LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

# Uma abordagem rápida e competitiva para o problema de minimização de trocas de ferramentas

**Leonardo Cabral da Rocha Soares**
Instituto Federal do Sudeste de Minas Gerais
Campus Manhuaçu, Manhuaçu, Minas Gerais, 36909-300, Brasil.
leonardo.soares@ifsudestemg.edu.br

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

**RESUMO**

Aborda-se o problema industrial prático conhecido na literatura como problema de minimização de trocas de ferramentas, cujo objetivo é determinar uma ordem de processamento de tarefas que minimize o número total de operações em *setups*. Para abordar este problema $\mathcal{NP}$-difícil, apresenta-se uma implementação da meta-heurística busca local iterativa. O método proposto é comparado ao método atual estado da arte para o problema, considerando-se todas as instâncias disponíveis na literatura. Os experimentos realizados demonstram que a qualidade das soluções reportadas são equivalentes aos melhores resultados conhecidos. Entretanto, o método proposto supera substancialmente o atual estado da arte quando consideramos o tempo de execução necessário para obtenção das soluções. Dada a natureza prática do problema, métodos mais rápidos são preferíveis para utilização em indústrias visando o planejamento da produção. Adicionalmente, apresentam-se novos melhores resultados para algumas instâncias amplamente utilizadas na literatura.

**PALAVRAS CHAVE. manufatura flexível, busca local iterativa, trocas de ferramentas.**

**Tópicos: OC, POI, MH**

**ABSTRACT**

We address the practical industrial problem known in the literature as job sequencing and tool switching problem, whose objective is to determine a job processing order that minimizes the total number of setup operations. To address this $\mathcal{NP}$-hard problem, we present an implementation of the iterated local search meta-heuristic. Considering all instances available in the literature, the proposed method is compared to the current state-of-the-art method for the problem solution. The experiments carried out demonstrate that the quality of the reported solutions is equivalent to the best-known results. However, the proposed method substantially outperforms the current state of the art when considering the execution time required to obtain the solutions. Given the practical nature of the problem, faster methods are preferable for use in industries aiming at production planning. Additionally, new better results are presented for some instances widely used in the literature.

**KEYWORDS. flexible manufacturing, iterated local search, tool switches.**

**Paper topics: OC, POI, MH**

## 1. Introdução

*Sistemas de manufatura flexíveis* (SMFs) têm sido amplamente utilizados por indústrias para lidar com requisitos de produção complexos e variáveis desde a década de 1980 [Calmels, 2019]. Um SMF é constituído por um conjunto de *máquinas flexíveis* interligadas por um sistema automático de manuseio de materiais. Cada máquina flexível é capaz de executar diversas operações diferentes, como lixar, cortar, furar, entre outras [Zeballos, 2010], desde que as ferramentas necessárias para tais operações estejam previamente carregadas em um compartimento específico, denominado *magazine*. Estes sistemas produtivos estão presentes em diversos segmentos industriais, tais como, indústrias siderúrgicas, automotivas, químicas, farmacêuticas, entre outras [González et al., 2015; Shirazi e Frizelle, 2001]. Entretanto, a literatura demonstra que o gerenciamento de produção em um SMF não é trivial [Stecke, 1983]. De fato, mesmo se considerarmos um ambiente industrial simples, composto por apenas uma máquina flexível, diversas decisões de escalonamento devem ser tomadas para que a produção possa ocorrer de forma eficiente.

Considera-se um SMF constituído por uma única máquina flexível. A produção é divida em estágios. A cada estágio, um conjunto de operações, aqui nomeado como *processamento* de uma *tarefa*, precisa ser executado sobre uma determinada matéria prima ou produto semi-acabado. Cada conjunto de operações requer um conjunto de ferramentas específico. Embora limitada, a capacidade do *magazine* da máquina flexível é suficiente para conter todas as ferramentas necessárias para o processamento de uma tarefa. Entretanto, dado o processamento sequencial de uma série de tarefas, pode ser necessário realizar trocas de ferramentas no *magazine* antes que a próxima tarefa possa ser iniciada.

De acordo com Van Hop e Nagarur [2004], as trocas de ferramentas consomem mais tempo do que qualquer outra operação de configuração em um SMF. Ademais, Beezão et al. [2017] menciona que entre 25% e 30% dos custos fixos e variáveis de um SMF ocorrem devido as trocas de ferramentas. Assim, para o SMF exemplificado, o planejamento de produção deve considerar a minimização das trocas de ferramentas visando diminuir o custo de produção e aumentar a produtividade do sistema. Na literatura, este planejamento de produção é conhecido como o problema de minimização de trocas de ferramentas (*job sequencing and tool switching problem*, SSP).

Introduzido por Tang [1986], o SSP consiste de dois problemas combinatórios: a definição da ordem de processamento para um conjunto de tarefas em uma máquina flexível e a determinação do plano de trocas de ferramentas. O primeiro problema é $\mathcal{NP}$-difícil [Crama et al., 1994], ou seja, não se conhece algoritmo determinístico polinomial para sua solução, sendo responsável pela complexidade do SSP. Para o segundo problema, dada uma ordem de processamento para as tarefas, o plano ótimo de trocas de ferramentas pode ser obtido utilizando-se a política de manter no *magazine* as ferramentas que serão utilizadas mais cedo (*keep tool needed soonest*, KTNS), apresentada por Tang e Denardo [1988].

