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

LORRAYNE CRISTINE FERREIRA SANTOS

REVENIMENTO PARALELO APLICADO AO PROBLEMA DE INDEXAÇÃO DE FERRAMENTAS

Ouro Preto, MG
2025

LORRAYNE CRISTINE FERREIRA SANTOS

REVENIMENTO PARALELO APLICADO AO PROBLEMA DE INDEXAÇÃO DE FERRAMENTAS

Monografia apresentada ao Curso de Ciência da Computação da 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
2025

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

FOLHA DE APROVAÇÃO

Lorrayne Cristine Ferreira Santos

Revenimento Paralelo Aplicado ao Problema de Indexação de Ferramentas

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 26 de Agosto de 2025.

Membros da banca

Marco Antonio Moreira de Carvalho (Orientador) - Doutor - Universidade Federal de Ouro Preto
Leonardo Cabral da Rocha Soares (Examinador) - Doutor - Instituto Federal do Sudeste de Minas Gerais - Campus Manhuaçu
André Luís Barroso Almeida (Examinador) - Doutor - Instituto Federal de Minas Gerais - Campus Ouro Preto

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 26/08/2025.

Documento assinado eletronicamente por **Marco Antonio Moreira de Carvalho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 27/08/2025, às 21:01, 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 **0965542** e o código CRC **0D43B3F6**.

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

Aos meus pais, que encontraram forças até em suas fraquezas para me apoiar, tornando-se o alicerce de cada conquista desta caminhada.

Agradecimentos

Primeiramente, agradeço ao professor Marco Antonio, cuja dedicação constante, paciência como educador e generosidade em transmitir saber fizeram toda a diferença. Um orientador extraordinário, cuja postura ética e compromisso acadêmico foram fonte de inspiração e referência durante esta jornada.

Agradeço também à República Cafofo, que foi meu lar em Ouro Preto e parte essencial desta caminhada. Entre dias fáceis e difíceis, encontrei ali uma família que levarei para toda a vida. A Leonardo Harmendani, agradeço pela compreensão nos momentos de incerteza e pelo incentivo contínuo que me fez avançar. E, por fim, a todos os amigos, que de diferentes formas contribuíram com palavras de apoio, companheirismo e leveza, tornando esta trajetória menos árdua e infinitamente mais significativa.

Resumo

A busca por maior eficiência em processos industriais impulsionou a necessidade de otimizar a alocação de ferramentas em máquinas de *controle numérico computadorizado* (do inglês, *computer numerical control*, CNC). Nessas máquinas, o *trocador automático de ferramentas* (do inglês, *automatic tool changer*, ATC) desempenha um papel essencial, sendo responsável pela troca automatizada das ferramentas na torreta, componente que armazena e organiza as ferramentas utilizadas no processo de usinagem. Essa torreta é composta por *slots*, posições individuais onde cada ferramenta é alocada. A disposição dessas ferramentas dentro da torreta impacta diretamente o desempenho do ATC, influenciando o tempo necessário para cada troca e, consequentemente, a eficiência operacional. Diante desse cenário, investiga-se o problema de indexação de ferramentas (ou *tool indexing problem*, TIP), cujo objetivo é definir a melhor configuração de alocação das ferramentas na torreta, buscando minimizar os deslocamentos necessários e reduzir o tempo total de operação. Trata-se de um problema combinatório complexo, normalmente tratado com métodos meta-heurísticos, dada a dificuldade de se obter soluções exatas em larga escala. Para lidar com essa complexidade, o revenimento paralelo (ou *parallel tempering*, PT) apresenta-se como uma abordagem promissora, fundamentada em técnicas de otimização inspiradas na física estatística. Ao explorar simultaneamente múltiplas soluções, o PT amplia a diversidade da busca, favorece a convergência e reduz a probabilidade de aprisionamento em mínimos locais. Nos experimentos computacionais realizados, o método proposto demonstrou desempenho consistente: em grande parte das instâncias, os resultados obtidos igualaram ou superaram os valores reportados na literatura, com diferenças mínimas nas poucas situações em que o melhor valor não foi atingido. Observou-se também elevada estabilidade entre execuções independentes e tempos de processamento reduzidos, mesmo em instâncias de maior complexidade, preservando a escalabilidade. Esses resultados reforçam o potencial do PT como alternativa robusta e eficiente para o TIP, conciliando qualidade de soluções geradas e baixo custo computacional.

Palavras-chave: *Parallel Tempering*, *Slots*, Torreta, Máquinas de Controle Numérico Computadorizado, Trocador Automático de Ferramenta.

Abstract

The pursuit of greater efficiency in industrial processes has driven the need to optimize tool allocation in computer numerical control (CNC) machines. In such machines, the automatic tool changer (ATC) plays a key role, being responsible for the automated exchange of tools within the turret, a component that stores and organizes the tools required for machining. This turret is composed of individual slots, where each tool is positioned. The arrangement of tools within the turret directly impacts ATC performance, influencing the time required for each change and, consequently, overall operational efficiency. Within this context, the Tool Indexing Problem (TIP) arises, aiming to define the best allocation of tools in the turret to minimize movements and reduce total operation time. As a combinatorial optimization problem, TIP is typically addressed with metaheuristic approaches, given the difficulty of obtaining exact solutions at larger scales. To deal with this complexity, parallel tempering (PT) emerges as a promising strategy, grounded in optimization techniques inspired by statistical physics. By simultaneously exploring multiple solutions, PT enhances search diversity, promotes convergence, and reduces the likelihood of entrapment in local minima. The computational experiments conducted demonstrate consistent performance: in most benchmark instances, the proposed method matched or outperformed the best-known results, with only minimal gaps when the optimum was not reached. Furthermore, the approach showed high stability across independent runs and reduced processing times, even for large-scale instances, preserving scalability. These findings highlight the potential of PT as a robust and efficient alternative for solving the TIP, successfully balancing solution quality with low computational cost.

Keywords: Parallel Tempering, Slots, Magazine, Computerized Numerical Control Machines, Automatic Tool Changer.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ACA	*ant colony algorithm*
ATC	*automatic tool changer*
BT	*busca tabu*
CNC	*computer numerical control*
FMS	*flexible manufacturing system*
GA	*genetic algorithm*
HS	*harmony search*
IPMTC	*identical parallel machines with tooling constraints*
MCMC	*Markov chain Monte Carlo*
MH	*Metropolis-Hastings*
MOOTIP	*multi-objective optimisation of tool indexing problem*
PT	*parallel tempering*
RCPMS	*resource-constrained parallel machine scheduling*
SSP	*job sequencing and tool switching problem*
TA	*threshold accepting*
TIP	*tool indexing problem*
VNS	*variable neighbourhood search*
WSA	*weighted superposition attraction*

Sumário

1 Introdução

A indústria moderna exige cada vez mais eficiência, flexibilidade e personalização, demandando níveis crescentes de adaptação e dinamismo nas operações industriais (Javaid et al., 2022). Nesse contexto, a evolução dos sistemas de produção tem sido fortemente impulsionada pela automação industrial e pela digitalização, redefinindo os padrões de eficiência e precisão no setor. Tecnologias como máquinas de *controle numérico computadorizado* (do inglês *computer numerical control*, CNC) destacam-se nesse cenário, permitindo a realização de operações complexas com maior exatidão e significativa redução de custos (Soori et al., 2024). Essas inovações têm promovido a criação de ambientes industriais mais adaptáveis, com novas possibilidades de integração e otimização no chão de fábrica (Oliveira et al., 2024; Monostori; Kadar; Bauernhansl, 2016).

Responsáveis por transformar instruções digitais em movimentos precisos, as máquinas CNC desempenham um papel central na condução de processos industriais com alto grau de controle e repetibilidade. Esses sistemas automatizados realizam uma série de *operações* com alta precisão, tais como: corte, modelagem, perfuração, entre outras, utilizando uma ampla gama de *ferramentas* especializadas. As ferramentas empregadas em máquinas CNC variam conforme sua função principal: fresas e serras circulares, por exemplo, são comumente utilizadas em centros de usinagem para cortes precisos; brocas desempenham um papel fundamental nos processos de perfuração; tornos fazem uso de insertos de metal duro, ferramentas de corte de ponta única e ferramentas de perfil para modelar superfícies cilíndricas, esféricas e cônicas (Yao et al., 2024).

A fabricação de uma peça ou produto geralmente segue uma *sequência de operações* bem definida, em que cada etapa utiliza ferramentas específicas para atender às exigências do projeto (Qudeiri et al., 2006). Essa coordenação precisa entre ferramentas e processos automatizados permite que as máquinas CNC realizem todas as operações com eficiência, eliminando limitações da usinagem manual, assegurando maior consistência nos resultados e contribuindo para a redução dos custos operacionais (Soori et al., 2024). Reconhecidas como um marco na automação industrial, essas máquinas e suas ferramentas tornaram-se pilares em diversos setores, oferecendo a flexibilidade necessária para atender a mercados em constante evolução e requisitos específicos de personalização.

