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

PEDRO LUCAS DAMASCENO
Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

BUSCA LOCAL ITERADA APLICADA AO SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS FLEXÍVEIS PARALELAS NÃO IDÊNTICAS

Ouro Preto, MG
2024

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

PEDRO LUCAS DAMASCENO

BUSCA LOCAL ITERADA APLICADA AO SEQUENCIAMENTO DE TAREFAS EM MÁQUINAS FLEXÍVEIS PARALELAS NÃO IDÊNTICAS

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

Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

Ouro Preto, MG
2024

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Pedro Lucas Damasceno Silva

Busca Local Iterada Aplicada ao Sequenciamento de Tarefas em Máquinas Flexíveis Paralelas Não Idênticas

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 7 de Fevereiro de 2024.

Membros da banca:

Marco Antonio Moreira de Carvalho (Orientador) - Doutor - Universidade Federal de Ouro Preto
Rodrigo César Pedrosa Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto
Leonardo Cabral da Rocha Soares (Examinador) - Doutor - Instituto Federal do Sudeste de Minas Gerais - Campus Manhuaçu

Marco Antonio Moreira de Carvalho, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 7/02/2024.

Documento assinado eletronicamente por **Marco Antonio Moreira de Carvalho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 08/02/2024, às 13:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0665682** e o código CRC **4945F46E**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Resumo

O sequenciamento de tarefas em máquinas flexíveis paralelas não idênticas (*job sequencing and tool switching problem with non-identical parallel machines*, SSP-NPM) é um problema que consiste em designar e sequenciar um conjunto de tarefas às máquinas de um sistema de manufatura flexível (*flexible manufacturing system*, FMS). Um FMS é caracterizado pela união de máquinas flexíveis através de uma linha de produção automatizada. Cada máquina é equipada com um *magazine* de capacidade limitada, que deve comportar ferramentas o suficiente para a realização de qualquer tarefa individualmente. Uma tarefa é definida como o processo de fabricação de um produto, que requer o acoplamento imediato de várias ferramentas, como parafusadeiras, lixadeiras e outras, à máquina durante a sua execução. Comumente é impossível comportar simultaneamente todas as ferramentas do sistema, o que faz necessário a interrupção das máquinas para a realização das trocas necessárias a fim de dar sequência ao processo de produção. Este estudo descreve o desenvolvimento e a aplicação da metaheurística busca local iterada, combinada com as buscas locais estruturadas no formato de descida em vizinhança variável, para a resolução do SSP-NPM. Neste trabalho são abordados, separadamente, os objetivos de minimização do *makespan* (maior tempo decorrido, dentre todas as máquinas, desde o início da operação até o término da última tarefa processada) e *flowtime* (soma dos tempos de conclusão de todas as tarefas). Resultados melhores ou equivalentes foram obtidos, em comparação ao atual estado da arte, para quase todos os subconjuntos de instâncias da literatura.

Palavras-chave: Sequenciamento. Máquinas Flexíveis. Busca Local Iterada.

Abstract

The sequencing of jobs in non-identical parallel flexible machines (job sequencing and tool switching problem with non-identical parallel machines, SSP-NPM) is a problem that involves assigning and sequencing a set of jobs to machines in a flexible manufacturing system (FMS). An FMS is characterized by the integration of flexible machines through an automated production line. Each machine is equipped with a capacity-limited magazine, which must store enough tools for the processing of any individual job. A job is defined as the manufacturing process of a product, which requires the immediate installation of various tools, such as screwdrivers, sanders, and others, to the machine during its execution. It is commonly impossible to simultaneously accommodate all the tools in the system, which requires machine interruptions for the necessary tool switches in order to continue the production process. This study describes the development and application of the iterated local search metaheuristic, combined with structured local search procedures in the form of variable neighborhood descent, for solving the SSP-NPM. In this work, the objectives of minimizing makespan (the longest elapsed time among all machines from the start of operation to the completion of the last processed job), and flowtime (the sum of completion times of all jobs) are separately addressed. Better or equivalent results were achieved, compared to the current state of the art, for almost all subsets of instances from the literature.

Keywords: Sequencing, Flexible Machines, Iterated Local Search.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

CNC	comando numérico computadorizado
FMS	*flexible manufacturing system*
FMAX	*makespan*
GPCA	*greedy pipe construction algorithm*
ILS	busca local iterada (*iterated local search*)
KTNS	*keep tool needed soonest*
MTSP	*minimization of tool switches problem*
RRT	*randomly removed tool*
SSP	*job sequencing and tool switching problem*
SSP-NPM	*job sequencing and tool switching problem with non-identical parallel machines*
TFT	*flowtime*
TS	trocas de ferramentas (*tool switches*)
VND	descida em vizinhança variável (*variable neighborhood descent*)

Lista de Símbolos

σ	desvio padrão
μ	média geral
T	tempo de execução

Sumário

1 Introdução

As companhias de manufatura estão enfrentando desafios devido às recentes transformações do mercado consumidor e o interesse crescente por produtos customizados (Zawadzki; Żywicki, 2016). Segundo o modelo fabril tradicional, uma nova linha de produção se faz necessária para produzir cada diferente produto; todavia, além dos entraves logísticos e financeiros envolvidos, a volatilidade com que os interesses do mercado se alteram (Panfilova; Okrepilov; Kuzmina, 2018) também inviabiliza essa forma de produção nos dias atuais. Dessa forma, para se adaptar à nova realidade socioeconômica e se manterem competitivas no mercado, as fábricas têm adotado o modelo de manufatura flexível, fato que deu origem aos conceitos de "indústria 4.0" e "*smart factories*" (Shrouf; Ordieres; Miragliotta, 2014).

As principais características das *smart factories* são a alta flexibilidade, a automação da produção, a coleta e análise de dados em tempo real e a utilização de sistemas de máquinas flexíveis de manufatura (*flexible manufacturing system*, FMS) (Yadav; Jayswal, 2018). Um FMS é caracterizado pela união de máquinas flexíveis através de uma linha de produção automatizada. Cada máquina é equipada com um *magazine* de capacidade limitada, que deve comportar ferramentas suficientes para a realização de qualquer tarefa individualmente. Uma tarefa pode ser descrita como o processo de fabricação de um produto, para o qual várias ferramentas, como parafusadeiras, lixadeiras e outras, devem estar prontamente acopladas à máquina durante a execução da tarefa. Comumente, é impossível comportar simultaneamente todas as ferramentas do sistema, o que torna necessário a interrupção das máquinas para a realização das trocas necessárias a fim de dar sequência ao processo de produção. Na maioria dos sistemas reais de manufatura, a troca de ferramentas e/ou componentes é o processo que consome mais tempo (Hop, 2005) e deve ser evitada sempre que possível, dadas as implicações financeiras decorrentes de uma linha de produção ociosa.

A fim de obter a melhor ordenação de um conjunto de tarefas, de modo a minimizar as trocas de ferramentas necessárias, foi definido o *minimization of tool switches problem* (MTSP) (Tang; Denardo, 1988), atualmente referido na literatura como *job sequencing and tool switching problem* (SSP). Nele estão contidos dois problemas: o sequenciamento das tarefas e a determinação do plano de trocas de ferramentas. O sequenciamento é um problema $\mathcal{NP}$-Difícil (Crama *et al.*, 1994), ou seja, não se conhece algoritmo capaz de solucioná-lo de forma exata em tempo determinístico polinomial. Já a determinação do plano de trocas pode ser realizada em tempo determinístico polinomial através da política de manutenção das ferramentas que serão utilizadas mais cedo, denominada *keep tool needed soonest* (KTNS) (Tang; Denardo, 1988), de complexidade $O(mn)$, onde m corresponde à quantidade de ferramentas do sistema e n corresponde às tarefas atribuídas à máquina, ou através do algoritmo *greedy pipe construction algorithm* (GPCA) (Cherniavskii; Goldengorin, 2022), de complexidade $O(cn)$, em que c corresponde à capacidade

do *magazine*.

Nesse contexto, diversas variações do problema original surgiram de forma a retratar diferentes cenários industriais. Este estudo aborda a variação do SSP em que há um conjunto de máquinas flexíveis não-idênticas, conhecido como problema de sequenciamento de tarefas em máquinas flexíveis paralelas não idênticas (*job sequencing and tool switching problem with non-identical parallel machines*, SSP-NPM), definido primeiramente por Calmels (2022).