Neste estudo, explora-se a conexão entre o SSP e um problema relacionado, o problema de minimização de blocos de uns consecutivos (*consecutive block minimization problem*, CBM). Apresenta-se uma implementação da meta-heurística busca local iterativa (*iterated local search*, ILS). Extensos experimentos computacionais consideraram todas as instâncias disponíveis na literatura e novos melhores resultados inéditos são reportados para algumas instâncias, além de uma redução substancial no tempo de execução quando comparado ao método estado da arte.

O restante do artigo é organizado da seguinte forma. O problema abordado é formalmente descrito na Seção 2. A Seção 3 apresenta a revisão da literatura sobre o SSP. O método proposto é descrito em detalhes na Seção 4. A Seção 5 descreve os experimentos realizados e os resultados obtidos. Por último, a Seção 6 apresenta as conclusões e trabalhos futuros.

## 2. Descrição do problema

Formalmente, considerando-se um SMF constituído por uma única máquina flexível cujo *magazine* pode conter até $C$ ferramentas simultaneamente, um conjunto $J = \{1, \ldots, n\}$ de tarefas a serem processadas, um conjunto $T = \{1, \ldots, m\}$ de ferramentas e um subconjunto $T_j$ ($T_j \in T$) de ferramentas necessárias para processar a tarefa $j$ ($j \in J$), em que $|T_j| \leq C$, o SSP consiste em determinar a ordem de processamento das tarefas de forma que o número total de trocas de ferramentas seja minimizado.

Dada sua ampla aplicação prática em diversos ambiente industriais, o SSP possui diversas variações e problemas relacionados [Calmels, 2019]. Em sua versão padrão, também conhecida na literatura como SSP *uniforme*, o SSP possui as seguintes características [Bard, 1988; Crama et al., 1994]: $(i)$ todas as ferramentas são consideradas idênticas e portanto podem ocupar qualquer posição livre no *magazine*; $(ii)$ somente uma ferramentas pode ser trocada por vez e o tempo necessário para a troca de uma ferramenta é considerado constante e idêntico para todas as ferramentas; $(iii)$ o conjunto de tarefas e o subconjunto de ferramentas necessárias para o processamento de cada tarefa é conhecido à *priori*; $(iv)$ o número de ferramentas requeridas para processar uma tarefa é menor ou igual à capacidade do *magazine*; e $(v)$ não são consideradas trocas de ferramentas originadas por desgastes ou quebras.

Uma instância do SSP pode ser representada por uma matriz binária $Q$. Cada linha da matriz corresponde a uma tarefa e cada coluna corresponde a uma ferramenta. Se uma tarefa $j$ requer a ferramenta $t$, então $q_{jt} = 1$. Caso contrário, $q_{jt} = 0$. A Tabela 1 apresenta um exemplo de instância para o SSP com $n = 6$, $T = \{1, \ldots, 10\}$ e $C = 6$.

A solução para o problema é obtida a partir da permutação $\pi$ das colunas do *plano de trocas de ferramentas* que também pode ser representado por uma matriz binária $R^{\pi} = \{r_{tj}\}$ com $t \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$. A cada etapa do plano de trocas de ferramentas, representada por suas colunas, uma tarefa é processada. As linhas do plano de trocas de ferramentas indicam as ferramentas carregadas no *magazine* antes do início do processamento de cada etapa. Um elemento $r_{tj} = 1$ indica que a ferramenta $t$ está carregada no *magazine* da máquina flexível durante o processamento da tarefa $j$, e $r_{tj} = 0$ indica o contrário.

Considere a instância apresentada na Tabela 1 e a permutação $\pi$ = {1,3,5,2,4,6}. A Tabela 2 apresenta o plano de trocas de ferramentas correspondente. Ferramentas inseridas ou removidas imediatamente antes do processamento de uma tarefa são apresentadas sublinhadas. Por exemplo, antes do processamento da tarefa 2, no quarto estágio, a ferramenta 9 foi removida e a ferramenta 4 inserida no *magazine*, assim, ambas os elementos aparecem sublinhados.

Tabela 1: Instância do SSP.

| Tarefas | Ferramentas | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Tabela 2: Plano de trocas de ferramentas.

| Ferramentas | Estágios/Tarefas | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 3 | 5 | 2 | 4 | 6 |
| 1 | 0 | <u>1</u> | <u>0</u> | 0 | <u>1</u> | 1 |
| 2 | 0 | <u>1</u> | 1 | 1 | 1 | 1 |
| 3 | <u>1</u> | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 0 | <u>1</u> | <u>0</u> | 0 |
| 5 | <u>1</u> | 1 | <u>0</u> | 0 | <u>1</u> | 1 |
| 6 | <u>1</u> | 1 | 1 | 1 | <u>0</u> | 0 |
| 7 | <u>1</u> | <u>0</u> | <u>1</u> | 1 | 1 | 1 |
| 8 | 0 | 0 | <u>1</u> | 1 | 1 | 1 |
| 9 | <u>1</u> | 1 | 1 | <u>0</u> | 0 | 0 |
| 10 | <u>1</u> | <u>0</u> | 0 | 0 | 0 | 0 |