Como exemplo do que foi mencionado, a Figura 1.1 ilustra uma peça usinada composta por 16 operações distintas realizadas em sequência, enumeradas de o_1 a o_{16}, contando com o uso de 10 ferramentas diferentes, especificadas de f_1 a f_{10}. Cada ferramenta é utilizada em operações específicas, e ferramentas distintas podem executar operações semelhantes. Por exemplo, as ferramentas f_6 e f_{10} são usadas para furações com broca helicoidal (o_{16} e o_{14}), enquanto as ferramentas f_2, f_3, f_4 e f_5 realizam o fresamento de canais (o_5, o_6, o_3 e o_4). Já as operações de

fresamento de degrau (o_2 e o_7) são realizadas pelas ferramentas f_1 e f_3, enquanto as operações de furação de centro (o_{11}, o_{13} e o_{15}) utilizam as ferramentas f_8 e f_9 .

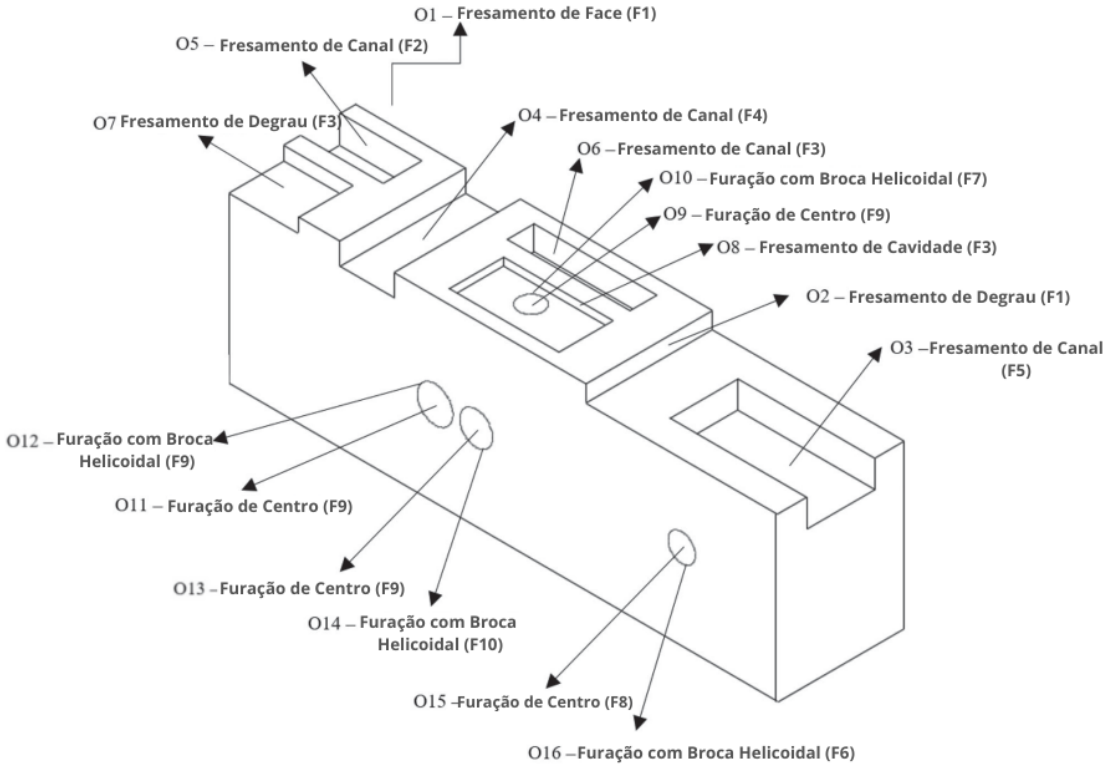

Figura 1.1 – Exemplo de componente e operações de produção. Adaptado de Baykasoglu, Yoruk e Yildiz (2024).

Essa flexibilidade é ainda mais evidente em sistemas de *manufatura flexível* (do inglês *flexible manufacturing systems*, FMS), uma abordagem na qual equipamentos e processos são projetados para se adaptar rapidamente a mudanças, seja nos tipos de produtos fabricados ou nos volumes de produção, oferecendo alta versatilidade e eficiência (Kaighobadi; Venkatesh, 1994). Nos FMS, a capacidade de realizar uma sequência completa de operações em uma única máquina CNC é significativamente aprimorada pelo uso do *trocador automático de ferramenta* (do inglês *automatic tool changer*, ATC). Esses dispositivos permitem que as máquinas alterem automaticamente as ferramentas durante o processo de usinagem, eliminando a necessidade de intervenção manual e, assim, otimizando o fluxo de trabalho e minimizando o tempo de inatividade (Muñoz-Benavent et al., 2019).

O ATC desempenha um papel essencial ao organizar as ferramentas em compartimentos individuais chamados *slots*, dispostos na *torreta*, uma estrutura rotativa integrada à máquina CNC. Esses *slots* permitem o posicionamento rápido e eficiente das ferramentas diretamente na posição de trabalho, otimizando o tempo de troca entre ferramentas e garantindo precisão durante as operações. Em máquinas em que o número de ferramentas necessárias para realizar uma determinada sequência de operações excede o limite da torreta, utiliza-se um *magazine*

adicional como sistema de armazenamento secundário, transferindo as ferramentas para a torreta conforme necessário (Duman, 2007). O *índice da ferramenta* representa a posição designada para cada ferramenta nos *slots* da torreta, desempenhando um papel crucial na troca e na execução eficiente de operações consecutivas (Baykasoğlu; Ozsoydan, 2016).

A Figura 1.2 apresenta uma torreta em que os índices das ferramentas estão representados sequencialmente de 1 a 10. As ferramentas, por sua vez, identificadas de F_1 a F_{10}, são alocadas de forma não ordenada nos *slots* e refletem as possíveis variações na distribuição dos tipos de ferramentas nos índices.

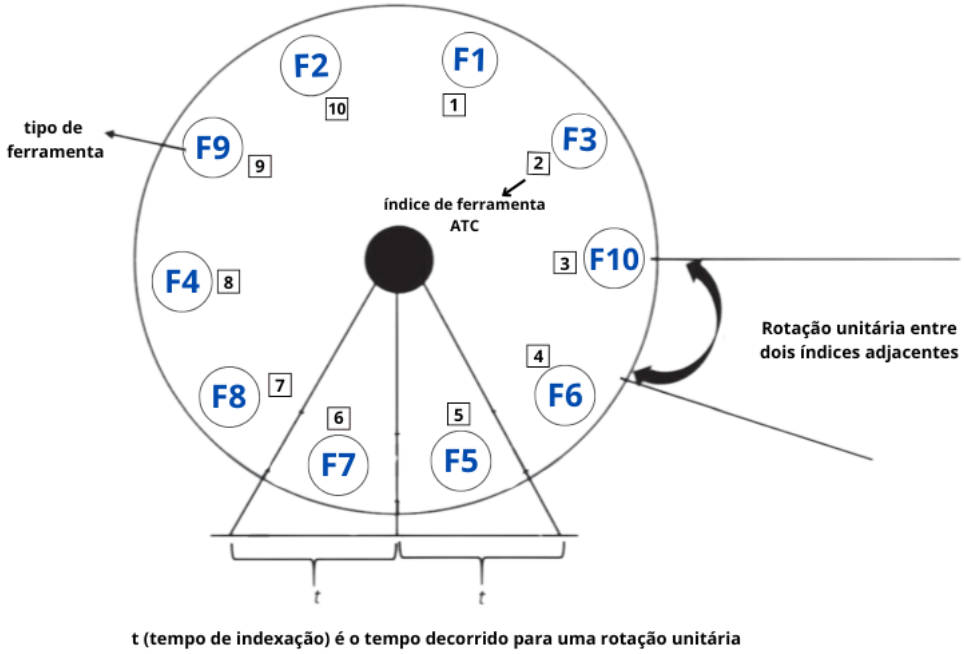

Figura 1.2 – Indexação de ferramentas em uma torreta (Dereli; Filiz, 2000).

Durante o processo de usinagem, a rotação da torreta pode ser realizada em ambas as direções, permitindo trazer a ferramenta correta para a posição de uso minimizando o movimento angular necessário. Essa transição, conhecida como *rotação unitária*, está representada na Figura 1.2, entre os índices de ferramenta 5 e 6, associados às ferramentas do tipo F_5 e F_7. O tempo requerido para essa alternância unitária de ferramentas, denominado *tempo de indexação da torreta* ou *tempo de indexação da ferramenta*, é ilustrado na referida Figura pela variável *t*, que indica cada rotação unitária. Ambos os processos são realizados automaticamente pelo ATC e resultam em um intervalo sem que nenhuma operação seja realizada sobre a peça, sendo este classificado como *tempo não produtivo* (Baykasoglu; Yoruk; Yildiz, 2024; Duman, 2007).

Neste contexto, dentre os processos mais relevantes envolvendo pelo ATC está o *problema de indexação de ferramentas* (do inglês *tool indexing problem*, TIP), introduzido por Dereli et al. (1998). Este problema combinatório está diretamente relacionado ao posicionamento eficiente das ferramentas nos *slots* da torreta e à busca pela minimização do tempo não produtivo.

