradas pelo VND deve ser previamente definida. Este processo de calibração das vizinhanças pode ser computacionalmente custoso (Souza, 2008) e não há garantias que a ordem estabelecida seja a melhor opção para todas as instâncias. Assim, visando adotar uma estratégia que extirpasse do método seu caráter sequencialmente ordenatório, Subramanian et al. (2010) apresentam o *método de descida randômica em vizinhança variável* (*random variable neighborhood descent*, RVND). Esta nova versão do VND distingue-se do original apenas pela adoção de um processo randômico de ordenação das vizinhanças. Ao adotar uma estratégia que inclui aleatoriedade em seu processo exploratório, o RVND é capaz de encontrar soluções inalcançáveis pelo processo determinístico e rígido de seu predecessor.

Para abordagem ao SSP, a melhor solução entre as reportadas pelas heurísticas descritas na Seção 4.1 é utilizada como solução inicial para o RVND. Em seguida, são aplicadas as buscas locais *maior arrependimento*, *2-opt* e *or-opt* em ordem aleatória. Quando um ótimo local comum a estas vizinhanças é encontrado, o método encerra-se e a solução é reportada. As buscas locais utilizadas são apresentadas em detalhes nas seções que se seguem.

4.2.1 *Busca local maior arrependimento*

Utilizada em uma abordagem recente ao SSP (Soares and Carvalho, 2024), a busca local *maior arrependimento* classifica as tarefas de uma dada solução para o SSP de acordo com a quantidade de trocas de ferramentas ocasionadas por seu respectivo escalonamento. Em seguida, as tarefas que mais demandaram trocas de ferramentas são realocadas. Todas as possíveis posições para reinserção são consideradas e a tarefa é mantida na posição em que resultar no menor número de trocas de ferramentas. A classificação das tarefas garante que o método reposicione primeiramente as tarefas críticas, ou seja, aquelas que mais estavam contribuindo negativamente para a qualidade da solução.

4.2.2 *Busca local 2-opt*

O clássico método *2-opt*, originalmente proposto por Croes (1958) para o TSP, quando aplicado ao SSP resulta na inversão de um segmento da solução. Por exemplo, sendo o escalonamento original $[1, 2, 3, 4, 5]$ e, considerando-se como pontos para a realização do movimento, as posições 2 e 4, após a o movimento obteríamos a solução vizinha $[1, 4, 3, 2, 5]$. Após a realização de todos os movimentos possíveis, aquele que resultar na melhor solução é mantido.

4.2.3 *Busca local or-opt*

Também proposto originalmente para o TSP, a busca local *or-opt* (Or, 1976) quando aplicada ao SSP seleciona três blocos consecutivos de tarefas para realocação. Esta estratégia permite que novas soluções sejam geradas ao mesmo tempo em que se mantém a estrutura sequencial de cada bloco, permitindo que regiões não alcançáveis pelas buscas locais anteriores sejam exploradas. Assim como nas buscas locais anteriores, todos os possíveis movimentos são executados e a melhor solução é mantida.

5 Experimentos

Uma série de experimentos computacionais foi realizada para definir a versão final do método proposto, mensurar a qualidade das soluções reportadas e avaliar seu comportamento. O ambiente computacional utilizado consiste de um computador com processador *Apple M2* de 3.49 GHz com 8 GB de memória RAM sob o sistema operacional *macOS Sonoma*. Todos os métodos foram implementados em C++, compilados com GCC 15.0.0 e as opções de otimização -O3 e -march = native.

O SSP possui quatro conjuntos *benchmark* de instâncias disponíveis na literatura (Crama et al., 1994; Yanasse et al., 2009; Catanzaro et al., 2015; Mecler et al., 2021). Entretanto, frente à alta qualidade dos métodos atuais para solução do SSP, apenas o conjunto mais recente, contendo os maiores problemas, permanece como um desafio. Este conjunto, proposto por Mecler et al. (2021), possui 60 instâncias divididas em três subgrupos (F_1, F_2, F_3) contendo 20 instâncias cada. As instâncias do subgrupo F_1 possuem 50 tarefas, 75 ferramentas e capacidade do *magazine* variando entre 25 e 40. O subgrupo F_2 possui instâncias com 60 tarefas, 75 ferramentas e capacidade do *magazine* variando entre 35 e 50. O último subgrupo, F_3, possui problemas com 70 tarefas, 105 ferramentas e capacidade do *magazine* variando entre 40 e 55. Todas as instâncias propostas por Mecler et al. (2021) foram consideradas nos experimentos realizados.