De acordo com a permutação $\pi = \{1, 3, 5, 2, 4, 6\}$, a primeira tarefa a ser processada

($j = 1$) requer somente cinco ferramentas $\{5, 6, 7, 9, 10\}$. Como a capacidade do *magazine* permite uma ferramenta adicional ($C = 6$), uma ferramenta requerida pelo próximo estágio de produção deve ser carregada. Neste exemplo, a ferramenta 3 foi selecionada. Para evitar trocas de ferramentas desnecessárias, uma ferramenta não utilizada no estágio atual mas requerida nos estágios futuros, deve ser mantida no *magazine* sempre que possível.

O plano de trocas de ferramentas é avaliado utilizando-se a Equação (1) proposta por Crama et al. [1994]. Esta equação calcula o número de inversões de zero para um na representação matricial, que representam as inserções de ferramentas no *magazine*. Para que o carregamento inicial de ferramentas seja considerado, uma coluna artificial com todos os elementos iguais a zero deve ser adicionada ao plano de trocas de ferramentas, ou seja, $R^{\pi}_{t0} = 0$. Para o exemplo apresentado, são realizadas 13 trocas de ferramentas.

$$Z_{SSP}(R^{\pi}) = \sum_{t \in T} \sum_{j \in J} r^{\pi}_{tj}(1 - r^{\pi}_{tj-1}) \tag{1}$$

O objetivo do SSP é obter uma permutação $\pi \in \Pi$ para o escalonamento de tarefas que minimize o total de trocas de ferramentas. A função objetivo do SSP é representada pela Equação (2), em que $\Pi$ é o conjunto de todas as possíveis permutações de tarefas.

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

Conforme mencionado, para evitar trocas de ferramentas desnecessárias, sempre que possível, uma ferramenta necessária nos estágios de produção futuros deve ser mantida no *magazine*. Quando uma ferramenta é removida e posteriormente reinserida no *magazine*, isto é evidenciado por uma *descontinuidade* na representação matricial do plano de trocas de ferramentas. Uma descontinuidade ocorre sempre que um elemento nulo ocorre entre dois conjuntos de elementos não-nulos em uma matriz binária.

O estudo de descontinuidades em matrizes binárias define o problema $\mathcal{NP}$-difícil conhecido na literatura como CBM. Embora o CBM e o SSP não sejam equivalentes, as estratégias utilizadas para solução do CBM podem resultar em boas soluções para o SSP. Assim, neste estudo, recentes avanços no estudo do CBM [Soares et al., 2020] são considerados como parte da estratégia para abordagem ao SSP.

## 3. Revisão da literatura

Desde sua publicação seminal na década de 1980, diversos autores têm abordado o SSP e suas variações [Soares e Carvalho, 2020; Rifai et al., 2022; Calmels, 2022; Cura, 2023]. Uma revisão ampla, contendo as principais variações pode ser encontrada em Calmels [2019]. A seguir, apresenta-se as principais contribuições para a versão original do SSP em ordem cronológica.

O SSP foi formalmente definido por Tang e Denardo [1988]. Neste estudo, o problema é modelado e resolvido como uma instância do problema do caixeiro viajante (*traveling salesman problem, TSP*). Além disso, os autores apresentam a política KTNS que permite definir em tempo determinístico polinomial a solução ótima o plano de trocas de ferramentas para uma dada sequência fixa de tarefas.

Crama et al. [1994] provou que o SSP pertence a classe de problemas $\mathcal{NP}$-difícil para qualquer instância com $C \geq 2$. Novamente, modela-se o problema como uma instância do TSP. Tal modelagem se tornou recorrente na literatura [Hertz et al., 1998; Djellab et al., 2000; Shirazi e Frizelle, 2001; Laporte et al., 2004; Ahmadi et al., 2018]. Entretanto, em um trabalho teórico, Yanasse e Lamosa [2006a,b] demonstram que esta abordagem é ineficiente, pois o SSP e o TSP não são equivalentes.

Um novo modelo de programação linear inteira para o SSP foi proposto por Laporte et al. [2004]. O modelo apresentado apresenta melhores valores de relaxação linear quando comparados ao modelo de Tang e Denardo [1988]. Além disso, são apresentados e comparados dois algoritmos para o problema, um método de *branch-and-bound* e um de *branch-and-cut*. Os experimentos realizados utilizaram instâncias geradas no próprio artigo. O método *branch-and-cut* mostrou-se apto a resolver de forma ótima instâncias com até nove tarefas e o método *branch-and-bound* instâncias com até 25 tarefas.

Um algoritmo enumerativo para o SSP é apresentado por Yanasse et al. [2009]. Os resultados obtidos foram comparados e superam os reportados por Laporte et al. [2004]. Em um trabalho subsequente, Senne e Yanasse [2009] apresentam três implementações de *beam search* baseadas no mesmo esquema enumerativo. Entretanto, os resultados obtidos pelos algoritmos propostos não foram comparados com outros métodos da literatura.