O gerenciamento adequado das ferramentas nos *slots* da torreta não apenas minimiza o tempo de troca, mas também impacta diretamente o *tempo de ciclo* — o período total necessário para concluir todas as etapas da fabricação de uma peça, abrangendo operações, movimentações e substituições de ferramentas. A otimização desse ciclo resulta em maior produtividade e redução de ociosidade do maquinário, aspectos cruciais para o desempenho de sistemas de manufatura flexível (Baykasoğlu; Ozsoydan, 2016; Amouzgar; Nourmohammadi; Ng, 2021). Essa necessidade de maximizar a eficiência do arranjo de ferramentas consolida o TIP como um desafio central na busca por operações produtivas e otimizadas (Dereli et al., 1998).

Para abordar esse problema combinatório, esta monografia empregará uma implementação paralela da meta-heurística *revenimento paralelo* (do inglês *parallel tempering*, PT), uma metodologia projetada para otimizar e explorar o espaço de soluções em problemas como o TIP. Originalmente desenvolvido em contextos da química, física e biologia (Hukushima; Nemoto, 1996; Hansmann, 1997), o PT foi amplamente utilizado para simular sistemas termodinâmicos e analisar configurações complexas em materiais e moléculas. Inspirado no conceito de revenimento físico, o método utiliza buscas paralelas em diferentes intensidades para equilibrar a exploração e o refinamento, permitindo superar mínimos locais e aprimorar continuamente as soluções avaliadas. Posteriormente adaptado para a otimização, o PT tornou-se uma ferramenta poderosa, sendo reconhecido recentemente por sua eficiência em lidar com cenários complexos e encontrar soluções robustas (Earl; Deem, 2005a).

1.1 Justificativa

O TIP, longe de ser apenas um problema teórico, apresenta impactos mensuráveis em ambientes industriais. Sua relevância se evidencia no contexto de FMS e máquinas CNC, onde a otimização do tempo de indexação de ferramentas pode resultar em ganhos significativos em produtividade e redução de custos. Estudos mostram que a redução de tempos não produtivos pode melhorar a eficiência geral do sistema em mais de 40%, destacando sua relevância em ambientes industriais dinâmicos (Dereli; Filiz, 2000). Além disso, em cenários reais, soluções de otimização têm demonstrado uma redução de até 70% no tempo de indexação, o que aumenta diretamente a eficiência dos processos de usinagem (Amouzgar; Nourmohammadi; Ng, 2021).

O TIP é classificado como um problema NP-difícil, o que implica que não há algoritmos conhecidos que o resolvam em tempo determinístico polinomial para todos os casos (Dereli; Filiz, 2000). Essa complexidade intrínseca desafia pesquisadores a desenvolverem métodos heurísticos e meta-heurísticos eficazes, capazes de fornecer soluções aproximadas dentro de um tempo computacional viável. Nesse cenário, a computação paralela tem emergido como uma abordagem poderosa, ampliando significativamente a capacidade de exploração e intensificação em problemas de otimização combinatória. Recentemente, o PT foi implementado de forma inovadora por Almeida, de Castro Lima e Carvalho (2025), utilizando paralelismo para a criação

de uma API moderna, resultando em ganhos expressivos de desempenho.

A relevância prática do TIP, aliada ao potencial do algoritmo PT, possibilita uma abordagem eficiente para a minimização do tempo de indexação, ao mesmo tempo em que considera as adaptações exigidas pelo ambiente industrial. Essa combinação favorece o desenvolvimento de soluções robustas e alinhadas às demandas reais do setor, atendendo a múltiplos critérios de desempenho. Assim, a escolha do TIP como problema central, juntamente com a aplicação do PT como método de solução, justifica-se tanto pela sua aplicabilidade prática quanto pela sua capacidade de contribuir para o avanço científico e para a obtenção de resultados promissores na área.

1.2 Objetivos

Esta monografia tem como objetivo explorar o TIP em processos de usinagem, aplicando o PT como abordagem para sua modelagem e solução. A proposta visa proporcionar uma avaliação detalhada da qualidade da modelagem e dos resultados obtidos. Para isso, serão seguidas as seguintes etapas:

1. Condução de uma revisão bibliográfica abrangente para estabelecer uma base teórica sólida sobre o PT e o TIP;
2. Investigação da modelagem do TIP utilizando o PT, compreendendo seus princípios e aplicabilidade;
3. Desenvolvimento da modelagem e implementação da solução do TIP baseada no PT;
4. Avaliação crítica da abordagem implementada, comparando-a com métodos consagrados na literatura, utilizando instâncias de teste disponíveis.

A execução dessas etapas permitirão uma análise detalhada da eficácia e eficiência da metodologia proposta em relação a outras técnicas e embasarão as atividades restantes para a conclusão da monografia.

1.3 Organização do Trabalho

Este trabalho está estruturado da seguinte maneira: o Capítulo 2 apresenta uma revisão da literatura, abordando os principais estudos relacionados ao tema. No Capítulo 3, são descritos os fundamentos teóricos que sustentam o estudo em relação ao TIP e ao PT. O Capítulo 4 detalha o processo de desenvolvimento da abordagem proposta, enquanto o Capítulo 5 apresenta os experimentos computacionais realizados e a análise dos resultados obtidos. Por fim, no Capítulo 6 são expostas as conclusões deste trabalho.

2 Trabalhos relacionados

Este capítulo apresenta uma contextualização da pesquisa, oferecendo um panorama dos estudos já realizados por outros autores sobre o TIP. Apesar de sua relevância em otimização de processos produtivos, o TIP ainda é um problema relativamente recente na literatura e, portanto, há um número reduzido de trabalhos diretamente relacionados a ele. A revisão desses trabalhos busca situar a pesquisa e identificar lacunas que justifiquem a aplicação do PT à resolução do TIP.

O TIP foi introduzido por Dereli; Filiz em 1998 e teve sua definição aprimorada por Dereli e Filiz (2000). No cenário proposto, a sequência de operações é predefinida e não pode ser alterada, o objetivo é minimizar o tempo total de indexação das ferramentas e o custo do deslocamento é expresso em unidades de tempo. O problema foi classificado como NP-difícil, o que significa que não se conhece um método viável para resolvê-lo em geral, exigindo abordagens meta-heurísticas para encontrar soluções eficientes em tempo hábil.

O método utilizado para explorar diferentes configurações de ferramentas dentro da torreta foi o algoritmo genético (AG), que envolve operadores como *crossover* e mutação, além de uma função de aptidão baseada no tempo total de indexação para avaliar a qualidade das soluções geradas. O artigo também apresenta três cenários distintos para o TIP:

1. **Número de ferramentas igual ao número de *slots* disponíveis na torreta**: Configuração convencional em que todas as ferramentas necessárias podem ser alocadas sem a necessidade de trocas externas ou duplicações. Esse cenário representa a forma mais simples de alocação e é ilustrado na Figura 2.1.

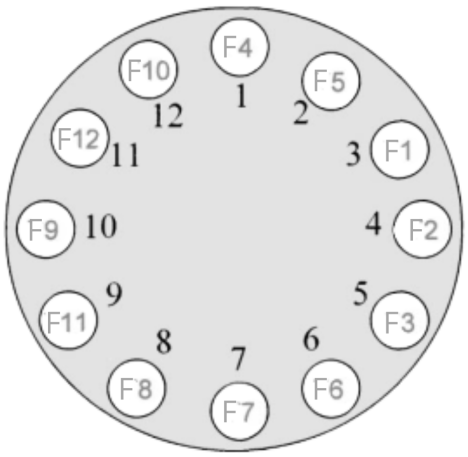

Figura 2.1 – Número de ferramentas e *slots* iguais. Adaptado de Dereli e Filiz (2000).

2. **Maior quantidade de *slots* disponíveis do que ferramentas necessárias**: Permite que ferramentas usadas com frequência sejam mantidas em mais de um *slot*, a duplicação ocorre para reduzir o tempo de movimentação da torreta. Figura 2.2.

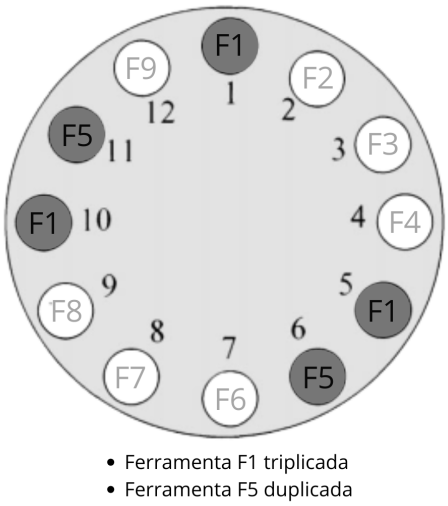

Figura 2.2 – Tipos de ferramenta com repetição. Adaptado de Dereli e Filiz (2000).

3. **Maior quantidade ferramentas do que *slots* disponíveis**: As ferramentas precisam ser armazenadas em *magazines* externos e recarregadas durante a operação, o que adiciona um tempo ao processo produtivo, considerado como fixo. Outro ponto relevante é a necessidade de um critério eficiente para saber qual ferramenta remover quando a torreta estiver cheia. Ilustrado na Figura 2.3.