5.1 Experimentos preliminares

Embora as heurísticas selecionadas sejam amplamente utilizadas em abordagens ao TSP, não há na literatura comparação direta entre elas no contexto do SSP. Assim, uma série de experimentos preliminares foram realizadas para aferir o comportamento destas heurísticas em comparação ao atual estado da arte para o SSP e para averiguar se há predominância de alguma sobre as outras, visando evitar o desperdício de tempo computacional com heurísticas que sistematicamente reportem piores resultados. As heurísticas foram individualmente executadas para todas as instâncias. Devido a aleatoriedade utilizada nos critérios de desempate, cada heurística foi executada 20 vezes. As instâncias foram agrupadas por tarefas (n), ferramentas (l) e capacidade do *magazine* (C) e os resultados médios comparados com o atual estado da arte, considerando-se os melhores resultados reportados pelos métodos de Mecler et al. (2021) e Almeida et al. (2025).

A Fig. 1 apresenta um gráfico *boxplot* comparando o *gap* médio reportado por cada heurística, a saber, vizinho mais próximo (1), Bellmore e Nemhauser (2), inserção mais barata (3), dominó (4) e atribuição em termos de uns (5). O *gap* foi calculado como $100 \times \frac{\text{solução heurística} - \text{Estado da arte}}{\text{Estado da arte}}$. Conforme pode ser visualizado no gráfico, as heurísticas vizinho mais próximo e Bellmore e Nemhauser dominam todas as outras em relação à qualidade das soluções obtidas, havendo reportado, respectivamente, um *gap* médio de 12, 84% e 12, 75%. Para estas duas heurísticas, o *boxplot* indica uma distribuição concentrada, com valores máximos e mínimos quartis próximos à mediana. A heurística vizinho mais distante reportou os piores resultados, apresentando um *gap* médio próximo a 70%. As demais heurísticas reportaram um *gap* médio próximo a 30%. Não havendo *outliers* ou casos específicos que justifiquem a utilização das heurísticas dominadas, apenas as heurísticas vizinho mais próximo e Bellmore e Nemhauser são comparadas para a geração da solução inicial utilizada pelo método RVND proposto, sendo utilizada a que reportar a melhor solução.

5.2 Avaliação do método proposto

Após a realização dos experimentos preliminares, definiu-se a versão final do método RVND proposto. Para aferir a qualidade das soluções reportadas, os resultados obtidos foram comparados ao método heurístico utilizado por Soares and Carvalho (2024) para geração de soluções iniciais. A Tabela 3 apresenta os resultados reportados por Soares and Carvalho (2024) e os resultados obtidos a partir de 20 execuções individuais do método proposto. Seguindo o padrão adotado por Mecler et al. (2021) os resultados são agrupados por número de tarefas (n), ferramentas (l) e capacidade do *magazine* (C). Além dos valores reportados pelo método de Soares and Carvalho (2024) (H_CBM), apresenta-se os resultados obtidos pelo método proposto

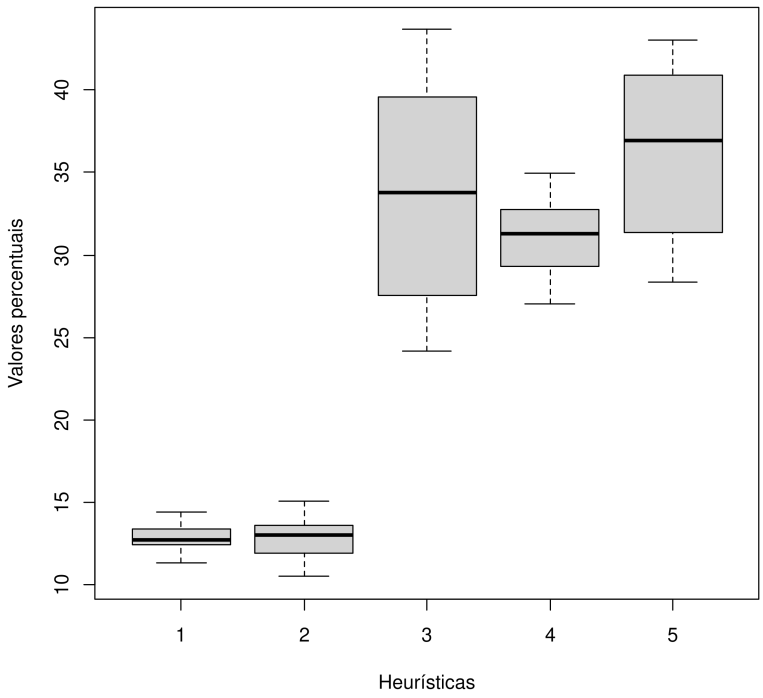

Figura 1: Gráfico *boxplot* dos *gaps* das heurísticas em comparação ao estado da arte.