Este trabalho considera como objetivos a minimização do *makespan* (maior tempo decorrido, dentre todas as máquinas, desde o início da operação até o término da última tarefa processada, FMAX) e o *flowtime* (soma dos tempos de conclusão de todas as tarefas, TFT), separadamente. O objetivo de minimizar a soma das trocas de ferramentas foi descartado por não se alinhar de maneira coerente com o contexto do problema. Para a minimização dos objetivos em questão, é proposta a aplicação da metaheurística busca local iterada (*iterated local search*, ILS) (Lourenço; Martin; Stützle, 2003), combinada com as buscas locais estruturadas no formato de descida em vizinhança variável (*variable neighborhood descent*, VND) (Hansen *et al.*, 2019). Os resultados obtidos são comparados ao atual estado da arte (Soares; Carvalho, 2024), utilizando o mesmo conjunto de instâncias proposto por Calmels (2022) e sob as mesmas limitações de tempo de execução.

1.1 Justificativa

O SSP possui inúmeras ocorrências em ambientes industriais, e problemas relacionados existem em diferentes áreas de aplicação. Um exemplo de aplicação do SSP em indústrias de manufatura pode ser encontrada na indústria eletrônica para o sequenciamento da produção de placas de circuito impresso (*printed circuit board*, PCB), quando diferentes componentes eletrônicos precisam ser montados por máquinas de fabricação de componentes nas PCBs (Wu *et al.*, 2022). As máquinas possuem uma capacidade limitada de alimentadores de componentes, de modo que trocas de componentes se tornam necessárias ao montar diferentes tipos de PCBs. Nesse contexto, o objetivo predominante é minimizar o número de trocas de componentes.

Outro exemplo de aplicação do SSP pode ser encontrado na indústria de empacotamento, na qual os rótulos das embalagens devem ser impressos com cores diversas. Nesse contexto, o objetivo mais comum é a minimização do tempo gasto – normalmente dependente da sequência das tarefas – para a limpeza da máquina e troca dos cartuchos de tinta (Burger *et al.*, 2015).

O SSP é um problema da classe $\mathcal{NP}$-Difícil (Crama *et al.*, 1994), o que o confere profunda complexidade teórica. Dado o desconhecimento de um algoritmo capaz de solucionar o problema em tempo determinístico polinomial, o desenvolvimento de métodos capazes de produzir soluções aproximadas tem se provado de suma importância no contexto socioeconômico atual (Calmels, 2019).

1.2 Objetivos

Em linhas gerais, o principal objetivo deste trabalho é a produção de uma nova determinação da metaheurística ILS, com propriedades específicas para o SSP-NPM, e aplicá-la ao problema a fim de produzir melhores resultados em comparação ao atual estado da arte. São objetivos específicos:

1. Realizar uma revisão dos trabalhos relacionados ao SSP-NPM e suas variações, com foco especial nas abordagens que lidam com máquinas em paralelo, a fim de compreender as diferentes técnicas utilizadas e identificar as melhores abordagens para o problema.

2. Desenvolver uma nova metaheurística ou adaptar alguma já existente ao SSP-NPM, incorporando elementos específicos relacionados ao problema.

3. Realizar experimentos computacionais para configurar o método desenvolvido, analisar suas propriedades a partir das estatísticas de execução e comparar os resultados obtidos com o atual estado da arte.

1.3 Organização do Trabalho

O restante deste trabalho está organizado como descrito a seguir. O Capítulo 2 apresenta a revisão da literatura relevante em relação à variação estudada do SSP. Em seguida, no Capítulo 3, uma base teórica sólida é estabelecida, abrangendo a definição do problema, os objetivos específicos considerados neste trabalho e uma introdução às técnicas empregadas no seu desenvolvimento. O Capítulo 4 discute minuciosamente a implementação das técnicas incorporadas no algoritmo, enquanto o Capítulo 5 apresenta os resultados obtidos pelo método desenvolvido, bem como uma comparação com o estado da arte. Por fim, o Capítulo 6 encerra este trabalho, fornecendo uma síntese das conclusões alcançadas e delineando as próximas etapas de pesquisa.

2 Revisão Bibliográfica

Conforme mencionado anteriormente, o SSP apresenta diversas variações, criadas para modelar diferentes contextos industriais. Por conseguinte, a revisão da literatura foi dividida em dois segmentos: um focado em trabalhos que abordam o problema uniforme, e outro dedicado a estudos que tratam de variações que consideram múltiplas máquinas em paralelo.

2.1 Versão uniforme

A literatura sobre o SSP conta com diversos trabalhos e uma ampla gama de variações. O SSP uniforme para uma única máquina, com tamanhos uniformes de ferramentas e tempos de configuração iguais e independentes de sequência, foi introduzido por Tang e Denardo (1988), que desenvolveram a política KTNS para solucionar o subproblema de determinação do plano de trocas de ferramentas em tempo polinomial. No mesmo ano, Bard (1988) formulou o problema de troca de ferramentas como programação não linear (PNL) e aplicou uma abordagem de relaxamento Lagrangiano para obter uma sequência de tarefas com alocação de ferramentas viável, seguido por uma técnica de busca local usando KTNS para explorar sequências vizinhas. Até então, as abordagens supracitadas não foram capazes de prover bons resultados mesmo para instâncias pequenas do problema. Em seguida, Crama *et al.* (1994) provaram que o SSP se trata de um problema $\mathcal{NP}$-Difícil, ou seja, não se conhece algoritmo capaz de solucioná-lo de forma determinística em tempo polinomial.

Diferentes modelagens do problema foram desenvolvidas nos anos seguintes como, por exemplo, a modelagem baseada em fluxo em redes proposta por Yanasse e Pinto (2002). Dada a classe de problemas do SSP, a proposta se provou promissora para instâncias práticas de pequeno porte ou com capacidades de armazenamento de ferramentas reduzidas. Em sequência, Senne e Yanasse (2009) propuseram três heurísticas baseadas no algoritmo *beam search* para o problema de minimização de trocas de ferramentas em um sistema de manufatura flexível. A primeira heurística, denominada *beam search algorithm* (BSA), utiliza uma abordagem de busca em largura para explorar o espaço de soluções em busca da sequência de tarefas com o menor número de trocas de ferramentas. A segunda heurística incorpora melhorias ao BSA, como a consideração de fatores de balanceamento de carga entre as máquinas e o uso de um mecanismo de vizinhança adaptativa para explorar soluções vizinhas de forma mais eficiente. A terceira heurística estende a segunda ao introduzir um mecanismo adaptativo para ajustar dinamicamente o tamanho do feixe (*beam*) durante a busca. Isso permite uma busca mais eficiente, ajustando o tamanho do feixe de acordo com a progressão da busca e a qualidade das soluções encontradas.

Diversas implementações de algoritmos meméticos – uma extensão do algoritmo genético tradicional – foram propostas em Amaya, Cotta e Fernández (2008), Amaya, Cotta e

Leiva (2010b), Amaya, Cotta e Leiva (2010a), Amaya, Cotta e Fernández-Leiva (2011), Amaya, Cotta e Fernandez-Leiva (2012) e Amaya, Cotta e Fernández-Leiva (2013). Em termos gerais, os algoritmos propostos combinam o clássico algoritmo genético com procedimentos de busca local, incluindo uma estrutura de vizinhança conhecida como *all-pairs* e uma abordagem de escalada mais íngreme. Além disso, esses trabalhos exploraram algoritmos cooperativos de busca, demonstrando que agentes meméticos com características heterogêneas e troca das melhores soluções superam agentes individuais. Também foram introduzidos algoritmos meméticos baseados em entropia cruzada, que utilizam diferentes componentes de busca local, e obtiveram um desempenho promissor em comparação com os algoritmos meméticos clássicos, especialmente quando utilizam múltiplas funções de massa de probabilidade.

Paiva e Carvalho (2017) propuseram uma metodologia que utiliza representação em grafos, uma heurística construtiva e uma nova determinação da metaheurística ILS para o SSP uniforme. Conforme explicado adiante no Capítulo 3, as trocas de ferramentas podem ser representadas por inversões de 0 para 1 nas matrizes binárias que representam o sequenciamento de tarefas em uma máquina. Diante disso, o método proposto se destaca na introdução da busca local *1-block grouping*, que visa minimizar os blocos de uns consecutivos na matriz em questão. O método superou os melhores resultados obtidos até então para todos os conjuntos de instâncias considerados na literatura.

O estado da arte atual para o SSP uniforme foi estabelecido por Mecler, Subramanian e Vidal (2021), que introduziram um algoritmo genético híbrido. Além de apresentar os melhores resultados até então para os conjuntos de instâncias da literatura, o trabalho em questão também propôs um novo conjunto de instâncias com o intuito de representar cenários mais fidedignos à realidade, para estimular futuras pesquisas e para permitir a comparação em instâncias mais competitivas.