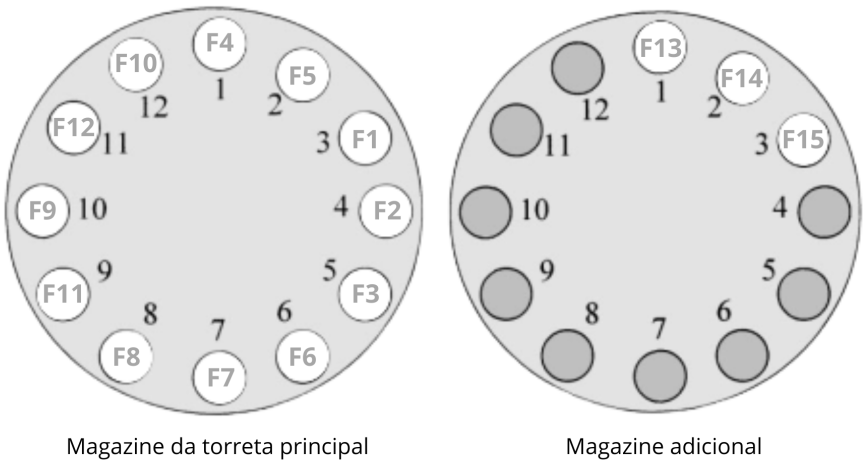

Figura 2.3 – Número de ferramentas superior aos *slots* disponíveis. Adaptado de Dereli e Filiz (2000).

A partir da definição dos diferentes cenários, observa-se que cada configuração demanda estratégias específicas para otimizar o tempo de indexação e reduzir os custos operacionais. Neste estudo, serão considerados especificamente o cenário 1 e o cenário 2, este último sem a duplicação de ferramentas.

O artigo de Baykasoglu e Dereli (2004) propôs uma abordagem baseada na meta-heurística *simulated annealing* (SA) para o TIP. Diferente do estudo anterior que utilizava AG (Dereli et al., 1998; Dereli; Filiz, 2000), os autores exploraram a capacidade do SA para reduzir o tempo total de indexação, levando em consideração a possibilidade de duplicação de ferramentas na torreta, cenário mostrado na Figura 2.2. Para uma das instâncias de teste, o AG levou 80 segundos para encontrar uma solução com 13 rotações (8,97 segundos de tempo total de indexação), enquanto o SA encontrou uma solução melhor em apenas 2,5 segundos, com 10 rotações (6,9 segundos de tempo de indexação). Além deste teste, os resultados gerais mostraram que, em comparação com o AG, o SA foi significativamente mais eficiente.

Krishna e Rao (2006) propuseram o uso de um algoritmo de colônia de formigas (*ant colony algorithm*, ACA) para resolver o problema de indexação de ferramentas. No estudo, foi considerado que o número de *slots* disponíveis na torreta era superior ao número de ferramentas necessárias, porém, não foi utilizada a duplicação de ferramentas mesmo com *slots* vazios. A abordagem foi comparada ao AG apresentado por Dereli e Filiz (2000), demonstrando maior eficiência na obtenção de soluções. Em um dos cenários analisados, o ACA encontrou a melhor configuração em 14 segundos, enquanto o AG necessitou de 26 segundos para alcançar o mesmo resultado.

Entre 2006 e 2015, houve um hiato na pesquisa sobre otimização de indexação de ferramentas em *slots* nas torretas de máquinas CNC. Durante esse período, a atenção da comunidade acadêmica e industrial foi direcionada a outros problemas de otimização e à evolução de novas metodologias em áreas distintas. No entanto, com o aumento das exigências industriais por soluções mais rápidas e eficientes, bem como a evolução da capacidade computacional, a pesquisa sobre a aplicação de meta-heurísticas ao TIP voltou a ganhar relevância.

O retorno das pesquisas sobre o TIP aconteceu com os estudos de Baykasoğlu e Ozsoydan (2015) e Baykasoğlu e Ozsoydan (2016), que consideraram a duplicação de ferramentas, conforme ilustrado na Figura 2.2. A abordagem proposta utiliza a técnica de *threshold accepting* (TA) e *variable neighbourhood search* (VNS) combinadas com um algoritmo de caminho mais curto. O TA, embora não seja uma meta-heurística propriamente dita, é um método de aceitação de soluções utilizado por meta-heurísticas para guiar a busca por soluções de menor custo. Quando comparada aos métodos baseados em SA (Baykasoglu; Dereli, 2004) e AG (Dereli; Filiz, 2000), a abordagem proposta obteve melhorias significativas. Em relação ao AG, a nova abordagem reduziu o tempo de indexação em aproximadamente 8,5%, passando de 138,12 segundos para 126,45 segundos. Já em um cenário mais complexo, a solução proposta alcançou um tempo de indexação de 251,78 segundos, enquanto o método baseado em SA registrou 267,92 segundos,

evidenciando a superioridade da nova abordagem.

Ghosh (2016) propôs a aplicação do algoritmo de busca tabu (ou *tabu search*, BT) ao TIP. O estudo considera o cenário com duplicação de ferramentas, conforme ilustrado na Figura 2.2. Algumas das instâncias avaliadas foram adaptadas do trabalho de Baykasoğlu e Ozsoydan (2015); no entanto, os resultados obtidos não foram diretamente comparados com outros métodos da literatura, sendo analisados unicamente com base na eficiência isolada da abordagem proposta.

Outro estudo de referência sobre o TIP foi realizado por Atta, Mahapatra e Mukhopadhyay (2018), em que foi proposta uma abordagem utilizando o algoritmo *harmony search* (HS). O artigo também trata da minimização das trocas externas, ou seja, situações em que ferramentas precisam ser removidas e substituídas manualmente do magazine externo para a torreta, o que adiciona um tempo fixo de atraso à operação, caso ilustrado na Figura 2.3. Em termos de resultados, o algoritmo HS demonstrou uma melhoria significativa, alcançando 16 novas melhores soluções entre as 27 instâncias avaliadas, resultando em uma redução de 10% no custo médio e uma diminuição de 15% no tempo total de indexação quando comparado ao método proposto por Ghosh (2016). Além disso, o algoritmo HS apresentou um tempo de convergência 30% mais rápido.

O artigo de Amouzgar, Nourmohammadi e Ng (2021) utilizou o algoritmo m-SPEA2 modificado para resolver o TIP de forma multi-objetivo (ou *multi-objective optimisation of tool indexing problem*, MOOTIP). O estudo propõe um modelo matemático inovador para otimizar o MOOTIP, considerando a duplicação de ferramentas e, pela primeira vez, considerando o desgaste das ferramentas e o tempo de vida útil das mesmas, fatores que não eram previamente considerados no TIP tradicional. O modelo proposto foi comparado com o método utilizado em uma indústria real. O estudo demonstrou uma redução significativa de 70% no tempo de indexação, que foi reduzido de 9 segundos para 2,8 segundos por peça.

O estudo conduzido por Baykasoglu, Yoruk e Yildiz (2024) propôs e avaliou diferentes abordagens para otimização do TIP. Inicialmente, os autores desenvolveram e testaram as formulações matemáticas baseadas em programação quadrática e programação quadrática linearizada, adaptando-as do problema de *layout* de instalações em corredor único. No entanto, ambas se mostraram ineficientes para instâncias médias e grandes, sendo viáveis apenas para instâncias pequenas. Como alternativa, foi desenvolvida uma nova formulação baseada em programação por restrições, que apresentou melhor desempenho ao fornecer soluções para instâncias médias, dentro dos limites computacionais estabelecidos.

Apesar do avanço com a programação com restrições, os testes demonstraram que métodos exatos ainda eram limitados para instâncias de grande porte. Assim, propôs-se, no mesmo estudo, o algoritmo *weighted superposition attraction* (WSA) como uma nova abordagem para o problema. Essa abordagem demonstrou sua capacidade de produzir as melhores soluções para todos os problemas de teste dentro de um tempo computacional razoável, superando, inclusive, alguns dos melhores resultados previamente registrados na literatura pelo HS, considerado até então

como estado da arte (Atta; Mahapatra; Mukhopadhyay, 2018).

Em termos de qualidade da solução, o WSA obteve o melhor valor absoluto em 14 instâncias e, em outras 20, apresentou desempenho equivalente ao de pelo menos um dos métodos comparados, o HS ou a programação com restrições e, não teve nenhuma instância na qual o WSA obteve um resultado pior. No entanto, ao analisar os tempos de execução, verifica-se que o WSA foi mais lento que o HS em todas as 34 instâncias avaliadas. Esses resultados destacaram a robustez do WSA na obtenção de soluções de alta qualidade, ainda que com um custo computacional relativamente maior. Este é o método considerado atual estado da arte para solução do TIP.

Com base na revisão dos trabalhos relacionados, é possível observar um avanço considerável nas abordagens aplicadas ao TIP, embora haja espaço para mais avanços na literatura restrita sobre o tema. As contribuições discutidas ao longo deste capítulo fornecem uma base sólida para o desenvolvimento de abordagens mais eficientes e flexíveis, como a aplicação do PT, que, considerando suas características, pode trazer benefícios consideráveis para solução do problema em questão.

3 Fundamentação teórica

Neste capítulo, são apresentados os fundamentos teóricos necessários para a compreensão deste estudo. Especificamente, a Seção 3.1 aborda o TIP, detalhando suas características e os desafios associados à sua resolução. Em seguida, na Seção 3.2, é descrito o método PT, utilizado neste trabalho como uma abordagem para solucionar o TIP.