(*RVND*) e o *gap* médio. Cada linha da tabela apresenta o resultado médio para um subconjunto contendo cinco instâncias. Os melhores valores são destacados em negrito.

n	l	C	H_CBM	*RVND*	*gap*(%)
50	75	25	331,80	**323,18**	-3,50
50	75	30	266,80	**253,57**	-6,30
50	75	35	222,00	**205,53**	-8,92
50	75	40	188,00	**169,22**	-11,49
60	90	35	515,80	**503,04**	-3,26
60	90	40	425,60	**404,00**	-6,34
60	90	45	358,80	**331,04**	-9,03
60	90	50	303,80	**274,90**	-10,80
70	105	40	704,20	**682,04**	-3,92
70	105	45	594,80	**563,25**	-6,25
70	105	50	510,20	**472,61**	-8,35
70	105	55	441,40	**376,00**	-10,60

Tabela 3: Comparação dos resultados obtidos com o método de Soares and Carvalho (2024)

Em comparação ao H_CBM, o método proposto reportou melhores soluções médias para todos os subconjuntos de instâncias, apresentando *gap* médio de -7,40%. Entre os *gaps* médios individuais, destaca-se o reportado para o subconjunto contendo os problemas de maiores dimensões (n = 50, l = 75, C = 40), -11,49%. O tempo demandado pelo método de Soares and Carvalho (2024) para geração das soluções iniciais não é reportado no artigo original, assim, para este experimento, não é possível comparar o tempo de execução dos métodos. Entre as heurísticas utilizadas pelo método proposto, a heurística do vizinho mais próximo foi utilizada em 51,5% das soluções e a heurística Bellmore e Nemhauser em 49,5%.

Para uma avaliação mais acurada do método proposto, os resultados obtidos são comparados ao atual estado da arte para o SSP. Conforme mencionado, os melhores resultados conhecidos para as instâncias aqui utilizadas variam entre os resultados reportados pelos métodos de Mecler et al. (2021) (HGS) e Almeida et al. (2025) (PT). Assim, nos experimentos aqui descritos, considera-se como valor de referência o melhor valor reportado por qualquer um destes métodos, calculado conforme demonstrado pela Eq. (3).

$$\min(HGS, PT) \tag{3}$$

A Tabela 4, seguindo o mesmo padrão definido para a tabela anterior, mostra a comparação entre o método proposto e o atual estado da arte (*EA*). Para o estado da arte, apresenta-se a média das melhores soluções ($S*$), a média das soluções (S) e o tempo médio de execução em segundos (T). Para o método proposto, além destes dados, apresenta-se o valor médio da solução inicial (I) reportado pelas heurísticas, o desvio padrão (σ) reportado considerando-se as 20 execuções individuais do método proposto, e *gap* médio, calculado como $100 \times \frac{\text{RVND}(S^*) - \text{EA}(S^*)}{\text{EA}(S^*)}$. Em seus respectivos trabalhos, Mecler et al. (2021) (HGS) e Almeida et al.

n	l	C	Estado da arte			RVND					
			S^*	S	T	I	S^*	S	T	σ	gap(%)
50	75	25	**293,40**	**293,56**	1004,01	323,18	302,00	308,49	23,82	3,37	2,93
50	75	30	**226.,20**	**226,40**	1107,12	253,57	236,80	244,37	21,61	3,73	4,69
50	75	35	**182,20**	**182,38**	1223,64	205,53	194,20	199,09	20,07	2,78	6,59
50	75	40	**149,80**	**149,92**	1342,78	169,22	160,60	165,08	18,84	2,57	7,20
60	90	35	**449,60**	**450,40**	12082,73	503,04	464,20	473,41	80,16	5,19	3,25
60	90	40	**359,60**	**360,88**	1414,80	404,00	375,80	386,35	68,57	5,16	4,50
60	90	45	**292,20**	**293,70**	9304,33	331,04	310,60	319,08	61,85	4,54	6,44
60	90	50	**241,00**	**241,94**	1679,71	274,90	260,20	266,79	54,56	3,54	7,97
70	105	40	**616,60**	**617,44**	22535,20	682,04	638,40	648,42	199,62	5,60	3,53
70	105	45	**504,00**	**504,90**	20405,51	563,25	527,40	539,04	179,69	6,28	4,64
70	105	50	**419,60**	**421,08**	17456,87	472,61	445,40	455,94	161,99	5,65	6,15
70	105	55	**353,60**	**354,09**	1967,96	399,90	376,00	386,86	147,86	5,55	6,45