Chaves et al. [2016] apresentam um algoritmo híbrido combinando um algoritmo de busca de agrupamento (*clustering search*, CS) com um algoritmo genético de chaves aleatórias viciadas (*biased random-key genetic algorithm*, BRKGA). Os experimentos computacionais utilizaram 1510 instâncias [Yanasse et al., 2009; Crama et al., 1994]. Novos melhores valores foram publicados para todos os problemas.

Paiva e Carvalho [2017] apresentam uma nova representação em grafos, uma heurística construtiva e uma nova implementação da meta-heurística ILS para o SSP. Entre as buscas locais utilizadas pela ILS, os autores introduzem uma busca local por agrupamento de blocos de uns consecutivos em uma matriz binária, explorando a conexão entre o SSP e o CBM. Os experimentos computacionais utilizaram 1670 instâncias [Yanasse et al., 2009; Crama et al., 1994; Catanzaro et al., 2015]. Os resultados obtidos foram comparados com os melhores valores conhecidos para todos os problemas. De acordo com os experimentos realizados, o método proposto reportou soluções equivalentes ou melhores para todas as instâncias consideradas.

Ahmadi et al. [2018] modelam o SSP como uma instância do TSP de segunda ordem (2-TSP) e o resolvem com uma implementação do algoritmo genético *q-learning*. Entretanto, os autores não detalham a métrica utilizada para determinação das distâncias utilizadas na modelagem. Os experimentos consideraram apenas apenas um subconjunto das instâncias disponíveis [Catanzaro et al., 2015; Crama et al., 1994]. Os resultados foram comparados com Paiva e Carvalho [2017] e apresentaram melhorias marginais.

Uma nova formulação linear inteira para o SSP, utilizando um modelo de multi-fluxo é apresentada por Silva et al. [2020]. Os experimentos utilizaram um subconjunto das instâncias disponíveis na literatura [Yanasse et al., 2009; Catanzaro et al., 2015]. Os resultados reportados superam os obtidos por modelos anteriores, entretanto, considerando-se o tempo limite estabelecido em uma hora, a modelagem não é capaz de resolver os maiores problemas dentre as instâncias utilizadas.

Mecler et al. [2021] apresentam um algoritmo de pesquisa genética híbrida (ou *hybrid genetic search*, HGS) para o SSP. Além do *benchmark* de instâncias disponível na literatura [Yanasse et al., 2009; Crama et al., 1994; Catanzaro et al., 2015], os experimentos computacionais utilizaram um novo conjunto contendo 60 instâncias com maiores dimensões, propostas no próprio artigo. Os resultados obtidos foram comparados com os reportados pelos métodos apresentados por [Chaves et al., 2016; Paiva e Carvalho, 2017; Ahmadi et al., 2018]. O método proposto reportou melhores resultados médios para todos os conjuntos de instâncias, tornando-se o atual estado da arte para o SSP.

## 4. Métodos

Em um trabalho recente, Soares et al. [2020] apresentam um método exato para solução do CBM. O método consiste em $(i)$ transformar o CBM em uma instância do problema de minimização de blocos circulares (*circular block minimization problem*, CIR); $(ii)$ transformar o CIR em uma instância do TSP em que a matriz de distância é calculada utilizando-se a distância de Hamming [Hamming, 1986]; e, $(iii)$ solução do problema com o resolvedor exato Concorde [Applegate et al., 2006].

Embora o SSP e o CBM sejam equivalentes apenas para o caso especial em que cada tarefa do SSP requer exatamente $C$ ferramentas para ser processada, a conexão entre estes dois problemas tem sido explorada com sucesso na literatura [Paiva e Carvalho, 2017; Soares e Carvalho, 2020]. Entretanto, não é esperado que esta modelagem, isoladamente, produza soluções de alta qualidade. Assim, apresenta-se uma implementação da meta-heurística ILS que explora o espaço de busca a partir de uma solução inicial gerada por esta modelagem. O método proposto é descrito a seguir.

### 4.1. Busca local iterativa

Conceitualmente simples, a meta-heurística ILS, a partir de uma dada solução inicial, explora o espaço de busca aplicando alternadamente mecanismos de *intensificação* e *diversificação* à procura de uma solução final de alta qualidade. Apesar de sua simplicidade conceitual, esta meta-heurística tem sido aplicada com sucesso a diversos problemas computacionalmente difíceis [Lourenço et al., 2019], incluindo aplicações diretas ao CBM [Soares et al., 2020] e ao próprio SSP [Paiva e Carvalho, 2017]. A implementação da meta-heurística ILS proposta para abordagem ao SSP é descrita a seguir.

Uma solução inicial é gerada utilizando-se a modelagem proposta por Soares et al. [2020] para o CBM. Para solução da modelagem, optou-se por utilizar o método *Lin-Kernighan heuristic*, LKH [Helsgaun, 2018], reconhecido na literatura como um dos métodos mais bem sucedidos na geração de soluções ótimas e subótimas para o TSP [Helsgaun, 2000]. Embora o resolvedor Concorde tenha sido utilizado na abordagem original, sua aplicação sucessiva nas instâncias disponíveis para o SSP mostrou-se instável. Diante disso, optou-se por utilizar o método LKH. Uma análise preliminar mostrou que tal diferença não impactou a qualidade das soluções iniciais obtidas.