3.1 Problema de indexação de ferramentas

O TIP aborda a alocação eficiente de ferramentas em uma torreta de máquinas CNC, visando minimizar o tempo não produtivo durante a execução de uma sequência fixa de operações. Neste trabalho, o TIP é considerado com as características e restrições descritas como se segue.

Tem-se uma lista de operações $O = [o_1, o_2, \ldots, o_n]$ e um conjunto de ferramentas $F = \{f_1, f_2, \ldots, f_m\}$. A sequência das operações O é fixa e predeterminada. Cada operação $o_i \in O$ requer ferramentas específicas que devem estar disponíveis na torreta no momento de sua execução. Uma torreta possui capacidade máxima C, que representa o número total de *slots* disponíveis para acomodar as ferramentas necessárias simultaneamente. Essa capacidade deve ser suficiente para eliminar substituições de ferramentas externas durante o processo. Cada ferramenta $f_j \in F$ deve ser alocada a um *slot* exclusivo e permanente na torreta, sendo permitido que algumas posições permaneçam vazias, desde que a capacidade C não seja excedida e não haja duplicações de ferramentas.

Para ilustrar os processos de usinagem, a Figura 1.1, introduzida anteriormente, apresenta um conjunto de operações, a ordem de execução das operações e as ferramentas requeridas em cada etapa. Com base nessa representação e respeitando as características previamente descritas, a Tabela 3.1 organiza de forma detalhada as informações das 16 operações. A tabela destaca três aspectos principais: a primeira linha identifica a ordem de execução das operações, numeradas de 1 a 16; a segunda linha identifica qual operação será executada, denominadas de o_1 a o_{16}; e a terceira linha relaciona as ferramentas utilizadas em cada operação, especificadas de f_1 a f_{10}. Essa estrutura proporciona uma visualização clara e organizada do processo produtivo, permitindo constatar que, ao longo das 16 operações, foram realizadas 11 movimentações para seleção de ferramentas na torreta.

Tabela 3.1 – Sequência de operações, números das operações e as ferramentas de corte utilizadas.

seq.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ope.	o_1	o_2	o_3	o_4	o_5	o_9	o_{10}	o_8	o_6	o_7	o_{11}	o_{12}	o_{13}	o_{14}	o_{15}	o_{16}
fer.	f_1	f_1	f_5	f_4	f_2	f_9	f_7	f_3	f_3	f_4	f_9	f_9	f_9	f_{10}	f_8	f_6

Como exemplo de uma possível solução dentro do contexto do TIP, anteriormente foi apresentada a Figura 1.2, contendo a configuração da alocação que origina a Tabela 3.2. A primeira linha da tabela representa os índices dos *slots* da torreta, numerados de 1 a 10, enquanto a segunda linha indica as ferramentas alocadas em cada *slot*.

Tabela 3.2 – Distribuição das ferramentas nos *slots* da torreta.

slot	1	2	3	4	5	6	7	8	9	10
ferramenta	f_1	f_3	f_{10}	f_6	f_5	f_7	f_8	f_4	f_9	f_2

Nessa configuração, as ferramentas foram distribuídas de forma aleatória nos 10 *slots* disponíveis na torreta, sem um planejamento específico para minimizar deslocamentos. Apesar disso, essa disposição permite avaliar os custos associados às trocas de ferramentas e a eficiência resultante dessa disposição. Nesta monografia, o custo de movimentação entre dois *slots* s_i e s_j é definido uniformemente como 1 unidade por troca, independentemente do tempo necessário para a movimentação. Para calcular a distância entre dois índices de *slots*, considera-se o menor deslocamento angular permitido pela rotação bidirecional da torreta. O custo individual de um deslocamento, denotado por $d_{i,j}$, é dado por:

$$d_{i,j} = \min\big(|j-i|,\ |i+C-j|\big), \tag{3.1}$$

em que:

- $d_{i,j}$ representa a menor distância circular entre as posições das ferramentas f_i e f_j na torreta;
- i é o índice do *slot* de origem (posição da ferramenta atual);
- j é o índice do *slot* de destino (posição da próxima ferramenta a ser utilizada);
- C é a capacidade total da torreta (número de *slots* disponíveis).

Dessa maneira, calcula-se o número mínimo de transições necessárias para mover uma ferramenta do *slot* i até o *slot* j, considerando que a torreta possui movimentação bidirecional (sentido horário e anti-horário). Por exemplo, ao calcular a distância entre os *slots* s_1 e s_7, o deslocamento no sentido horário é $|7-1| = 6$, enquanto no sentido anti-horário é $|1+10-7| = 4$. Assim, o custo mínimo de deslocamento é 4.

O custo total de indexação é obtido pela soma dos custos individuais de cada transição de operação, conforme definido pela Equação 3.2.

$$I_{\text{total}} = \sum_{i=1}^{n} I_i \tag{3.2}$$

Em que:

- I_{total} representa o custo total de indexação;
- d_i representa o custo de indexação associado à i-ésima transição entre ferramentas;
- n representa o número total de transições realizadas durante o processo.

Para facilitar a análise do posicionamento das ferramentas e o cálculo dos custos associados, a organização dos *slots* pode ser representada por uma matriz de distâncias, que resume os deslocamentos necessários entre todos os pares de ferramentas alocadas. Seguindo a configuração apresentada na Figura 1.2, foi elaborada a matriz de distâncias exibida na Tabela 3.3.

Tabela 3.3 – Matriz de distâncias entre ferramentas para uma torreta bidirecional.

	f_1	f_3	f_{10}	f_6	f_5	f_7	f_8	f_4	f_9	f_2
f_1	0	1	2	3	4	5	4	3	2	1
f_3	1	0	1	2	3	4	5	4	3	2
f_{10}	2	1	0	1	2	3	4	5	4	3
f_6	3	2	1	0	1	2	3	4	5	4
f_5	4	3	2	1	0	1	2	3	4	5
f_7	5	4	3	2	1	0	1	2	3	4
f_8	4	5	4	3	2	1	0	1	2	3
f_4	3	4	5	4	3	2	1	0	1	2
f_9	2	3	4	5	4	3	2	1	0	1
f_2	1	2	3	4	5	4	3	2	1	0

Nessa matriz, as ferramentas foram alocadas nos 10 *slots* disponíveis, organizadas na sequência $f_1, f_3, f_{10}, f_6, f_5, f_7, f_8, f_4, f_9, f_2$, conforme detalhado na Tabela 3.2. Cada célula da matriz indica o custo de movimentação entre dois *slots*, refletindo diretamente o custo de alternância entre as ferramentas posicionadas nesses *slots*. Essa representação facilita a avaliação da eficiência da disposição das ferramentas e a identificação de possíveis otimizações no arranjo.

Com base na matriz de distâncias, associada à sequência de operações e às ferramentas correspondentes, é possível calcular os custos relacionados ao tempo não produtivo do processo de usinagem. Durante as transições entre operações, o custo de indexação é atribuído apenas quando ocorre movimentação entre os *slots* da torreta. Caso a ferramenta utilizada em uma operação seja a mesma da operação subsequente, o custo de indexação é considerado nulo, pois não há deslocamento angular entre os *slots*.

A Tabela 3.4 detalha essas transições em quatro colunas principais. A primeira coluna indica a sequência numérica das transições realizadas, de 1 a 15, facilitando o acompanhamento do processo. A segunda coluna mostra quais são as operações consecutivas realizadas, representadas como $o_i \rightarrow o_j$. A terceira coluna utiliza o mesmo padrão para apresentar as ferramentas envolvidas em cada transição, indicando se houve troca ou se a mesma ferramenta foi reutilizada. Por fim, a quarta coluna apresenta o custo associado a cada transição de ferramentas, calculado com base no menor deslocamento angular necessário para realizar a troca.

Tabela 3.4 – Tabela de transições entre operações e ferramentas.

sequência	**operações**	**ferramentas**	**custo**
1	$o_1 \rightarrow o_2$	$f_1 \rightarrow f_1$	0
2	$o_2 \rightarrow o_3$	$f_1 \rightarrow f_5$	4
3	$o_3 \rightarrow o_4$	$f_5 \rightarrow f_4$	3
4	$o_4 \rightarrow o_5$	$f_4 \rightarrow f_2$	2
5	$o_5 \rightarrow o_9$	$f_2 \rightarrow f_9$	1
6	$o_9 \rightarrow o_{10}$	$f_9 \rightarrow f_7$	3
7	$o_{10} \rightarrow o_8$	$f_7 \rightarrow f_3$	4
8	$o_8 \rightarrow o_6$	$f_3 \rightarrow f_3$	0
9	$o_6 \rightarrow o_7$	$f_3 \rightarrow f_4$	4
10	$o_7 \rightarrow o_{11}$	$f_4 \rightarrow f_9$	1
11	$o_{11} \rightarrow o_{12}$	$f_9 \rightarrow f_9$	0
12	$o_{12} \rightarrow o_{13}$	$f_9 \rightarrow f_9$	0
13	$o_{13} \rightarrow o_{14}$	$f_9 \rightarrow f_{10}$	4
14	$o_{14} \rightarrow o_{15}$	$f_{10} \rightarrow f_8$	4
15	$o_{15} \rightarrow o_{16}$	$f_8 \rightarrow f_6$	3
custo total			**33**

Na primeira transição ($o_1 \rightarrow o_2$), a ferramenta f_1 é utilizada em ambas as operações, resultando em um custo nulo. Já na segunda transição ($o_2 \rightarrow o_3$), a ferramenta f_1 é substituída por f_5, incorrendo em um custo de 4 unidades. Esse padrão se repete ao longo da tabela, permitindo uma análise clara das transições e de seus impactos no custo total de indexação. Nesse exemplo, o custo total para a execução das 16 operações planejadas é de 33 unidades, conforme apresentado na última linha.