2.2 Variações com múltiplas máquinas

A primeira variação a considerar múltiplas máquinas em paralelo foi definida por Bard (1988), onde constam múltiplas etapas com diferentes máquinas interligadas. No trabalho em questão, foi proposto um modelo de PNL para o problema de troca de ferramentas com múltiplas máquinas conectadas sequencialmente e soluções ótimas foram obtidas para cenários com até 2 máquinas, 20 tarefas e 36 ferramentas. Posteriormente, Khan *et al.* (2000) apresentou uma heurística gulosa para minimizar as trocas de ferramentas em dois centros de usinagem controlados por sistemas de comando numérico computadorizado (CNC) paralelos e idênticos.

Fathi e Barnette (2002) apresentaram três heurísticas para múltiplas máquinas em paralelo. Um procedimento de melhoria testado para combinações de diferentes estratégias de busca local (inserção e troca) foi comparado a uma heurística de processamento de lista e uma heurística construtiva utiliza *multi-start* para o problema correspondente de máquina única e, em seguida,

gera uma subsequência para cada máquina. Os resultados mais promissores foram obtidos a partir dos procedimentos de busca local combinados.

Özpeynirci, Gökgür e Hnich (2016) e Gökgür, Hnich e Özpeynirci (2018) abordaram a variação do SSP que considera máquinas paralelas não relacionadas a fim de minimizar o *makespan* levando em consideração restrições de quantidade de cópias de cada ferramenta no sistema, mas desconsiderando restrições de capacidade ou tempo de troca de ferramentas. Ambos os trabalhos apresentam um modelo de programação inteira e a aplicação da metaheurística busca tabu para a resolução do problema envolvendo conjuntos de instâncias que consideram até 3 máquinas, 15 tarefas e 8 ferramentas, e a comparação dos resultados obtidos por cada método é apresentada.

Beezão *et al.* (2017) apresentaram dois modelos de programação linear inteira (PLI) para o problema de troca de ferramentas com máquinas paralelas e tempos de processamento específicos. Ambas as formulações demonstraram dificuldades para solucionar instâncias de 15 tarefas. Diante disso, foi desenvolvida uma variação da metheurística busca adaptativa em grande vizinhança (*adaptative large neighborhood search*, ALNS), que é comparadada às heurísticas de melhoria de Fathi e Barnette (2002). Mesmo para instâncias grandes, com até 200 trabalhos, 10 máquinas e 40 ferramentas, a ALNS supera claramente os métodos dos trabalhos anteriores.

Soares e Carvalho (2020) abordaram a variação do problema que considera máquinas paralelas com restrições de ferramentas através da aplicação da meta-heurística *biased random-key genetic algorithm* (BRKGA) (Gonçalves; Resende, 2011) combinada com procedimentos de busca local. Os resultados obtidos foram comparados aos dois conjuntos do único *benchmark* de instâncias, cada um contendo 1440 unidades, disponível na literatura e se provaram melhores ou equivalentes em 99,86% das instâncias, estabelecendo o novo estado da arte para a variação do SSP em questão. Ademais, o tempo de execução reportado pelo método desenvolvido foi 96,50% mais rápido em relação a Beezão *et al.* (2017).

Em sequência, Soares e Carvalho (2022) apresentaram uma nova aplicação do BRKGA, também hibridizado com buscas locais, à variação que considera múltiplas máquinas idênticas, com mesmo tempo de *setup*, e com ferramentas compartilhadas. Os experimentos computacionais realizados constataram a trivialidade do único *benchmark* existente na literatura até então. Diante disso, novas 270 instâncias foram propostas e o método proposto reportou um gap médio de -11,14% em comparação aos melhores resultados da literatura até então.

2.2.1 Máquinas paralelas não idênticas

Uma revisão geral da literatura acerca do SSP e suas variações foi elaborada por Calmels (2019), que abrangeu desde a primeira definição do problema até as variações reportadas no ano de 2018. Em Calmels (2022), a variação inédita do SSP abordada neste trabalho foi definida, a qual considera máquinas flexíveis paralelas não idênticas. No referido artigo, foi estabelecida

uma formulação matemática para o problema e um método de busca local iterada, ambos com tempo computacional limitado a 3.600 segundos, foi implementado para propor soluções para um conjunto de 640 instâncias, também propostas no mesmo estudo. Como objetivos foram considerados a minimização do *makespan*, *flowtime* e trocas de ferramentas, sendo o último de relevância questionável no contexto de máquinas paralelas com diferentes capacidades do *magazine*, como detalhado no Capítulo 4. O artigo exibe os resultados obtidos pelo método desenvolvido com os reportados pelo modelo de programação inteira mista (MIP). O MIP foi capaz de gerar a solução ótima para todos os objetivos apenas nos dois menores subconjuntos de instâncias e para alguns subconjuntos pouco maiores no objetivo de trocas de ferramentas, justificando o uso de métodos aproximados. Comparando os resultados, é possível concluir que o método heurístico desenvolvido por Calmels (2022) foi capaz de produzir boas soluções, inclusive algumas ótimas, para os menores subconjuntos de instâncias e com baixo desvio padrão.

Em Cura (2023), um método baseado na hibridização entre buscas locais e algoritmo genético foi elaborado, resultando em soluções com uma melhoria média de 13,34% em relação aos resultados reportados por Calmels (2022), considerando as mesmas 640 instâncias supracitadas. No trabalho em questão, o autor utilizou, além do KTNS, a política *randomly removed tool* (RRT) para obter uma resolução mais rápida, embora não necessariamente ótima, do plano de trocas de ferramentas e diferentes operadores de busca local. São eles: o operador de *swap*, que troca tarefas entre máquinas; o operador de inserção, que realoca as tarefas para posições mais adequadas nos sequenciamentos das máquinas; o operador de inserção de blocos, que realoca blocos de tarefas no sequenciamento de uma mesma máquina, e o algoritmo genético puro. Por fim, foram expostos os resultados obtidos através de diversos experimentos, também limitados a uma hora de execução por instância, que combinaram as diferentes políticas de resolução do plano de trocas e operadores de buscas locais. Foi possível concluir que determinaro plano de trocas através do KTNS se provou mais eficiente ao fornecer resultados melhores, e que o algoritmo genético possui grande potencial dada sua característica de processamento em paralelo e a forte tendência do mercado a produzir *hardware* com cada vez mais núcleos de processamento.

O estado da arte para o SSP-NPM foi redefinido em 2024 por Soares e Carvalho (2024), que apresentaram duas versões da metaheurística BRKGA (referidas no trabalho como BRKGA$_{Cmax}$ e BRKGA$_{TFT}$) hibridizadas com três procedimentos de busca local, organizados em um VND, para especializar o método ao SSP-NPM. No método em questão, o VND é aplicado à população elite, e as buscas locais são aplicadas linearmente aos demais indivíduos da geração atual. Essa abordagem confere ao método uma orientação mais eficiente na exploração do espaço de soluções, resultando na geração de populações melhor adaptadas à função objetivo já nas primeiras gerações. Os resultados reportados pelos métodos foram melhores ou equivalentes em comparação a Cura (2023) para todos os subconjuntos de instâncias, alcançando uma distância percentual de -12,38% e -11,90% para os objetivos de minimização do *makespan* e *flowtime*, respectivamente, e testes estatísticos confirmaram a diferença significativa entre os resultados comparados. Além disso, outra contribuição deste trabalho foi a demonstração de como o objetivo de minimização

das trocas de ferramentas é incoerente com contextos práticos do problema.

Como evidenciado, toda a literatura sobre o SSP-NPM é muito recente, desde a definição proposta por Calmels (2022) à publicação do estado da arte, da autoria de Soares e Carvalho (2024). Este trabalho se situa em sequência, e propõe uma nova determinação da metaheurística ILS aplicada ao SSP-NPM.

3 Fundamentação Teórica

A fundamentação teórica deste estudo tem como objetivo fornecer uma base sólida para o desenvolvimento da pesquisa, incluindo ambos a formulação matemática do problema em questão e a explanação sobre as heurísticas adotadas ao longo do desenvolvimento deste trabalho.

3.1 Definição

Formalmente, dado um FMS contendo um conjunto de ferramentas $T = \{1, \ldots, t\}$; um conjunto de máquinas flexíveis não-idênticas paralelas $M = \{1, \ldots, m\}$, com capacidades para comportar até c_i ($i \in M$) ferramentas simultaneamente e um conjunto de tarefas $J = \{1, \ldots, j\}$ a serem processadas, o SSP-NPM consiste em determinar a designação e sequenciamento das tarefas em cada máquina. Além das capacidades dos *magazines* C_i ($i \in M$), as máquinas também diferem em relação ao tempo de processamento de cada tarefa p_{ij} ($i \in M$, $j \in J$) e ao tempo necessário para realização de uma troca de ferramentas sw_i ($i \in M$). Embora não seja explicitado na definição do problema, por Calmels (2022), as instâncias desenvolvidas no referido trabalho possuem restrições de elegibilidade de máquina; isto é, há instâncias onde algumas tarefas requerem mais ferramentas do que uma ou mais máquinas do sistema são capazes de comportar. Assim como nos trabalhos anteriores, este também respeita essa restrição e não atribui às máquinas tarefas cuja necessidade de ferramentas exceda a capacidade de seu magazine.