Tabela 4: Comparação dos resultados obtidos com o atual estado da arte.

(2025) não consideram o carregamento inicial de tarefas como trocas. Tal abordagem não é consenso na literatura. Como neste trabalho todas as inserções de ferramentas no *magazine* são consideradas como trocas, os valores originais publicados foram corrigidos para se considerar o carregamento inicial das ferramentas.

Considerando-se todos os subconjuntos de instâncias, o método proposto reportou um *gap* médio de apenas 5,36%, variando entre *gaps* médios individuais de 2,93% e 7,97%. Considerando-se que o objetivo do RVND aqui proposto é gerar soluções iniciais para métodos mais robustos, os baixos *gaps* reportados indicam uma altíssima qualidade das soluções. O tempo computacional médio requerido foi de apenas 87,30 segundos, correspondendo, em média, a apenas 1,15% do tempo demandado pelo método estado da arte, demonstrando a viabilidade da aplicação do método proposto na geração de soluções iniciais. O desvio padrão médio foi de apenas 4,97, demonstrando a eficiência do método em gerar soluções com alta qualidade e baixa variação em execuções individuais.

Em relação às buscas locais, considerando-se todas as execuções do método proposto, a busca local *or-opt* foi a que mais contribuiu para a qualidade final da solução, sendo responsável por 74,50% das melhorias. Em seguida, temos a busca local *maior arrependimento* e *2-opt*, responsáveis respectivamente por 14,11% e 11,39% das melhorias.

6 Conclusões

Sendo um problema industrial prático de ampla aplicação, é essencial que os métodos para solução do SSP obtenham resultados em baixo tempo computacional. Embora as abordagens recentes ao SSP apresentem soluções com alta qualidade, o tempo de execução de tais métodos representa um desafio às suas aplicações práticas. Assim, neste trabalho, propõe-se uma abordagem para geração de soluções válidas com alta qualidade que possam ser utilizadas como soluções iniciais por métodos mais robustos, com a intenção de acelerar o processo exploratório e melhorar a qualidade final dos resultados obtidos.

A abordagem apresentada consiste na geração de uma solução válida utilizando-se heurísticas propostas originalmente para o problema do caixeiro viajante e, posterior melhoria desta solução através de algoritmos de busca local organizados em um método aleatório de descida em vizinhança variável. Os experimentos computacionais realizados comparam os resultados reportados com uma estratégia prévia da literatura para geração de soluções iniciais e com o atual estado da arte para o problema. Frente a estratégia anterior para a geração de soluções iniciais, o método proposto apresenta melhorias significativas. Novos melhores resultados foram apresentados para todas as instâncias consideradas, reportando-se um *gap* médio de -7,40%. Em comparação ao atual estado da arte, embora não seja o objetivo deste trabalho uma comparação direta, o método proposto mostrou-se competitivo, reportando *gaps* médios variando entre 2,93% e 7,97%.

Os trabalhos futuros derivados deste estudo se concentrarão no desenvolvimento de novas abordagens para o SSP que, utilizando o método proposto para geração de soluções iniciais, consigam evoluir o atual estado da arte para o problema.

7 Agradecimentos

Agradecemos à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e ao Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais (IF Sudeste MG) pelo apoio institucional e financeiro concedido para a realização deste trabalho. Agradecemos também ao revisor anônimo e ao Editor pelas valiosas considerações.

Referências

Ahmadi, E., Goldengorin, B., Süer, G. A. and Mosadegh, H. (2018). A hybrid method of 2-tsp and novel learning-based ga for job sequencing and tool switching problem, *Applied Soft Computing* **65**: 214 – 229. http://dx.doi.org/10.1016/j.asoc.2017.12.045.