Para ilustrar como diferentes configurações afetam o custo de indexação, foi realizada uma redistribuição aleatória das ferramentas. Assim, uma nova Tabela 3.5 de distribuição das ferramentas foi gerada para refletir essa configuração específica dos *slots* da torreta.

Tabela 3.5 – Nova distribuição das ferramentas nos *slots* da torreta.

slot	1	2	3	4	5	6	7	8	9	10
ferramenta	f_2	f_3	f_1	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}

Após a redistribuição das ferramentas nos *slots*, foi gerada uma nova matriz de distâncias, representada na Tabela 3.6. Essa matriz reflete os custos de movimentação entre os *slots* com base na configuração atual.

Tabela 3.6 – Matriz de distâncias entre ferramentas para a segunda alocação.

	f_2	f_3	f_1	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
f_2	0	1	2	3	4	5	4	3	2	1
f_3	1	0	1	2	3	4	5	4	3	2
f_1	2	1	0	1	2	3	4	5	4	3
f_4	3	2	1	0	1	2	3	4	5	4
f_5	4	3	2	1	0	1	2	3	4	5
f_6	5	4	3	2	1	0	1	2	3	4
f_7	4	5	4	3	2	1	0	1	2	3
f_8	3	4	5	4	3	2	1	0	1	2
f_9	2	3	4	5	4	3	2	1	0	1
f_{10}	1	2	3	4	5	4	3	2	1	0

Para avaliar o impacto dessa configuração alternativa, foi elaborada a Tabela 3.7, que apresenta os custos associados às transições entre operações e ferramentas. Com essa nova disposição, o custo total de indexação foi reduzido para 27, evidenciando como uma reorganização eficiente pode melhorar o desempenho do sistema.

Tabela 3.7 – Tabela revisada de transições entre operações, ferramentas e custos associados ao tempo de indexação.

sequência	**operações**	**ferramentas**	**custo**
1	$o_1 \rightarrow o_2$	$f_1 \rightarrow f_1$	0
2	$o_2 \rightarrow o_3$	$f_1 \rightarrow f_5$	2
3	$o_3 \rightarrow o_4$	$f_5 \rightarrow f_4$	1
4	$o_4 \rightarrow o_5$	$f_4 \rightarrow f_2$	3
5	$o_5 \rightarrow o_9$	$f_2 \rightarrow f_9$	2
6	$o_9 \rightarrow o_{10}$	$f_9 \rightarrow f_7$	2
7	$o_{10} \rightarrow o_8$	$f_7 \rightarrow f_3$	5
8	$o_8 \rightarrow o_6$	$f_3 \rightarrow f_3$	0
9	$o_6 \rightarrow o_7$	$f_3 \rightarrow f_4$	2
10	$o_7 \rightarrow o_{11}$	$f_4 \rightarrow f_9$	5
11	$o_{11} \rightarrow o_{12}$	$f_9 \rightarrow f_9$	0
12	$o_{12} \rightarrow o_{13}$	$f_9 \rightarrow f_9$	0
13	$o_{13} \rightarrow o_{14}$	$f_9 \rightarrow f_{10}$	1
14	$o_{14} \rightarrow o_{15}$	$f_{10} \rightarrow f_8$	2
15	$o_{15} \rightarrow o_{16}$	$f_8 \rightarrow f_6$	2
custo total			**27**

A análise realizada evidencia que o custo total de indexação é altamente sensível à disposição das ferramentas na torreta. Arranjos mais eficientes minimizam o tempo não produtivo, destacando a importância do TIP como um problema de otimização no contexto da produção industrial. A busca pela melhor forma de alocar as ferramentas desempenha um papel fundamental no aumento da eficiência do sistema produtivo. Classificado como um problema *NP-difícil*, conforme destacado por Dereli e Filiz (2000), o TIP não possui algoritmos conhecidos capazes

de resolvê-lo em tempo determiístico polinomial para todos os casos. Diante dessa complexidade, técnicas como heurísticas e meta-heurísticas são amplamente utilizadas, permitindo alcançar soluções aproximadas de alta qualidade dentro de limites computacionais viáveis.

3.2 Revenimento paralelo

Durante a segunda guerra mundial, dentro do projeto Manhattan, cientistas enfrentaram dificuldades para modelar processos físicos complexos, como a propagação de nêutrons em materiais fissíveis. Diante desse desafio, o matemático Stanisław Ulam identificou que certos problemas de difícil solução analítica, poderiam ser resolvidos por meio de simulações estatísticas baseadas em amostragem aleatória. Inspirado por essa aleatoriedade, ele desenvolveu um método para aproximar soluções de problemas determinísticos utilizando processos estocásticos. Com a colaboração de John von Neumann, essa abordagem foi refinada e formalizada, originando o método de Monte Carlo, que se tornou uma ferramenta essencial para a resolução de problemas computacionalmente intensivos em diversas áreas (Benov, 2016).

O método de Monte Carlo consiste em realizar experimentos computacionais que geram amostras aleatórias para inferir propriedades de distribuições complexas ou resolver problemas matemáticos de difícil solução analítica. Embora essas técnicas tenham se mostrado extremamente úteis, especialmente em simulações físicas, elas apresentavam limitações em espaços de alta dimensionalidade, em que a eficiência da amostragem diminuía drasticamente. Para superar essas dificuldades, surgiu o método de Monte Carlo via cadeias de Markov (do inglês *Markov Chain Monte Carlo*, MCMC) , que representa uma evolução significativa nesse campo.

A ideia central do MCMC é construir uma cadeia de Markov, isto é, uma sequência de variáveis aleatórias em que a probabilidade de transição para o próximo *estado* — entendido como a configuração específica dos parâmetros ou variáveis do sistema — depende unicamente do estado atual, e não dos estados anteriores. Essa propriedade, conhecida como *memória curta* ou *propriedade markoviana*, permite modelar a dinâmica de sistemas estocásticos de forma simplificada.

Nesse contexto, a cadeia MCMC é projetada de modo que sua *distribuição estacionária*, isto é, a distribuição de probabilidade à qual a cadeia converge após um grande número de iterações, coincida com a *distribuição de interesse*, que representa a distribuição alvo da qual se deseja obter amostras, geralmente associada à solução de um problema estatístico ou à estimação de parâmetros (Jones; Qin, 2022). Dessa forma, é possível amostrar eficientemente distribuições complexas, mesmo em espaços de alta dimensionalidade.

Um exemplo clássico desse processo é a cadeia de Markov de múltiplos estados, que modela a transição entre três ou mais estados distintos. Como ilustrado na Figura 3.1, cada estado é representado por uma variável discreta, e as transições entre estados são determinadas por probabilidades condicionais. A transição do estado x_t para x_{t+1} depende exclusivamente do

estado atual, conforme a propriedade de memória curta. Cada estado i pode evoluir para qualquer outro estado j com uma certa probabilidade p_{ij}. Uma cadeia de Markov com três estados, como no exemplo, possui $3^2 = 9$ probabilidades de transição distintas, formando um *kernel de transição*, que podem ser estimadas com base em modelos probabilísticos (Wilks, 2019).

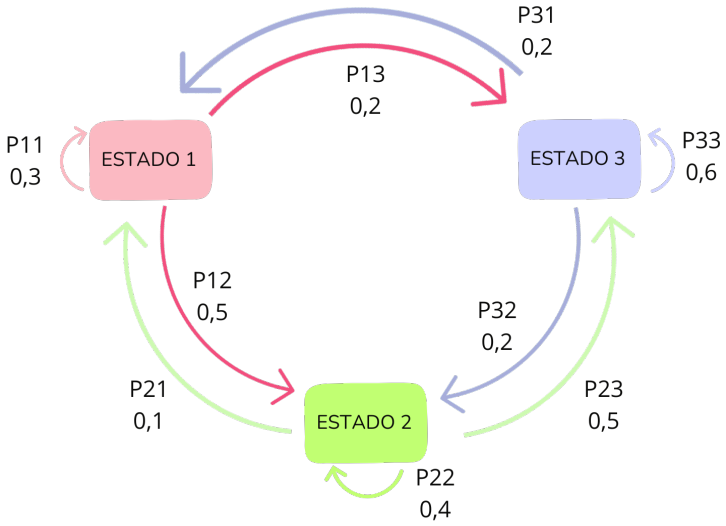

Figura 3.1 – Cadeia de Markov de múltiplos estados, em que as setas indicam as probabilidades de transição entre os estados.