Tabela 3.1 – Matriz N_{tf} de ferramentas requeridas por tarefa da instância de exemplo.

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

Para exemplificar o SSP-NPM, considere a instância “ins1_m=2_j=10_t=10_var=1”, que descreve um FMS com $M = \{1, 2\}$, $T = \{1, \ldots, 10\}$, $J = \{1, \ldots, 10\}$, $C = \{5, 7\}$ e $sw = \{2, 4\}$. Os requisitos de ferramentas por tarefa de uma instância são comumente expressos por meio de uma matriz binária N_{tf}. Nessa representação, as linhas correspondem às ferramentas, enquanto as colunas representam as tarefas. Valores de $n_{tf} = 1$ indicam que a ferramenta t deve estar acoplada

à máquina durante o processamento da tarefa f, e $n_{tf} = 0$ significa que a ferramenta t não é necessária para o processamento da tarefa f. De forma similar, também é possível utilizar uma matriz binária para representar a necessidade de ferramentas e o sequenciamento das tarefas em uma máquina. Nessa representação, temos que $R^i = \{r^i_{tj}\}$ ($i \in M$, $t \in T$, $j \in J$), na qual as linhas representam as ferramentas e as colunas representam as tarefas designadas à máquina i, em ordem de processamento. A Tabela 3.1 exemplifica a representação de uma matriz N_{tf} e a Tabela 3.2 exemplifica a representação de matrizes R^i.

Tabela 3.2 – Matrizes R^i de atribuição aleatória para a instância "ins1_m=2_j=10_t=10_var=1".

Máquina 1					
Ferramentas	10	7	9	3	8
1	1	0	0	1	1
2	0	1	0	0	1
3	0	1	1	1	1
4	1	0	1	0	0
5	0	1	1	1	1
6	1	0	1	0	0
7	0	0	0	1	1
8	1	0	0	0	0
9	1	1	0	1	0
10	0	0	1	0	0

Máquina 2					
Ferramentas	1	4	6	2	5
1	0	0	0	0	1
2	1	0	0	1	1
3	0	1	1	0	0
4	1	1	0	0	0
5	0	0	1	1	0
6	0	0	0	0	0
7	0	0	1	1	1
8	1	0	0	0	0
9	0	1	1	0	0
10	0	0	1	1	0

O número de trocas de ferramentas no FMS é calculado pela Equação (3.1). A equação soma as inversões de 0 para 1 nas matrizes R^i, que representam as trocas de ferramentas necessárias para a conclusão do processamento do subconjunto A_i de tarefas atribuídas à máquina i, tal que $i \in M$ e $A_i \subseteq J$. Nesta variação do SSP, o carregamento inicial do *magazine* não é contabilizado como trocas de ferramentas. Na atribuição aleatória da Tabela 3.2, são realizadas 11 trocas no total; em uma solução ótima para esse objetivo, são realizadas apenas 3.

$$\text{TS} = \sum_{i \in M} \sum_{j=2}^{j=|A_i|} \sum_{t \in T} r^i_{tj}(1 - r^i_{tj-1}) \tag{3.1}$$

O *makespan* é definido pela Equação (3.2), que soma o tempo de processamento das tarefas e o tempo gasto para efetuar as trocas de ferramentas individualmente para cada máquina do FMS. O valor do *makespan* corresponde ao maior tempo de término de processamento de uma máquina do FMS para a conclusão do seu subconjunto de tarefas designadas. Na atribuição aleatória da Tabela 3.2, o valor do *makespan* é de 55 unidades de tempo, enquanto em uma solução ótima seria de apenas 29.

$$\text{FMAX} = \arg\max \left\{ \sum_{j \in A_i} p_{ij} + sw_i \times \sum_{j=2}^{j=|A_i|} \sum_{t \in T} r^i_{tj}(1 - r^i_{tj-1}) \right\}, \forall i \in M \tag{3.2}$$

Por fim, o *flowtime* é calculado pela soma das variáveis k_i, que marcam os tempos de completude de cada uma das as tarefas processadas no FMS, como descrito na Equação (3.3). O tempo de completude de uma tarefa é calculado somando todo o tempo gasto pelas tarefas e trocas de ferramentas anteriores até a conclusão da tarefa em questão. Para a atribuição aleatória da Tabela 3.2, o *flowtime* total é de 274 unidades de tempo, enquanto a solução ótima para esse objetivo possui valor de 129.

$$\text{TFT} = \sum_{i \in J} k_i \tag{3.3}$$

Conforme demonstrado por Soares e Carvalho (2024), o objetivo de minimização de trocas de ferramentas é incoerente com contextos práticos. Isso porque, em um sistema com máquinas não-idênticas notadamente distintas em termos de capacidade do *magazine*, a minimização das trocas resulta na concentração de tarefas nas máquinas com maior capacidade de *magazine*. Isso produz ociosidade injustificada nas demais máquinas do sistema, resultando em tempos de produção proibitivos. Por conta disso, o objetivo também foi desconsiderado neste trabalho.

3.2 Busca local iterada

Partindo de uma solução inicial, a ILS utiliza buscas locais para aprimorar essa solução, explorando as vizinhanças e movendo-se para soluções melhores sempre que possível. Em seguida, o método introduz um elemento de diversidade por meio da perturbação, que altera aleatoriamente a solução para escapar de mínimos locais e regiões de baixa qualidade do espaço de busca.

Algoritmo 1 Busca Local Iterada

```
1:  s ← heuristicaConstrutiva();
2:  s ← buscasLocais(s);
3:  iteracoes ← 0
4:  enquanto iteracoes < maxIteracoes faça
5:      s' ← perturbarSolucao(s);
6:      s' ← buscasLocais(s');
7:      se s' ⩽ s então
8:          s ← s';
9:      fim se
10:     iteracoes ← iteracoes + 1;
11: fim enquanto
```

A cada iteração, a solução resultante da perturbação e das buscas locais é comparada com a melhor solução encontrada até o momento. Se a nova solução for melhor, ela substitui a melhor solução anterior. Esse processo de refinamento iterativo continua por um número definido de iterações, por tempo ou por outra condição estipulada. O pseudocódigo da ILS pode ser observado no Algoritmo 1.

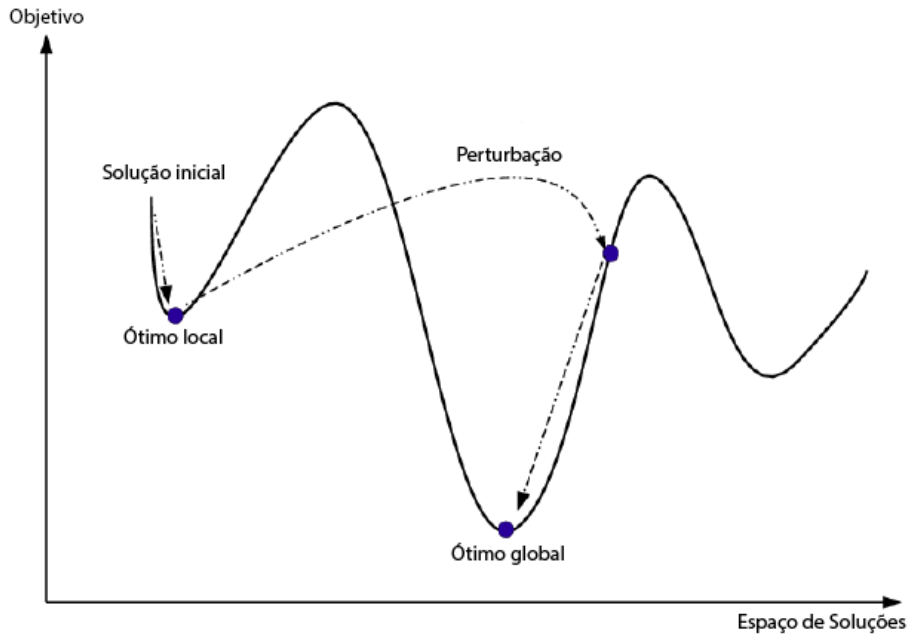

Figura 3.1 – Ilustração da trajetória realizada pela ILS. Adaptado de Hamon *et al.* (2013).