Almeida, A. L. B., de Castro Lima, J. and Carvalho, M. A. M. (2025). Revisiting the parallel tempering algorithm: High-performance computing and applications in operations research, *Computers & Operations Research* **178**: 107000. https://doi.org/10.1016/j.cor.2025.107000.

Bard, J. F. (1988). A heuristic for minimizing the number of tool switches on a flexible machine, *IIE Transactions*

20(4): 382–391. http://dx.doi.org/10.1080/07408178808966195.

Basirzadeh, H. (2014). Ones assignment method for solving traveling salesman problem, *Journal of mathematics and computer science* **10**(4): 258–265. https://doi.org/10.22436/jmcs.010.04.04.

Beezão, A. C., Cordeau, J.-F., Laporte, G. and Yanasse, H. H. (2017). Scheduling identical parallel machines with tooling constraints, *European Journal of Operational Research* **257**(3): 834–844. https://doi.org/10.1016/j.ejor.2016.08.008.

Bellmore, M. and Nemhauser, G. L. (1968). The traveling salesman problem: a survey, *Operations Research* **16**(3): 538–558. https://doi.org/10.1287/opre.16.3.538.

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. https://doi.org/10.1080/00207543.2018.1505057.

Calmels, D. (2022). An iterated local search procedure for the job sequencing and tool switching problem with non-identical parallel machines, *European Journal of Operational Research* **297**(1): 66–85. https://doi.org/10.1016/j.ejor.2021.05.005.

Catanzaro, D., Gouveia, L. and Labbé, M. (2015). Improved integer linear programming formulations for the job sequencing and tool switching problem, *European Journal of Operational Research* **244**(3): 766 – 777. http://dx.doi.org/10.1016/j.ejor.2015.02.018.

Chaves, A. A., Lorena, L. A. N., Senne, E. L. F. and Resende, M. G. C. (2016). Hybrid method with CS and BRKGA applied to the minimization of tool switches problem, *Computers & Operations Research* **67**: 174–183. http://dx.doi.org/10.1016/j.cor.2015.10.009.

Crama, Y., Kolen, A., Oerlemans, A. and Spieksma, F. (1994). Minimizing the number of tool switches on a flexible machine, *International Journal of Flexible Manufacturing Systems* **6**: 33–54. http://dx.doi.org/10.1007/BF01324874.

Croes, G. A. (1958). A method for solving traveling-salesman problems, *Operations research* **6**(6): 791–812. https://doi.org/10.1287/opre.6.6.791.

Cura, T. (2023). Hybridizing local searching with genetic algorithms for the job sequencing and tool switching problem with non-identical parallel machines, *Expert Systems with Applications* **223**: 119908. https://doi.org/10.1016/j.eswa.2023.119908.

da Silva, T. T., Chaves, A. A. and Yanasse, H. H. (2021). A new multicommodity flow model for the job sequencing and tool switching problem, *International Journal of Production Research* **59**(12): 3617–3632. https://doi.org/10.1080/00207543.2020.1748906.

Djellab, H., Djellab, K. and Gourgand, M. (2000). A new heuristic based on a hypergraph representation for the tool switching problem, *International Journal of Production Economics* **64**(1): 165–176. https://doi.org/10.1016/S0925-5273(99)00055-9.

Gray, A. E., Seidmann, A. and Stecke, K. E. (1993). A synthesis of decision models for tool management in automated manufacturing, *Management Science* **39**(5): 549–567. https://doi.org/10.1287/mnsc.39.5.549.

Hertz, A., Laporte, G., Mittaz, M. and Stecke, K. E. (1998). Heuristics for minimizing tool switches when scheduling part types on a flexible machine, *IIE Transactions* **30**(8): 689–694. http://dx.doi.org/10.1080/07408179808966514.

Ismail, A. H. (2019). Domino algorithm: a novel constructive heuristics for traveling salesman problem, *IOP Conference Series: Materials Science and Engineering*, Vol. 528, IOP Publishing, p. 012043. https://doi.org/10.1088/1757-899X/528/1/012043.

Laporte, G., Salazar-Gonzalez, J. J. and Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem, *IIE Transactions* **36**(1): 37–45. http://dx.doi.org/10.1080/07408170490257871.

Mancio, V. G. and Sellitto, M. A. (2017). Sistemas flexíveis de manufatura: definições e quadro de trabalho para futura pesquisa, *Revista GEINTEC-Gestão, Inovação e Tecnologias* **7**(2): 3760–3773. https://doi.org/10.7198/geintec.v7.i2.709.