Um dos marcos iniciais no surgimento dos métodos MCMC foi a introdução do *algoritmo de Metropolis* (Metropolis et al., 1953). Esse algoritmo permitiu, pela primeira vez, gerar amostras a partir de uma distribuição de interesse mesmo quando sua forma explícita não era conhecida. A estratégia baseia-se na construção de uma cadeia de Markov, conforme discutido anteriormente, em que cada novo estado candidato é aceito ou rejeitado de acordo com uma função associada à configuração do sistema. Esse mecanismo garante que, ao longo do tempo, a cadeia convergirá para a distribuição desejada.

A generalização do algoritmo de Metropolis foi proposta por Hastings em 1970, originando o chamado algoritmo de *Metropolis-Hastings* (MH). Essa versão ampliou a aplicabilidade do método, permitindo que as transições entre estados fossem realizadas de acordo com uma distribuição proposta, flexibilizando o critério de aceitação. Para assegurar que as transições entre estados respeitem a distribuição de equilíbrio do sistema, utiliza-se com maior frequência o critério de Boltzmann, que se consolidou como o mais empregado na prática, ainda que não seja a única alternativa existente. A Equação 3.3 apresenta sua formulação.

$$P(E,T) = e^{-\frac{(\Delta E)}{T}} \tag{3.3}$$

Em que:

- e é a base dos logaritmos naturais;

- $\Delta E = f(s') - f(s)$ são as energias dos estados s e s';
- T é o parâmetro de temperatura.

No contexto de métodos de amostragem estatística e otimização combinatória, os conceitos de energia, temperatura e estado têm interpretações específicas. A energia é interpretada como o custo ou a qualidade de uma solução no espaço de busca. Em problemas de otimização, como o TIP, a energia pode corresponder ao tempo total de indexação ou ao número de trocas de ferramentas, dependendo da métrica que se deseja otimizar. Assim, encontrar estados de menor energia equivale a identificar soluções de menor custo.

Além disso, o estado do sistema representa uma configuração específica das variáveis de decisão do problema, como a disposição das ferramentas em um magazine no TIP. O algoritmo transita entre esses estados utilizando movimentos de vizinhança, que são pequenas modificações na configuração atual. A escolha estratégica dessas vizinhanças é crucial para garantir uma exploração eficiente do espaço de busca.

A temperatura, por sua vez, representa um parâmetro de controle que define a probabilidade de aceitar soluções ótimas ou subótimas. Em temperaturas altas, o algoritmo permite a aceitação de soluções de maior custo com maior probabilidade, o que ajuda a explorar o espaço de busca e a evitar o aprisionamento em mínimos locais. Isso ocorre porque, em temperaturas altas, o fator exponencial do critério de Boltzmann suaviza as diferenças de energia, tornando pequenas variações irrelevantes. Conforme a temperatura diminui, a aceitação de soluções piores se torna menos provável, o que intensifica a busca em torno dos mínimos locais, promovendo a convergência para soluções de alta qualidade. Esse mecanismo de controle permite um balanceamento dinâmico entre exploração global e intensificação local (Earl; Deem, 2005b).

O Algoritmo 3.1 descreve o funcionamento do algoritmo de Metropolis, que recebe como entrada uma temperatura T, utilizada para determinar a aceitação das soluções, e um inteiro $N_{\max}$, que define o comprimento da cadeia de Markov. O processo inicia com a definição da solução inicial s_0 (linha 2). A variável de controle k é inicializada com zero (linha 3) e representará o número de soluções vizinhas geradas. Em seguida, a cadeia de Markov é construída iterativamente até atingir seu comprimento final (linha 4).

Para cada iteração, uma nova solução s' é gerada a partir da solução atual s_k (linha 5). Posteriormente, calcula-se a variação de energia ΔE, correspondente à diferença entre o custo da nova solução e da solução atual (linha 6). O critério de aceitação da nova solução é verificado na linha 7. Se a nova solução s' apresenta um custo menor do que s_k, esta é imediatamente aceita. Caso contrário, a nova solução só será aceita com probabilidade definida pelo critério de Boltzmann e um número aleatório $U(0, 1)$, que representa um valor extraído de uma distribuição uniforme entre 0 e 1. Se o número do critério de Boltzmann for maior do que esse número aleatório, a nova solução s' é aceita (linha 8). Caso contrário, s' é rejeitada, e a solução atual s_k permanece inalterada (linha 10). Após cada decisão, a variável de controle k é incrementada

(linha 11), e o processo se repete até que o critério de parada seja atingido. Finalmente, ao término do laço, o algoritmo retorna a última solução aceita s_k (linha 12).

Algoritmo 3.1: Algoritmo de Metropolis

Entrada: Temperatura T e comprimento máximo da cadeia de Markov $N_{\max}$

Saída: Solução s_k

1 Inicializar s_0 com uma solução inicial;
2 $s_0 \leftarrow$ solução inicial;
3 $k \leftarrow 0$;
4 **enquanto** $k \leq N_{\max}$ **faça**
5 Gerar uma nova solução s' a partir de s_k;
6 Calcular $\Delta E = f(s') - f(s_k)$;
7 **se** $\Delta E \leq 0$ ***ou*** $e^{-\frac{(\Delta E)}{T}} > U(0,1)$ **então**
8 $s_{k+1} \leftarrow s'$;
9 **senão**
10 $s_{k+1} \leftarrow s_k$;
11 $k \leftarrow k + 1$;
12 **retorna** s_k;

Apesar da eficácia do algoritmo de Metropolis em muitas aplicações, sua performance pode ser limitada ao lidar com distribuições *multimodais* — aquelas que apresentam dois ou mais picos de alta probabilidade separados por regiões de baixa densidade. Nesses cenários, a cadeia de Markov pode permanecer presa em um modo local sem conseguir explorar outras regiões do espaço de estados. Isso ocorre porque a aceitação de novos estados depende diretamente da diferença de energia entre configurações, dificultando a transição entre mínimos locais quando a temperatura é baixa.

Para mitigar essa limitação, um dos métodos introduzidos foi o PT, também conhecido como *Replica Exchange Monte Carlo*, inicialmente proposto por Swendsen; Wang em 1986. O PT foi projetado para melhorar a eficiência de simulações de Monte Carlo em sistemas físicos complexos, tornando-se uma abordagem essencial para problemas que envolvem barreiras de energia elevadas e regiões irregulares de soluções. A ideia central do PT é executar múltiplas réplicas do sistema, ou cadeias de Markov, em paralelo, cada uma operando em uma temperatura distinta.

No PT, cada réplica do sistema é associada a uma temperatura específica. Geralmente, as temperaturas são dispostas de maneira crescente, formando uma sequência$T_1 < T_2 < \cdots < T_\kappa$. As réplicas em temperaturas mais altas têm maior probabilidade de aceitar transições para estados de energia maior, o que permite que elas se desloquem mais livremente pelo espaço de busca. Em contraste, réplicas em temperaturas mais baixas têm menor tolerância a aumentos de energia, concentrando-se na exploração detalhada de mínimos locais. Deste modo, a utilização

de temperaturas variadas permite que algumas réplicas explorem o espaço de busca de maneira mais ampla, enquanto outras se concentram em refinar soluções em regiões de menor energia.

A característica que torna o PT uma extensão poderosa do MCMC é a possibilidade de realizar trocas periódicas de temperatura entre réplicas adjacentes. Isso significa que uma réplica que está explorando amplamente em uma temperatura elevada pode, eventualmente, trocar de temperatura com outra réplica operando em uma temperatura mais baixa. Esse mecanismo possibilita que boas soluções encontradas em temperaturas altas sejam refinadas em temperaturas mais baixas, e que soluções presas em mínimos locais possam escapar para regiões de energia mais elevada antes de retornarem à exploração mais detalhada.

A Equação 3.4 define a probabilidade relativa de aceitar uma troca de temperaturas entre duas réplicas i e j em temperaturas adjacentes, com base na diferença de *energia* e na diferença de *temperatura* entre as réplicas envolvidas.

$$P_{\text{troca}} = \min\left(1, \exp\left[(\beta_i - \beta_j)(E_i - E_j)\right]\right) \tag{3.4}$$

Em que:

- $\exp$ representa a função exponencial, equivalente a e^x, em que e é a base dos logaritmos naturais;
- $\beta_i = \frac{1}{T_i}$ e $\beta_j = \frac{1}{T_j}$ são os inversos das temperaturas associadas às réplicas i e j;
- E_i e E_j são as energias dos estados das respectivas réplicas.

Essa formulação assegura que:

- Se a troca resulta em uma diminuição global de energia ($E_j < E_i$), então o critério de Boltzmann gera um valor maior que 1. Nesse caso, a função $\min(1, \cdot)$ define $P_{\text{troca}} = 1$, e a troca é aceita com certeza, promovendo a convergência para estados de menor energia;
- Se a troca resulta em um aumento de energia ($E_j > E_i$), então o critério de Boltzmann gera um valor entre 0 e 1, e P_{troca} será igual a esse valor. A troca, portanto, será aceita com probabilidade proporcional à diferença de energia entre os estados.