Uma ilustração da trajetória realizada pela ILS pode ser observada na Figura 3.1. Nela é possível observar a solução inicial caminhando para um ótimo local através das buscas locais, o deslocamento do espaço de soluções sob análise devido à perturbação e, por fim, o descobrimento do ótimo global através da exploração da nova vizinhança. É importante ressaltar que, em problemas combinatórios como o SSP-NPM, o espaço de soluções adquire proporções muito grandes, mesmo em instâncias relativamente pequenas. Dessa forma, faz-se importante o desenvolvimento de uma boa solução inicial pois, partindo de uma solução razoável, menos iterações são necessárias para que o método convirja a um ótimo local de boa qualidade ou até mesmo ao ótimo global.

A ILS possui alguns parâmetros a serem definidos, como o número máximo de iterações, o grau de perturbação utilizado, o critério de aceitação e outros. Tais parâmetros foram calibrados seguindo testes estatísticos detalhados no Capítulo 5.

3.3 Descida em vizinhança variável

As buscas locais realizadas pela ILS, detalhadas na Seção 4.2, foram estruturadas no formato de descida em vizinhança variável (VND) (Hansen *et al.*, 2019), seguindo a lógica de *first improvement*. Dessa forma, a busca reinicia a partir da primeira melhora encontrada na vizinhança da solução sob análise. O Algoritmo 2 detalha a implementação do VND e a disposição das buscas locais, também determinada por testes estatísticos descritos no Capítulo 5.

Algoritmo 2 Descida de Vizinhança Variável

```
1: k ← 1
2: enquanto k ≠ 6 faça
3:     se k = 1 então
4:         se jobExchangeLocalSearch(s') então k ← 1 senão k ← k + 1;
5:     fim se
6:     se k = 2 então
7:         se jobInsertionLocalSearch(s') então k ← 1 senão k ← k + 1;
8:     fim se
9:     se k = 3 então
10:        se swapLocalSearch(s') então k ← 1 senão k ← k + 1;
11:    fim se
12:    se k = 4 então
13:        se twoOptLocalSearch(s') então k ← 1 senão k ← k + 1;
14:    fim se
15:    se k = 5 então
16:        se oneBlockLocalSearch(s') então k ← 1 senão k ← k + 1;
17:    fim se
18: fim enquanto
```

A ideia central do VND é alternar entre diferentes estruturas de vizinhança, cada uma definida por um conjunto específico de movimentos ou operadores. Essa variação de vizinhanças permite explorar diferentes regiões do espaço de busca de forma mais abrangente, aumentando as chances de encontrar soluções ótimas locais.

4 Desenvolvimento

Este trabalho combina a metaheurística ILS – uma técnica que consiste em gerar uma solução inicial e realizar buscas locais e perturbações a fim de abrangir diferentes vizinhanças até que se atinja o critério de parada especificado – ao VND, que explora diferentes regiões até que um ótimo local comum a todas seja encontrado. Ademais, buscas locais diferentes dos trabalhos anteriores relacionados ao SSP-NPM foram implementadas e calibradas como descrito na Seção 5.1. A implementação do método e os resultados obtidos estão disponíveis *online* [1].

4.1 Heurística construtiva

A qualidade da solução inicial é de suma importância para a eficiência da ILS. Ao partir de uma solução ruim, mais iterações e, consequentemente, mais tempo são necessários para que ocorra a convergência da solução em direção a ótimos locais de menor custo. Diante disso, foi observado que o critério de maior relevância para a minimização de todos os objetivos tratados neste trabalho é a redução das trocas de ferramentas. Isso porque as diferenças dos tempos de processamento das tarefas é significativamente menos relevante em relação ao tempo consumido para a realização das trocas.

Tabela 4.1 – Solução inicial construída a partir da instância exemplificada na Tabela 3.1, cujos valores distam de apenas 3 e 38 unidades para as soluções ótimas em relação aos objetivos FMAX e TFT, respectivamente.

Máquina 1				
Ferramentas	0	9	3	8
1	0	1	0	0
2	1	0	0	0
3	0	0	1	1
4	1	1	1	1
5	0	0	0	1
6	0	1	0	1
7	0	0	0	0
8	1	1	0	0
9	0	1	1	0
10	0	0	0	1

Máquina 2						
Ferramentas	6	7	4	1	5	2
1	0	1	1	0	0	1
2	1	1	1	1	0	0
3	1	1	0	0	1	1
4	0	0	0	0	0	0
5	1	1	0	1	1	1
6	0	0	0	0	0	0
7	0	1	1	1	1	1
8	0	0	0	0	0	0
9	1	0	0	0	1	1
10	0	0	0	1	1	0

A heurística proposta para a construção de soluções iniciais baseia-se na minimização das inversões de 0 para 1 nas matrizes R^m. Para isso, cada máquina do FMS recebe uma tarefa aleatória. Em seguida, seleciona-se a máquina com a menor razão de trocas de ferramentas por tarefas atribuídas em todo o FMS. A essa máquina é atribuída a tarefa mais similar à última do

[1] <https://github.com/pedroldm/ic_ssp_npm>

seu sequenciamento. A similaridade é calculada somando-se todas as ferramentas necessárias e desnecessárias em comum entre a última tarefa atribuída à máquina e todas as outras tarefas ainda não atribuídas.

4.2 Buscas locais

As buscas locais são responsáveis por explorar a vizinhança da solução sob análise, realizando movimentos que a alteram de forma pontual. É importante que sejam métodos complementares, de forma que cada busca altere a solução de formas diferentes a fim de evitar a reanálise e a convergência prematura para um ótimo local não global. Neste estudo foram consideradas buscas locais de amplo uso no âmbito do SSP, e algumas são inéditas em relação a trabalhos anteriores sobre o SSP-NPM.

A fim de limitar o espaço de busca nas vizinhanças e evitar a realização e análise de movimentos improdutivos, um novo critério de similaridade de tarefas foi adicionado às buscas locais realizadas durante o laço iterativo principal da ILS. Após o refinamento da solução inicial – que desconsidera o critério em pauta – gerado pela heurística construtiva, as buscas locais só realizam movimentos cujas tarefas envolvidas possuam determinado grau de similaridade em relação ao conjunto de ferramentas requeridas. Essa abordagem se baseia na percepção de que colunas com muitas distinções implicam em mais inversões na matriz binária R^i – que já terá as tarefas mais similares aproximadas após o refinamento da solução inicial –, o que se traduz no aumento do número de trocas de ferramentas realizadas.

Para exemplificar, considere um grau de similaridade de 80% e a matriz N_{tf} da Tabela 3.1. Nela é possível observar que as tarefas 3 e 6 possuem apenas duas diferenças no conjunto de tarefas requeridas e, portanto, movimentos que envolvam essas duas tarefas seriam realizados e analisados. Já as tarefas 1 e 2 possuem cinco diferenças no conjunto de ferramentas requeridas e, portanto, movimentos que envolvam essas duas tarefas não seriam realizados e analisados.

4.2.1 Busca local *job exchange*

Nesta busca local, um movimento consiste em trocar de posição duas tarefas de uma mesma máquina. Para o objetivo de minimização do TFT, essa busca abrange, a princípio, todos os pares de tarefas dos sequenciamentos do FMS, o que a confere complexidade assintótica $\mathcal{O}(|J|^2)$. Para o objetivo de minimização do FMAX, são consideradas apenas trocas de tarefas na máquina crítica – a responsável pelo *makespan* –. Considerando o exemplo da Tabela 3.2, um possível movimento consiste em trocar de posição as tarefas 10 e 3 da máquina 1, conforme exemplificado na Figura 4.1.

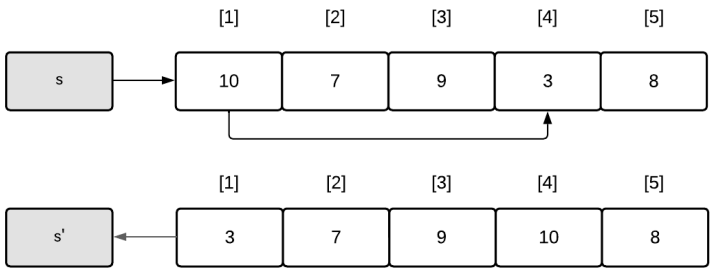

Figura 4.1 – Exemplo de movimento realizado pela busca local *job exchange*.

4.2.2 Busca local *job insertion*

Nesta busca local, um movimento consiste no reposicionamento de uma tarefa. A tarefa é retirada do sequenciamento de uma máquina e, em seguida, inserida no sequenciamento de outra. A princípio, todas as possíveis posições para a tarefa retirada são verificadas, o que também confere a esta busca local a complexidade assintótica $\mathcal{O}(|J|^2)$. A Figura 4.2 exemplifica um movimento realizado pela busca local, no qual a tarefa 6 da máquina 1 é realocada para a 3ª posição do sequenciamento da máquina 2.