Mecler, J., Subramanian, A. and Vidal, T. (2021). A simple and effective hybrid genetic search for the job sequencing and tool switching problem, *Computers & Operations Research* **127**: 105153. http://dx.doi.org/10.1016/j.cor.2020.105153.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search, *Computers & Operations Research* **24**(11): 1097–1100. Disponível em https://www.sciencedirect.com/science/article/pii/S0305054897000312.

Or, O. (1976). *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking*, Ph.d. dissertation, Northwestern University, Evanston, IL, USA.

Paiva, G. S. and Carvalho, M. A. M. (2017). Improved heuristic algorithms for the job sequencing and tool switching problem, *Computers & Operations Research* **88**: 208–219. https://doi.org/10.1016/j.cor.2017.07.013.

Rifai, A. P., Mara, S. T. W. and Norcahyo, R. (2022). A two-stage heuristic for the sequence-dependent job sequencing and tool switching problem, *Computers & Industrial Engineering* **163**: 107813. https://doi.org/10.1016/j.cie.2021.107813.

Senne, E. L. F. and Yanasse, H. H. (2009). Beam search algorithms for minimizing tool switches on a flexible manufacturing system, *in* V. Mladenov, K. Orovoulas, S. Tzafestas, S. Mladenova and E. Martinelli (eds), *Proceedings of The 11th WSEAS International Conference on*

Mathematical and Computational Methods In Science and Engineering (MACMESE '09), WSEAS Press, Rhodes, Greece, pp. 68–72.

Shirazi, R. and Frizelle, G. D. M. (2001). Minimizing the number of tool switches on a flexible machine: an empirical study, *International Journal of Production Research* **39**(15): 3547–3560. http://dx.doi.org/10.1080/00207540110060888.

Silveira, L. M. d. (2019). *Um estudo sobre métodos heurísticos aplicados ao sequenciamento da produção em sistemas de manufatura flexíveis*, Dissertação (mestrado), Universidade Federal de Uberlândia (UFU), Uberlândia, MG. Disponível em http://dx.doi.org/10.14393/ufu.di.2019.1245.

Soares, L. C. R. (2023). *Otimização de processos produtivos em sistemas de manufatura flexível*, Tese de doutorado, Universidade Federal de Ouro Preto (UFOP), Ouro Preto, MG. Disponível em https://www.repositorio.ufop.br/items/17c5dad2-2648-4b78-a60a-16cd43b51be5.

Soares, L. C. R. and Carvalho, M. A. M. (2020). Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints, *European Journal of Operational Research* **285**(3): 955–964. https://doi.org/10.1016/j.ejor.2020.02.047.

Soares, L. C. R. and Carvalho, M. A. M. (2024). Uma nova abordagem rápida e competitiva para o problema de minimização de trocas de ferramentas, *Anais do LVI SBPO*, SOBRAPO, Fortaleza. http://dx.doi.org/10.59254/sbpo-2024-193410.

Soares, L. C., Reinsma, J. A., Nascimento, L. H. and Carvalho, M. A. (2020). Heuristic methods to consecutive block minimization, *Computers & Operations Research* **120**: 104948. http://dx.doi.org/10.1016/j.cor.2020.104948.

Souza, M. J. F. (2008). Inteligência computacional para otimização, *Notas de aula, Departamento de Computaçao, Universidade Federal de Ouro Preto* . Disponível em http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf.

Subramanian, A., Drummond, L. M., Bentes, C., Ochi, L. S. and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery, *Computers & Operations Research* **37**(11): 1899–1911. https://doi.org/10.1016/j.cor.2009.10.011.

Tang, C. (1986). A job scheduling model for a flexible manufacturing machine, *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, Vol. 3, IEEE, pp. 152–155.

Tang, C. S. and 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. http://dx.doi.org/10.1287/opre.36.5.767.

Yanasse, H. H. and Lamosa, M. J. P. (2006). On solving the minimization of tool switches problem using graphs, *XII International Conference on Industrial Engineering and Operations Management*, Fortaleza, Brazil, pp. 1–9.

Yanasse, H. H., Rodrigues, R. d. C. M. and Senne, E. L. F. (2009). Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas, *Gestão & Produção* **16**(3): 370–381. http://dx.doi.org/10.1590/S0104-530X2009000300005.