Na etapa de intensificação, são utilizados os métodos de *busca local inversão de elementos adjacentes*, *2-opt* e *busca local de maior arrependimento*, nesta ordem. O método busca local inversão de elementos adjacentes é uma variação do tradicional método 2-swap. Um movimento desta busca local consiste na troca de dois elementos diretamente adjacentes na solução. O método 2-opt consiste na inversão de todos os elementos em um intervalo selecionado aleatoriamente. O método busca local de maior arrependimento, classifica cada tarefa de acordo com o número de trocas de ferramentas gerado pelo seu escalonamento em sua posição atual. Posteriormente, esta busca local reposiciona cada tarefa na posição em que gerar o menor número de trocas de ferramentas. As tarefas são reposicionadas de acordo com a ordem estabelecida pela classificação, garantido que as tarefas que mais originavam trocas de ferramentas sejam reposicionadas primeiro.

Para a etapa de diversificação foi utilizado o método *ponte dupla*. Proposto originalmente para o TSP, quando aplicado ao SSP este método divide o escalonamento de tarefas em quatro pontos. Em seguida, os componentes contendo os segmentos da solução são reordenados gerando uma solução vizinha que mantém a estrutura interna de cada componente.

Foram utilizados três critérios de aceitação, a saber, soluções com melhor valor de avaliação; soluções com igual valor de avaliação mas com maior número de ferramentas removidas e reinseridas ao longo do plano de trocas de ferramentas; e, soluções com valor de avaliação e número de reinserções de ferramentas iguais à solução atual mas com maior número de reinserções de uma mesma ferramenta. Os critérios adicionais de aceitação foram incorporados com o objetivo de guiar a trajetória do ILS por soluções com características que indicam um maior potencial de melhoria.

Como critério de parada do método foi utilizado o número máximo de iterações. A quantidade máxima de iterações e a ordem de aplicação das buscas locais foram definidos nos experimentos preliminares descritos na Seção 5.

As etapas do método proposto são descritas no Algoritmo 1. A solução inicial é construída utilizando-se a modelagem proposta para o CBM (linha 1). As buscas locais inversão de elementos adjacentes, 2-opt e maior arrependimento são aplicadas à solução inicial (linhas 2 a 4). O laço principal (linhas 5 a 13) assegura a execução da método até que o critério de parada seja alcançado. A solução atual $s$ é perturbada utilizando-se um movimento do método ponte dupla, resultando na solução vizinha $s'$ (linha 6). A etapa de intensificação da meta-heurística é conduzida pela aplicação sequencial dos métodos de buscas locais inversão de elementos adjacentes (linha 7), 2-opt (linha 8) e maior arrependimento (linha 9). Caso $s'$ atenda aos critérios de aceitação para uma nova solução, a solução atual é atualizada com $s'$ (linhas 10 a 12). Após alcançado o critério de parada, a solução atual $s$ é retornada (linha 14).

**Algoritmo 1:** Busca local iterativa

**Entrada:** Instância do problema;
**Saída:** Solução $s$;
1 $s \leftarrow$ soluçãoInicial();
2 **aplique** a busca local inversão de elementos adjacentes a $s$;
3 **aplique** a busca local 2-opt a $s$;
4 **aplique** a busca local maior arrependimento a $s$;
5 **enquanto** *critério de parada não for encontrado* **faça**
6 $s' \leftarrow$ **execute** um movimento do método ponte dupla em $s$;
7 **aplique** a busca local inversão de elementos adjacentes a $s'$;
8 **aplique** a busca local 2-opt a $s'$;
9 **aplique** a busca local maior arrependimento a $s'$;
10 **se** $s'$ *atende aos critérios de aceitação* **então**
11 $s \leftarrow s'$;
12 **fim**
13 **fim**
14 **retorne** $s$;

## 5. Experimentos

Uma série de experimentos computacionais foi realizada para mensurar a qualidade das soluções reportadas pelo método proposto e avaliar seu comportamento. O ambiente computacional utilizado consiste de um computador com processador *Intel Xeon E5-2660* de 2.2 GHz com 384 GB de memória RAM sob o sistema operacional CentOS 6.8. Todos os métodos foram implementados em C++, compilados com GCC 11.2.0 e as opções de otimização -O3 e -march = native.

Os parâmetros do ILS foram definidos utilizando-se o método *offline* de configuração automática de algoritmos de otimização *irace* [López-Ibáñez et al., 2016], a saber, a ordem de aplicação das buscas locais e o número máximo de iterações, definido como 1500.

Conforme mencionado, o SSP possui quatro conjuntos de instâncias disponíveis na literatura, sendo os três primeiros considerados instâncias clássicas e o último um conjunto recente contendo os maiores problemas. Em ordem cronológica, o primeiro conjunto de instâncias foi proposto por Crama et al. [1994] e contém 160 problemas divididos em quatro grupos ($C_1$, $C_2$, $C_3$, $C_4$). O segundo, proposto por Yanasse et al. [2009], contém 1350 problemas divididos em cinco grupos

($A$, $B$, $C$, $D$, $E$). O terceiro, introduzido por Catanzaro et al. [2015], possui 160 instâncias divididas em quatro grupos ($datA$, $datB$, $datC$, $datD$). O quarto, proposto por Mecler et al. [2021], divide-se em três grupos ($F_1$, $F_2$, $F_3$) e, conforme mencionado, possui os problemas com as maiores dimensões. Todas as instâncias foram consideradas nos experimentos realizados.