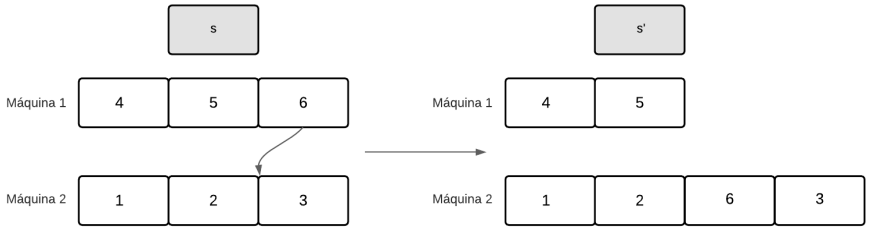

Figura 4.2 – Exemplo de movimento realizado pela busca local *job insertion*.

4.2.3 Busca local *swap*

Nesta busca local, tarefas de máquinas diferentes são permutadas contanto que a restrição de elegibilidade permita. Sua complexidade assintótica também é de $\mathcal{O}(|J|^2)$, e um possível movimento está exemplificado na Figura 4.3, na qual as tarefas 2 e 6, de máquinas diferentes, são permutadas para gerar uma nova solução.

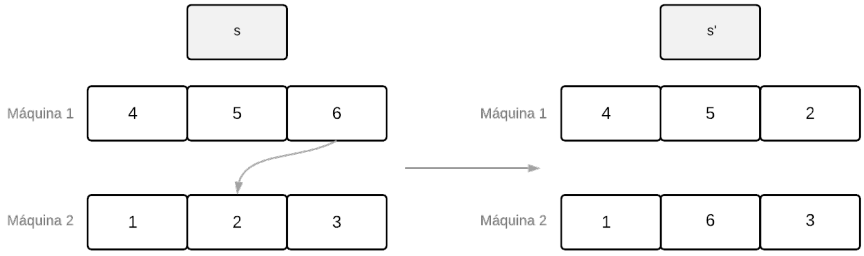

Figura 4.3 – Exemplo de movimento realizado pela busca local *swap*.

4.2.4 Busca local *2-opt*

O *2-opt* é um algoritmo heurístico originalmente utilizado para melhorar soluções aproximadas para o problema do caixeiro viajante. Neste trabalho, o *2-opt* foi adaptado de forma a inverter intervalos de tarefas dos sequenciamentos das máquinas do FMS. Para os objetivos de TS e TFT, todos os possíveis pares de tarefas são examinados, resultando na complexidade assintótica $\mathcal{O}(|J|^2)$. Para o objetivo de FMAX, apenas pares da máquina crítica são considerados. Considerando o exemplo da Tabela 3.2, um possível movimento seria a inversão do intervalo compreendido entre as tarefas 10 e 3 da máquina 1, conforme exemplificado na Figura 4.4.

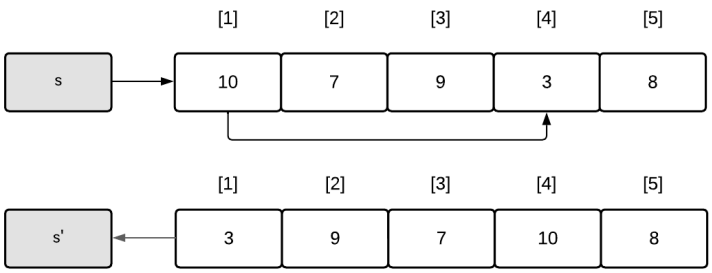

Figura 4.4 – Exemplo de movimento realizado pela busca local *2-opt*.

4.2.5 Busca local *1-block grouping*

O *1-block grouping* (Paiva; Carvalho, 2017) foi desenvolvido visando minimizar as inversões na matriz R^m. Primeiro, são identificados os blocos de um ou mais uns consecutivos em uma linha. Em seguida, as colunas dos blocos são permutadas a fim de aproximar os 1s e diminuir o número total de blocos. Entretanto, experimentos demonstraram que os movimentos realizados pelo *1-block grouping* não estavam produzindo soluções melhores. Isso se deve ao fato dos movimentos realizados estarem redundantes em relação aos da busca local *job exchange*. Por isso, o método foi levemente alterado de forma a mover blocos, e não colunas. A Tabela 4.2 ilustra um movimento realizado pela *1-block grouping*, onde bloco [3, 8] é movido para próximo do bloco [10], reduzindo para apenas um a quantidade de blocos de uns na primeira linha da matriz de sequenciamento.

Tabela 4.2 – Exemplo de movimento realizado pelo *1-block grouping*.

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

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

A complexidade assintótica do *1-block grouping* é $\mathcal{O}(|T||J|^2)$ no pior caso, o que pode implicar na necessidade de limitar o número de soluções vizinhas a serem analisadas. Além disso, pode haver semelhança de blocos de uns entre as linhas, o que ocasiona a reanálise de soluções.

4.3 Perturbação

A fim de direcionar o deslocamento da solução para vizinhanças mais promissoras, duas técnicas de perturbação foram adequadas à ILS. A primeira consiste na realocação da tarefa crítica – aquela cuja soma do tempo de processamento e do tempo necessário para a realização das trocas de ferramentas seja a maior – da máquina crítica. i.e., aquela cujo *makespan* seja o maior dentre todas do sistema. A segunda na realocação de uma tarefa aleatória da máquina crítica. O grau da perturbação e da utilização de cada perturbação também foi definido por testes estatísticos detalhados no Capítulo 5.

Algoritmo 3 Perturbação

```
1: tamanhoPerturbacao ← grau × qtdeTarefas;
2: enquanto tamanhoPerturbacao > 0 faça
3:     se rand()% <= porcentagemTarefaCritica então
4:         perturbacaoTarefaCritica();
5:     senão
6:         perturbacaoMaquinaCritica();
7:     fim se
8:     tamanhoPerturbacao ← tamanhoPerturbacao − 1;
9: fim enquanto
```

A primeira perturbação - a responsável por realocar a tarefa crítica da máquina crítica - possui uma porcentagem de viés responsável por direcionar a tarefa realocada à máquina com menor *makespan*. Essa porcentagem também foi determinada por testes estatísticos.

5 Experimentos Computacionais

O ambiente computacional utilizado para a realização dos experimentos consiste em um computador com processador Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, 16GB de memória RAM DDR4 3200MHz, e sob o sistema operacional Ubuntu 22.04.1 LTS. O método proposto foi implementado em C++ e compilado utilizando o GCC versão 11.3.0, com as opções de otimização -O3 e march=native. Assim como feito por Soares e Carvalho (2024), o tempo de execução máximo para uma instância foi de 3.600 segundos. Devido aos componentes de aleatoriedade, o método foi executado de forma independente 10 vezes para cada instância.

As instâncias propostas por Calmels (2022) estão disponíveis *online* [1]. Elas estão divididas em dois subconjuntos: SSP-NPM-I e SSP-NPM-II. O primeiro subconjunto consiste em grupos de 20 instâncias que envolvem no máximo 3 máquinas, 20 ferramentas e 20 tarefas. Já o segundo subconjunto é composto por grupos de 80 instâncias que possuem até 6 máquinas, 120 ferramentas e 120 tarefas. Além disso, o segundo subconjunto é subdividido em instâncias densas e esparsas – referente à quantidade de ferramentas necessárias para as tarefas – e em instâncias de alto e baixo custo de trocas de ferramentas.

5.1 Experimentos preliminares

Para ajustar os parâmetros da mais recente versão do algoritmo e contrastá-la com a sua predecessora, foram conduzidos testes estatísticos e experimentos reduzidos. Uma vez definida a melhor versão, os resultados reportados por ela são utilizados na comparação com o atual estado da arte, na Subseção 5.2.

5.1.1 Definição de parâmetros

A disposição das buscas locais, o número máximo de iterações realizados pela ILS, o grau de similaridade de tarefas que restringe as buscas locais e o método e grau de perturbação foram definidos conforme testes estatísticos realizados pelo pacote iRace (López-Ibáñez *et al.*, 2016). Todas as possibilidades de ordenação das buscas foram avaliadas, inclusive a utilização ou não de cada método. A disposição reportada para as buscas locais foi: *job exchange*, *job insertion*, *swap* e *2-opt* e a melhor configuração de parâmetros reportada está destacada em negrito na Tabela 5.1.

[1] <https://github.com/TerhiS/ILS_SSP-NPM>

Tabela 5.1 – Parâmetros calibrados pelo iRace.

Iterações	Grau de similaridade	Grau de perturbação	Perturbação da tarefa crítica	Viés de realocação
750	0,6	0,05	0,6	0,4
1000	**0,7**	0,075	0,65	**0,5**
1500	0,8	**0,085**	**0,7**	0,6
		0,1		0,7

5.1.2 Comparação entre versões

Uma vez concluído o desenvolvimento e a configuração da nova versão do método proposto, foi observado em algumas instâncias maiores que a nova versão não reportava resultados melhores ou até mesmo equivalentes à versão anterior. Diante disso, para avaliar a performance da nova versão e compará-la à versão anterior, foram realizados experimentos por amostragem.

Para as instâncias menores, do subconjunto SSP-NPM-I, os experimentos foram reproduzidos de forma idêntica aos realizados durante a experimentação da primeira versão do método. As médias dos melhores resultados reportados foram completamente equivalentes às da primeira versão, com variações irrisórias de centésimos.

Já para a instâncias maiores, do subconjunto SSP-NPM-II, foi realizada a amostragem dos conjuntos devido à grande quantidade de instâncias. Para representar os conjuntos fielmente, foram selecionadas aleatoriamente 5 instâncias de cada combinação de propriedades, sendo elas: instâncias densas ou esparsas e alto ou baixo custo de troca de ferramentas. Assim como na experimentação da versão anterior, cada instância foi resolvida 10 vezes, no mesmo ambiente computacional e sob o mesmo limite de tempo de execução. Os resultados obtidos para as 105 instâncias da amostra foram: 54,28% equivalentes, 40,00% piores e apenas 5,71% melhores. Em termos de diferença percentual, os resultados piores diferiram, em média, de 13,94% e os melhores de, em média, 1,72%, também em relação aos resultados reportados pela versão original do algoritmo.

A inferioridade dos resultados reportados pela segunda versão do método pode ser explicada pela imprevisibilidade do espaço de soluções e pelo impacto do viés na trajetória de busca. Ao adicionar componentes a fim de guiar a trajetória, como o enviesamento na perturbação, por exemplo, regiões do espaço de busca são completamente desconsideradas para que outras sejam priorizadas. Entretanto, embora as regiões despriorizadas sejam, a princípio, pouco promissoras segundo o contexto do problema, ainda é possível que nelas estejam contidos mínimos locais melhores do que o conhecido. E estes mínimos seriam encontrados por um método menos específico, como a primeira versão do método proposto neste trabalho, por exemplo.

Diante do exposto, foi constatada a superioridade da primeira versão do método, que está disponível *online* no mesmo repositório anterior, sob a *tag* "v1.0"[2], junto dos resultados a serem utilizados nas comparações subsequentes e de um validador de soluções desenvolvido pelo aluno Gabriel Carvalho.

[2] <https://github.com/pedroldm/ic_ssp_npm/releases/tag/v1.0>

5.2 Comparação com o estado da arte

As Tabelas 5.2 e 5.3 apresentam as comparações entre as médias dos melhores resultados reportados pela ILS para cada conjunto de instâncias e os valores correspondentes ao estado da arte. Valores em negrito destacam os melhores resultados encontrados para o subconjunto de instâncias correspondente. O *gap(%)* foi calculado seguindo a fórmula $100 \times \frac{\text{ILS-BRKGA}}{BRKGA}$, sendo BRKGA os resultados do estado da arte (Soares; Carvalho, 2024). Os valores reportados foram arredondados de forma natural. Nos subconjuntos do SSP-NPM-I, contendo 20 instâncias cada, a ILS reportou resultados equivalentes em praticamente todos os sobconjuntos de instâncias.

Tabela 5.2 – Resultados obtidos para o conjunto de instâncias SSP-NPM-I.

M	T	F	$\mathbf{FMAX_{BRKGA}}$	$\mathbf{FMAX_{ILS}}$	$\mathbf{gap}(\%)$	$\mathbf{TFT_{BRKGA}}$	$\mathbf{TFT_{ILS}}$	$\mathbf{gap}(\%)$
2	10	10	29	29	0,00	129	**128**	-0,77
2	10	15	37	37	0,00	163	163	0,00
2	15	10	43	**42**	-2,32	272	**268**	-1,47
2	15	15	52	52	0,00	338	338	0,00
3	15	15	25	25	0,00	160	160	0,00
3	15	20	31	31	0,00	191	191	0,00
3	20	15	32	32	0,00	273	273	0,00
3	20	20	42	42	0,00	352	352	0,00

Para os subconjuntos do SSP-NPM-II, que contém 80 instâncias cada, a ILS obteve melhores resultados em vários sunconjuntos , apresentando uma média de *gap(%)* de -6,46 e -5,92% nos objetivos de minimização do *makespan* e *flowtime*, respectivamente.

Tabela 5.3 – Resultados obtidos para o conjunto de instâncias SSP-NPM-II.

M	T	F	$\mathbf{FMAX_{BRKGA}}$	$\mathbf{FMAX_{ILS}}$	$\mathbf{gap}(\%)$	$\mathbf{TFT_{BRKGA}}$	$\mathbf{TFT_{ILS}}$	$\mathbf{gap}(\%)$
4	40	60	304	**276**	-9,21	4547	**4362**	-4,06
4	40	120	608	**525**	-13,65	9221	**8625**	-6,46
4	60	60	446	**425**	-4,70	10532	**9997**	-5,07
4	60	120	895	**810**	-9,49	21190	**19978**	-5,71
6	80	120	862	**802**	-6,96	27656	**25337**	-8,38
6	120	120	**1320**	1377	4,13	67517	**63544**	-5,88

Comparando os melhores resultados de cada instância individualmente, para o objetivo de minimização do *makespan*, a ILS apresentou resultados iguais aos do BRKGA para 209 instâncias, resultados melhores para 272 – com maior diferença de 864 unidades de tempo – e piores para 159 instâncias – com maior diferença de 2188 unidades de tempo –. Já para o objetivo de minimização do *flowtime*, a ILS equiparou os resultados reportados pelo BRKGA em 154 instâncias, apresentou resultados melhores em 427 – com maior diferença de 21109 unidades de tempo – e resultados piores em 59 instâncias – com maior diferença de 28402 unidades de tempo.

Análises estatísticas foram realizadas para comparar o método proposto com o método de Soares e Carvalho (2024). O teste de normalidade Shapiro-Wilk(Shapiro; Wilk, 1965), com intervalo de confiança de 95%, rejeitou a hipótese nula de que os resultados comparados, reportados pela ILS desenvolvida neste trabalho $W = (0,722; 0,596)$ e $p - value = (3,18^{-31}; 4,99^{-36})$ e pelo método de Soares e Carvalho (2024) $W = (0,755; 0,586)$ e $p - value = (1,09^{-29}; 2,4^{-36})$,

poderiam ser modelados de acordo com uma distribuição normal. Consequentemente, foi aplicado o teste não paramétrico de Wilcoxon (Woolson, 2007), com nível de significância de 0,05, que resultou em $V = (19069; 14441)$ e $p = (2,32^{-26}; 2,911^{-47})$ para os objetivos de *makespan* e *flowtime*, respectivamente. Os valores de p indicam que há discrepância estatística significativa entre os resultados dos objetivos considerados, o que reforça a robustez do método desenvolvido neste trabalho.

A Tabela 5.4 apresenta dados relativos à média geral (μ), ao desvio padrão (σ) e ao tempo de execução (T), expresso em segundos, obtidos através da ILS, referentes a cada conjunto de instâncias. A ILS apresentou um elevado desvio padrão em relação ao grupo de maiores instâncias (6 máquinas, 120 tarefas e 120 ferramentas). Essa variação ocorre devido à dificuldade do método em alcançar a convergência dentro do limite de tempo estabelecido para a execução, de 3.600 segundos.

Tabela 5.4 – Média geral, desvio padrão e tempo de execução reportados pela ILS.

M	T	F	**FMAX**	μ	σ	**T**	**TFT**	μ	σ	**T**
2	10	10	29	29	0,02	0,6	128	130	0,01	0,6
2	10	15	37	37	0,02	0,7	163	164	0,00	0,7
2	15	10	42	43	0,02	2,0	268	272	0,01	2,3
2	15	15	52	53	0,01	2,3	338	340	0,01	2,7
3	15	15	25	26	0,01	1,6	160	160	0,00	2,6
3	15	20	31	32	0,01	2,0	191	192	0,00	3,0
3	20	15	32	33	0,01	3,1	273	273	0,00	4,9
3	20	20	42	42	0,01	4,6	352	353	0,01	8,7
4	40	60	276	288	0,03	61,0	4362	4478	0,02	244,1
4	40	120	525	549	0,03	68,6	8625	8819	0,02	269,2
4	60	60	425	439	0,02	261,8	9997	10225	0,01	3092,5
4	60	120	810	843	0,03	271,7	19978	20415	0,01	2966,8
6	80	120	802	837	0,05	1548,4	25337	26073	0,03	3610,9
6	120	120	1377	2255	0,29	3652,7	63544	92620	0,25	3723,9

Diante do exposto, é possível concluir que o método proposto foi capaz de produzir boas soluções com baixo desvio padrão para a maioria dos conjuntos de instâncias. Ademais, o tempo limite foi atingido apenas nos dois maiores grupos de instâncias, o que indica a capacidade do método de alcançar bons resultados em tempo razoável.