A Tabela 3 apresenta, para os conjuntos de instâncias clássicos, os resultados que constituem o atual estado da arte bem como os valores médios obtidos a partir de dez execuções individuais do método proposto. Seguindo o padrão adotado por Mecler et al. [2021], os valores reportados apresentam a média das médias de cada subconjunto. Para o método detentor do estado da arte (HGS), os valores médios das soluções reportadas no trabalho original foram aumentados em $C$ unidades para considerar as trocas de ferramentas iniciais, conforme considerado pelo método aqui proposto. Para ambos os métodos, apresenta-se a média das melhores soluções reportadas ($S^*$) e a média das soluções ($S$). Em adição, para o ILS apresenta-se a média das soluções iniciais ($I$), o desvio padrão de $S$ ($\sigma$), e a distância percentual ($gap$) entre a melhor solução reportada pelos métodos ILS e HGS, calculada como $100 \times \frac{\text{ILS}(S^*) - \text{HGS}(S^*)}{\text{HGS}(S^*)}$. Os melhores valores de solução são destacados em negrito.

Tabela 3: Resultados para as instâncias clássicas.

| | HGS | | ILS | | | | |
|---|---|---|---|---|---|---|---|
| Grupo | $S^*$ | $S$ | $I$ | $S^*$ | $S$ | $\sigma$ | $gap(\%)$ |
| $C_1$ | **11,18** | **11,18** | 12,18 | **11,18** | **11,18** | 0,00 | 0,00 |
| $C_2$ | **22,00** | **22,00** | 24,58 | **22,00** | 22,00 | 0,01 | 0,00 |
| $C_3$ | **79,33** | 79,72 | 103,76 | 79,35 | 79,44 | 0,08 | 0,03 |
| $C_4$ | 157,03 | 157,24 | 183,23 | **157,00** | 157,22 | 0,22 | -0,02 |
| $A$ | **23,18** | **23,18** | 24,18 | **23,18** | **23,18** | 0,00 | 0,00 |
| $B$ | **23,87** | **23,87** | 25,14 | **23,87** | **23,87** | 0,00 | 0,00 |
| $C$ | 28,02 | 28,02 | 30,70 | **27,80** | **27,80** | 0,00 | -0,79 |
| $D$ | **25,05** | **25,05** | 28,56 | **25,05** | 25,08 | 0,04 | 0,00 |
| $E$ | **16,89** | **16,89** | 18,38 | **16,89** | **16,89** | 0,00 | 0,00 |
| $datA$ | **10,85** | **10,85** | 11,55 | **10,85** | **10,85** | 0,00 | 0,00 |
| $datB$ | **21,78** | **21,78** | 24,00 | **21,78** | **21,78** | 0,00 | 0,00 |
| $datC$ | **74,68** | 74,73 | 89,93 | 74,70 | 74,99 | 0,29 | 0,03 |
| $datD$ | **157,10** | 157,36 | 186,58 | 157,15 | 157,88 | 0,63 | 0,03 |

Considerando-se todo o *benchmark* de instâncias clássicas, o ILS reportou um *gap* médio de -0,06%, variando entre -0,79% e 0,03%. Embora a diferença entre as soluções seja apenas marginal, novos melhores resultados inéditos são apresentados para os grupos $C_4$ e $C$. Em média, o ILS precisou de 147,29 iterações para melhorar a solução inicial em 14,68%. Entre as buscas locais, 2-opt é a que mais contribuiu para a qualidade final da solução, respondendo em média por 67,89% das melhorias. Em seguida, tem-se a busca local inversão de elementos adjacentes e a busca local maior arrependimento, responsáveis, por 20,02% e 12,09% das melhorias, respectivamente. O desvio padrão médio reportado para este conjunto é de apenas 0,10, indicando a capacidade do método de gerar soluções independentes com baixa variação. Para este conjunto de instâncias, o tempo de execução é pequeno, sendo considerado desprezível. Ambos os métodos reportam tempo de execução não superior a cinco minutos para resolver todas as instâncias.

Seguindo o mesmo padrão apresentado para a tabela anterior, a Tabela 4 apresenta os resultados para o novo conjunto de instâncias. Para uma comparação justa do tempo de execução dos algoritmos, o método HGS original, disponibilizado pelos autores, foi executado para este conjunto

de instâncias no mesmo ambiente computacional utilizado por este estudo. Os resultados reportados pelo HGS são compatíveis com os reportados pelo trabalho original. Em acréscimo aos dados apresentados pela tabela anterior, apresenta-se o número de tarefas ($n$), ferramentas ($m$), capacidade do *magazine* ($C$), e tempo médio de execução ($T$) em segundos.

Tabela 4: Resultados para o conjunto de instâncias proposto por Mecler et al. [2021].

| | | | HGS | | | ILS | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $m$ | $C$ | $S^*$ | $S$ | $T$ | $I$ | $S^*$ | $S$ | $T$ | $\sigma$ | $gap(\%)$ |
| 50 | 75 | 25 | **293,60** | 293,82 | 3303,42 | 331,80 | **293,60** | 293,64 | 2240,41 | 0,08 | 0,00 |
| 50 | 75 | 30 | **226,40** | 227,10 | 2694,11 | 266,80 | 226,80 | 226,96 | 1983,58 | 0,26 | 0,18 |
| 50 | 75 | 35 | **182,40** | 183,08 | 2731,85 | 222,00 | 182,60 | 183,22 | 1755,40 | 0,43 | 0,11 |
| 50 | 75 | 40 | **149,80** | 150,42 | 2300,50 | 188,00 | 150,20 | 150,86 | 1612,24 | 0,55 | 0,27 |
| 60 | 90 | 35 | **449,60** | 450,60 | 12082,73 | 515,80 | 450,60 | 451,60 | 7137,42 | 0,71 | 0,22 |
| 60 | 90 | 40 | **360,00** | 361,14 | 10979,21 | 425,60 | 360,20 | 361,50 | 6138,69 | 0,67 | 0,06 |
| 60 | 90 | 45 | **292,20** | 293,70 | 9304,33 | 358,80 | 292,80 | 294,18 | 5337,49 | 0,74 | 0,21 |
| 60 | 90 | 50 | **241,20** | 242,88 | 8107,61 | 303,80 | 242,40 | 243,76 | 4609,05 | 0,86 | 0,50 |
| 70 | 105 | 40 | **616,60** | 617,58 | 23693,08 | 704,20 | 617,20 | 618,58 | 13029,04 | 0,77 | 0,10 |
| 70 | 105 | 45 | **504,00** | 505,10 | 21302,32 | 594,80 | 504,60 | 505,52 | 11773,04 | 0,67 | 0,12 |
| 70 | 105 | 50 | **419,60** | 421,06 | 17612,52 | 510,20 | 419,80 | 421,76 | 10821,81 | 0,88 | 0,05 |
| 70 | 105 | 55 | **353,60** | 355,02 | 16899,57 | 441,40 | 354,00 | 355,88 | 9799,70 | 1,06 | 0,11 |

Comparado com o HGS, o ILS reportou um *gap* médio de 0,16%, variando entre -0,41% e 1,26%. Considerando-se os valores individuais, o ILS apresenta *gap* menor ou igual a zero para 58,33% das instâncias deste conjunto. A ausência de diferenças substanciais entre os resultados reportados para este novo conjunto de instâncias assemelha-se ao estado das instâncias clássicas, dificultando a análise das diferenças entre os métodos. Além disso, a falta de soluções ótimas conhecidas ou limites inferiores de boa qualidade torna impossível determinar se há espaço disponível para uma melhoria significativa nos resultados gerados. Para tentar resolver esta limitação, solicitamos aos autores de Silva et al. [2020] que executassem o modelo de multi-fluxo para as instâncias deste conjunto. Infelizmente, dentro de um limite de tempo de 3600 segundos, não foram reportados resultados ótimos e os limitantes inferiores reportados não foram justos, com valores próximos de $m - C$ em geral.

Em um resultado notável, o tempo de execução do ILS é 41,81% menor do que o requerido por HGS, em média. Dada a origem industrial do SSP, um tempo de execução excessivamente elevado pode apresentar-se como um obstáculo à aplicação de métodos computacionais no planejamento prático da produção. Considerando todas as execuções individuais para todas as instâncias deste conjunto, o tempo máximo de execução reportado pelo HGS foi de 47.486,00 segundos enquanto o tempo máximo de execução do ILS foi de 14.601,50 segundos, portanto, 69,25% menor.

A convergência média do ILS ocorreu com 913,83 iterações tendo melhorado a solução inicial em 19,01%. O comportamento das buscas locais é similar ao reportado para o conjunto de instâncias clássicas. A busca local 2-opt é a que mais contribuiu para a qualidade final da solução respondendo por 79,20% das melhorias. Em seguida, tem-se as buscas locais inversão de elementos adjacentes e a busca local maior arrependimento, responsáveis por 13,65% e 7,14% das melhorias, respectivamente. Novamente, o baixo desvio padrão reportado, em média apenas 0,64, confirma a consistência do ILS em gerar soluções de alta qualidade com baixa variação.

Uma segunda versão do ILS usando soluções iniciais aleatórias ($\text{ILS}_r$) foi implementada para avaliar a convergência do método proposto. Em média, o $\text{ILS}_r$ precisou de 907 iterações para melhorar a solução inicial aleatória em 47,63%. Apesar da piora significativa na qualidade das

soluções iniciais, o ILS$_r$ convergiu para soluções com qualidade equiparável à sua versão anterior, reportando um *gap* médio de 0,28% sem aumentar significativamente o tempo de execução, demonstrando a robustez do método proposto.