6 Conclusão

O *job sequencing and tool switching problem with non-identical parallel machines* é um problema $\mathcal{NP}$-Difícil com ampla aplicação no contexto das indústrias de manufatura. Neste trabalho, foram propostos duas definições da metaheurística ILS aplicadas ao problema. A primeira versão, com componentes mais generalistas, reportou bons resultados, mas com altos desvios padrões para os conjuntos de instâncias maiores. Já a segunda versão, com componentes mais específicos para uma trajetória um pouco mais orientada, apresentou resultados inferiores.

Esse fato é explicado pelas propriedades dos problemas combinatórios, nos quais o espaço de busca é, muitas das vezes, imprevisível. Ao orientar a trajetória, desprezamos regiões que julgamos não promissoras com base nas propriedades do problema em questão. Entretanto, é possível que nessas regiões estejam contidos mínimos locais – ou até mesmo globais – que poderiam ser encontrados sem a trajetória guiada.

Experimentos intensivos foram realizados utilizando a primeira versão do método desenvolvido e considerando o único conjunto de 640 instâncias da literatura, divididos em 14 subconjuntos. Resultados melhores ou equivalentes foram obtidos para quase todos os subconjuntos; entretanto, o método apresentou alto desvio padrão em relação ao maior conjunto de instâncias, com 6 máquinas e 120 tarefas a serem executadas. Isso se deve à limitação do tempo de execução de apenas 1 hora, que dificulta a convergência do método em instâncias maiores.

Referências

AMAYA, J. E.; COTTA, C.; FERNÁNDEZ, A. J. A memetic algorithm for the tool switching problem. In: SPRINGER. **Hybrid Metaheuristics: 5th International Workshop, HM 2008, Málaga, Spain, October 8-9, 2008. Proceedings 5**. Málaga, Spain, 2008. p. 190–202.

AMAYA, J. E.; COTTA, C.; FERNÁNDEZ-LEIVA, A. J. Memetic cooperative models for the tool switching problem. **Memetic Computing**, Springer, v. 3, p. 199–216, 2011.

AMAYA, J. E.; COTTA, C.; FERNANDEZ-LEIVA, A. J. Solving the tool switching problem with memetic algorithms. **AI EDAM**, Cambridge University Press, v. 26, n. 2, p. 221–235, 2012.

AMAYA, J. E.; COTTA, C.; FERNÁNDEZ-LEIVA, A. J. Cross entropy-based memetic algorithms: An application study over the tool switching problem. **International Journal of Computational Intelligence Systems**, Taylor & Francis, v. 6, n. 3, p. 559–584, 2013.

AMAYA, J. E.; COTTA, C.; LEIVA, A. J. F. Hybrid cooperation models for the tool switching problem. **Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)**, Springer, p. 39–52, 2010.

AMAYA, J. E.; COTTA, C.; LEIVA, A. J. F. A memetic cooperative optimization schema and its application to the tool switching problem. In: SPRINGER. **Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11**. Krakow, Poland, 2010. p. 445–454.

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

BEEZãO, A. C. *et al.* Scheduling identical parallel machines with tooling constraints. **European Journal of Operational Research**, v. 257, n. 3, p. 834–844, 2017. ISSN 0377-2217.

BURGER, A. P. *et al.* Scheduling multi-colour print jobs with sequence-dependent setup times. **Journal of Scheduling**, Springer, v. 18, p. 131–145, 2015.

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

CALMELS, D. An iterated local search procedure for the job sequencing and tool switching problem with non-identical parallel machines. **European Journal of Operational Research**, Elsevier, v. 297, n. 1, p. 66–85, 2022.

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

CRAMA, Y. *et al.* Minimizing the number of tool switches on a flexible machine. **International Journal of Flexible Manufacturing Systems**, v. 6, p. 33–54, 1994.

CURA, T. Hybridizing local searching with genetic algorithms for the job sequencing and tool switching problem with non-identical parallel machines. **Expert Systems with Applications**, v. 223, p. 119908, 2023. ISSN 0957-4174.

FATHI, Y.; BARNETTE, K. Heuristic procedures for the parallel machine problem with tool switches. **International Journal of Production Research**, Taylor & Francis, v. 40, n. 1, p. 151–164, 2002.

GÖKGÜR, B.; HNICH, B.; ÖZPEYNIRCI, S. Parallel machine scheduling with tool loading: a constraint programming approach. **International Journal of Production Research**, Taylor & Francis, v. 56, n. 16, p. 5541–5557, 2018.

GONÇALVES, J. F.; RESENDE, M. G. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, Springer, v. 17, n. 5, p. 487–525, 2011.

HAMON, J. *et al.* Feature selection in high dimensional regression problems for genomic. 06 2013.

HANSEN, P. *et al.* **Variable neighborhood search**. Boston, Massachusetts: Springer, 2019.

HOP, N. V. The tool-switching problem with magazine capacity and tool size constraints. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 35, n. 5, p. 617–628, 2005.

KHAN, B. K. *et al.* A generalized procedure for minimizing tool changeovers of two parallel and identical cnc machining centres. **Production Planning & Control**, Taylor & Francis, v. 11, n. 1, p. 62–72, 2000.

LÓPEZ-IBÁÑEZ, M. *et al.* The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Elsevier, v. 3, p. 43–58, 2016.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. **Iterated local search**. New York City, NY: Springer, 2003.

MECLER, J.; SUBRAMANIAN, A.; VIDAL, T. A simple and effective hybrid genetic search for the job sequencing and tool switching problem. **Computers & Operations Research**, v. 127, p. 105153, 2021. ISSN 0305-0548.

ÖZPEYNIRCI, S.; GÖKGÜR, B.; HNICH, B. Parallel machine scheduling with tool loading. **Applied Mathematical Modelling**, Elsevier, v. 40, n. 9-10, p. 5660–5671, 2016.

PAIVA, G. S.; CARVALHO, M. A. M. Improved heuristic algorithms for the job sequencing and tool switching problem. **Computers & Operations Research**, v. 88, p. 208–219, 2017. ISSN 0305-0548.

PANFILOVA, O.; OKREPILOV, V.; KUZMINA, S. Globalization impact on consumption and distribution in society. In: EDP SCIENCES. **Matec web of conference**. 17, Avenue du Hoggar, Parc d'Activités de Courtabœuf, B.P. 112, F-91944 Les Ulis Cedex A, France, 2018. v. 170, p. 01032.

SENNE, E. L. F.; YANASSE, H. H. Beam search algorithms for minimizing tool switches on a flexible manufacturing system. In: **XI WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering, MACMESE**. Baltimore, MD, USA: [s.n.], 2009. v. 9, n. 11, p. 68–72.

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

SHROUF, F.; ORDIERES, J.; MIRAGLIOTTA, G. Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In: IEEE. **2014 IEEE international conference on industrial engineering and engineering management**. Selangor Darul Ehsan, Malaysia, 2014. p. 697–701.

SOARES, L. C.; CARVALHO, M. A. Application of a hybrid evolutionary algorithm to resource-constrained parallel machine scheduling with setup times. **Computers & Operations Research**, v. 139, p. 105637, 2022. ISSN 0305-0548.

SOARES, L. C.; CARVALHO, M. A. Biased random-key genetic algorithm for the job sequencing and tool switching problem with non-identical parallel machines. **Computers & Operations Research**, v. 163, p. 106509, 2024. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054823003738>.

SOARES, L. C. R.; CARVALHO, M. A. M. Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. **European Journal of Operational Research**, v. 285, n. 3, p. 955–964, 2020. ISSN 0377-2217.

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

WOOLSON, R. F. Wilcoxon signed-rank test. **Wiley encyclopedia of clinical trials**, Wiley Online Library, p. 1–3, 2007.

WU, K. *et al.* Job scheduling of diffusion furnaces in semiconductor fabrication facilities. **European Journal of Operational Research**, v. 301, n. 1, p. 141–152, 2022. ISSN 0377-2217.

YADAV, A.; JAYSWAL, S. Modelling of flexible manufacturing system: a review. **International Journal of Production Research**, Taylor & Francis, v. 56, n. 7, p. 2464–2487, 2018.

YANASSE, H. H.; PINTO, M. J. The minimization of tool switches problem as a network flow problem with side constraints. **XXXIV SBPO–Simpósio Brasileiro de Pesquisa Operacional, realizado no Rio de Janeiro, RJ**, v. 8, p. 86, 2002.

ZAWADZKI, P.; ŻYWICKI, K. Smart product design and production control for effective mass customization in the industry 4.0 concept. **Management and production engineering review**, Production Engineering Committee of the Polish Academy of Sciences, Polish ..., 2016.