## 6. Conclusão

Neste estudo, apresentou-se uma nova abordagem para o SSP, um problema $\mathcal{NP}$-difícil com grande aplicação prática em diversos segmentos industriais. Avanços recentes em um problema relacionado, CBM, foram considerados na implementação da meta-heurística busca local iterativa. O método proposto foi comparado ao atual estado da arte para o SSP considerando-se todas as instâncias disponíveis na literatura. Embora as diferenças entre os métodos sejam apenas marginais, apresentou-se novos melhores resultados para dois grupos de instâncias clássicas amplamente utilizadas. Ademais, o método proposto mostrou-se, em média, 41,87% mais rápido que o atual método estado da arte para o problema. Dada a característica prática do SSP, a obtenção de soluções com qualidade equiparável em menor tempo computacional apresenta-se como uma vantagem estratégia para utilização do método em cenários reais. Os trabalhos futuros diretamente derivados deste estudo se concentrarão em uma análise profunda das instâncias disponíveis, buscando evidências que apontem para possibilidade ou não de melhorias nos resultados já publicados.

## 7. Agradecimentos

Esta pesquisa foi apoiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, 408341/2018-1, pela Universidade Federal de Ouro Preto e Instituto Federal do Sudeste de Minas Gerais. Os autores expressam seu especial agradecimento à Tiago Tibúrcio da Silva e Antônio Augusto Chaves, que gentilmente executaram o modelo de multi-fluxo para as maiores instâncias utilizadas nos experimentos computacionais.

## Referências

Ahmadi, E., Goldengorin, B., Süer, G. A., e 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. ISSN 1568-4946.

Applegate, D., Bixby, R., Chvatal, V., e Cook, W. (2006). Concorde TSP solver. URL `http://www.math.uwaterloo.ca/tsp/concorde/`.

Bard, J. F. (1988). A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, 20(4):382–391. ISSN 0740-817X.

Beezão, A. C., Cordeau, J.-F., Laporte, G., e Yanasse, H. H. (2017). Scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research*, 257(3):834–844.

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.

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.

Catanzaro, D., Gouveia, L., e 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. ISSN 0377-2217.

Chaves, A. A., Lorena, L. A. N., Senne, E. L. F., e 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. ISSN 0305-0548.

Crama, Y., Kolen, A. W. J., Oerlemans, A. G., e Spieksma, F. C. R. (1994). Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6(1):33–54. ISSN 0920-6299, 1572-9370.

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.

Djellab, H., Djellab, K., e 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.

González, M. A., Oddi, A., Rasconi, R., e Varela, R. (2015). Scatter search with path relinking for the job shop with time lags and setup times. *Computers & Operations Research*, 60:37–54.

Hamming, R. W. (1986). *Coding and information theory*. Prentice-Hall, Inc.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106 – 130. ISSN 0377-2217.

Helsgaun, K. (2018). Lin-kernighan heuristic. URL `http://akira.ruc.dk/~keld/research/LKH/`.

Hertz, A., Laporte, G., Mittaz, M., e Stecke, K. E. (1998). Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE Transactions*, 30(8):689–694. ISSN 0740-817X.

Laporte, G., Salazar-Gonzalez, J. J., e Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36(1):37–45. ISSN 0740-817X.

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

Lourenço, H. R., Martin, O. C., e Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics*, p. 129–168. Springer.

Mecler, J., Subramanian, A., e Vidal, T. (2021). A simple and effective hybrid genetic search for the job sequencing and tool switching problem. *Computers & Operations Research*, 127:105153.

Paiva, G. S. e Carvalho, M. A. M. (2017). Improved heuristic algorithms for the job sequencing and tool switching problem. *Computers & Operations Research*, 88:208–219.

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

Senne, E. L. F. U. e Yanasse, H. H. (2009). Beam search algorithms for minimizing tool switches on a flexible manufacturing system. *Proceedings of The 11th Wseas International Conference on Mathematical and Computational Methods In Science and Engineering (MACMESE '09)*, p. 68–72.

Shirazi, R. e 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. ISSN 0020-7543.

Silva, T. T. d., Chaves, A. A., e Yanasse, H. H. (2020). A new multicommodity flow model for the job sequencing and tool switching problem. *International Journal of Production Research*, p. 1–16.

Soares, L. C. R. e 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.

Soares, L. C., Reinsma, J. A., Nascimento, L. H., e Carvalho, M. A. (2020). Heuristic methods to consecutive block minimization. *Computers & Operations Research*, 120:104948.

Stecke, K. E. (1983). Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management Science*, 29(3):273–288.

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

Tang, C. S. e Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part I: Minimization of the number of tool switches. *Operations Research*, 36(5):767–777. ISSN 0030-364X.

Van Hop, N. e Nagarur, N. N. (2004). The scheduling problem of pcbs for multiple non-identical parallel machines. *European Journal of Operational Research*, 158(3):577–594.

Yanasse, H. H. e Lamosa, M. J. P. (2006a). An application of the generalized travelling salesman problem: the minimization of tool switches problem. In *International Annual Scientific Conference of the German Operations Research Society*, p. 90–100, Bremen, Germany.

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

Yanasse, H. H., Rodrigues, R. d. C. M., e 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. ISSN 0104-530X.

Zeballos, L. (2010). A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 26 (6):725–743.