A Figura 3.2 ilustra a dinâmica das trocas de réplicas entre quatro temperaturas distintas (T_0, T_1, T_2, T_3). Cada nível de temperatura representa uma réplica do sistema, em que configurações são geradas e testadas em diferentes regiões do espaço de estados. As réplicas em temperaturas mais altas (T_3 e T_2) exploram de maneira mais ampla o espaço de busca, enquanto réplicas em temperaturas mais baixas (T_1 e T_0) concentram-se no refinamento de soluções em torno dos mínimos locais. O fluxo das transições entre réplicas adjacentes é representado pelas

setas, indicando tentativas de troca que podem ser aceitas ou rejeitadas conforme o critério de aceitação do método.

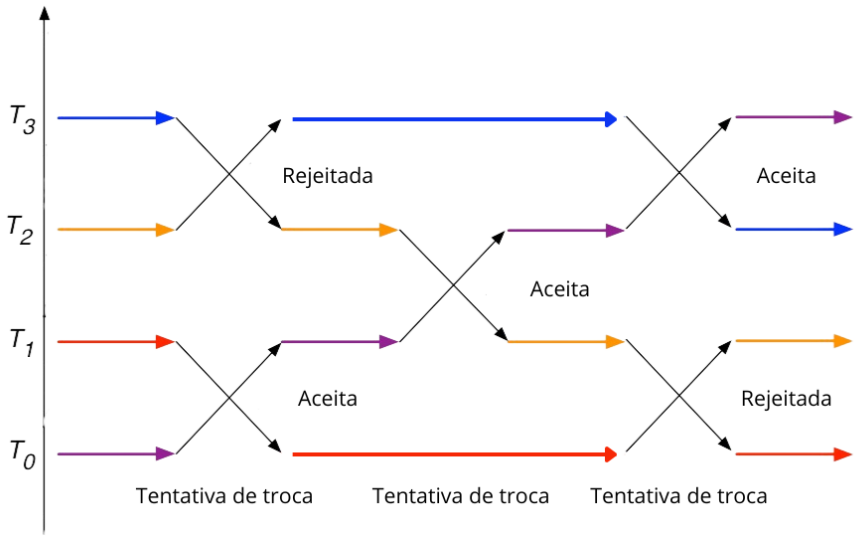

Figura 3.2 – Ilustração do processo de troca de soluções entre diferentes temperaturas no PT. Adaptado de Nagai et al. (2016).

No primeiro ciclo de tentativas, observa-se que a troca entre T_0 e T_1 foi aceita, entretanto, a troca entre T_2 e T_3 foi rejeitada. No segundo ciclo, novamente são realizadas tentativas de troca entre pares de réplicas adjacentes. Desta vez, foi aceita a troca entre T_1 e T_2, enquanto as outras temperaturas não realizaram tentativa de troca. No terceiro ciclo, foi realizada a troca entre T_2 e T_3 e rejeitada a troca entre T_1 e T_0. Esse comportamento reflete a natureza probabilística do critério de aceitação, que regula a transição entre estados com base na diferença de energia e temperatura das réplicas envolvidas.

Para ilustrar a relação básica destes componentes, o Algoritmo 3.2 apresenta o funcionamento do PT. O algoritmo recebe como entrada um vetor T, que contém as temperaturas das diferentes réplicas do sistema, um vetor S, em que cada posição $S[i]$ corresponde a uma solução associada à temperatura $T[i]$, e um inteiro N_{max}, que define o comprimento máximo da cadeia de Markov para cada réplica. Como saída, apresenta o vetor de soluções S atualizado.

O algoritmo inicia verificando se o critério de parada foi atingido (linha 1). Esse critério é definido previamente ao algoritmo. Dentro do laço principal, a variável *índice* é inicializada com zero (linha 2), representando a réplica inicial dentro do conjunto de temperaturas. Para cada réplica, a solução s (linha 4) e a temperatura correspondente t (linha 5) são obtidas a partir do vetor de soluções S e do vetor de temperaturas T, respectivamente. Na linha 6, a variável k é inicializada com zero para controlar o comprimento das cadeias de Markov geradas.

Na fase de exploração local, entre as linhas 7 e 12, segue-se o modelo definido pelo algoritmo de Metropolis (Algoritmo 3.1). Finalizada a fase de exploração local, a solução otimizada s

é armazenada de volta no vetor de soluções $S[\text{índice}]$ (linha 13). A seguir, é verificado se a réplica atual possui uma réplica anterior cuja cadeia de Markov já foi gerada (linha 14), pois trocas só podem ocorrer entre réplicas adjacentes. Se essa condição for verdadeira, é calculada a diferença entre os inversos das temperaturas $\Delta\beta$ (linha 15) e a variação de energia ΔE entre as soluções associadas às réplicas vizinhas (linha 16). A troca entre as réplicas é aceita automaticamente se $\Delta\beta \cdot \Delta E \geq 0$ (linha 17). Caso contrário, a troca ainda pode ser aceita probabilisticamente. Se a troca for aceita, os valores $S[\text{índice}]$ e $S[\text{índice} - 1]$ são trocados (linha 18). Após a verificação de trocas, a variável índice é incrementada (linha 19), garantindo que todas as réplicas sejam percorridas. O laço principal continua até que o critério de parada seja atingido, momento em que o algoritmo retorna o vetor de soluções S atualizado (linha 20).

Algoritmo 3.2: Algoritmo Revenimento Paralelo básico

Entrada: Vetor de temperaturas T, vetor de soluções S, comprimento máximo da cadeia de Markov $N_{\max}$.
Saída: Vetor de soluções S

1 **enquanto** *critério de parada não atingido* **faça**
2 índice $\leftarrow 0$; `// Inicializa o índice da réplica`
3 **enquanto** *índice* $<$ *tamanho*(T) **faça**
4 $s \leftarrow S[\text{índice}]$;
5 $t \leftarrow T[\text{índice}]$;
6 $k \leftarrow 0$;
7 **enquanto** $k \leq N_{max}$ **faça**
8 $s' \leftarrow$ Gerar_Vizinho(s) ;
9 $\Delta E \leftarrow f(s') - f(s)$;
10 **se** $\Delta E \leq 0$ ***ou*** $\exp(-\Delta E/k_B t) > U(0,1)$ **então**
11 $s \leftarrow s'$;
12 $k \leftarrow k + 1$;
13 $S[\text{índice}] \leftarrow s$;
14 **se** *índice* > 0 **então**
15 $\Delta\beta \leftarrow \frac{1}{k_B T[\text{índice}-1]} - \frac{1}{k_B T[\text{índice}]}$;
16 $\Delta E \leftarrow f(S[\text{índice} - 1]) - f(S[\text{índice}])$;
17 **se** $\Delta\beta \cdot \Delta E \geq 0$ ***ou*** $\exp(-\Delta\beta \cdot \Delta E) > U(0,1)$ **então**
18 Trocar$(S[\text{índice}], S[\text{índice} - 1])$;
19 índice $\leftarrow$ índice $+ 1$;
20 **retorna** S ; `// Retorna o vetor de soluções atualizado`

Implementações tradicionais do PT, apesar de eficientes na busca por soluções, não exploram o paralelismo de CPU, o que pode resultar em tempos de execução elevados para problemas de alta complexidade. Visando superar essa limitação, Almeida, de Castro Lima e Carvalho (2025) propuseram uma abordagem na qual o método de PT foi revisitado e uma API (*application programming interface*) específica foi desenvolvida para sua implementação otimizada e paralela. O estudo destaca a importância do PT na pesquisa operacional e propõe sua

adaptação para problemas computacionalmente intensivos, integrando conceitos de computação de alto desempenho e algoritmos de busca heurística.

Diferente das abordagens anteriores, essa implementação foi desenvolvida para eliminar a redundância das execuções sequenciais do PT, tornando-o mais eficiente e adequado para problemas de grande escala. Para isso, uma API foi projetada para explorar de forma otimizada o paralelismo em arquiteturas *multi-core*, permitindo a execução simultânea de múltiplas cadeias de Markov e aprimorando a troca de réplicas entre diferentes temperaturas. Além de reduzir o tempo computacional, essa implementação também facilita sua aplicação e adoção por pesquisadores e profissionais da área, promovendo uma utilização mais ampla do método.

Os experimentos conduzidos com a API desenvolvida por Almeida, de Castro Lima e Carvalho (2025) foram focados em três problemas de otimização amplamente estudados na literatura. Entre eles, destacam-se o problema de minimização de trocas de ferramentas (ou *job sequencing and tool switching problem*, SSP), o problema de escalonamento de tarefas em máquinas paralelas idênticas com restrições de ferramentas (ou *identical parallel machines with tooling constraints*, IPMTC) e o problema de sequenciamento de máquinas paralelas com limitações de recursos (ou *resource-constrained paralllel machine scheduling*, RCPMS).

Cada um desses problemas foi avaliado em comparação com métodos de referência na literatura, destacando desafios específicos relacionados à alocação de ferramentas, ao balanceamento de carga e à otimização do tempo de processamento. A avaliação desses cenários permitiu verificar a eficiência e a aplicabilidade da API proposta, demonstrando seu desempenho em diferentes contextos de otimização.

O SSP visa a redução da quantidade de trocas de ferramentas ao longo de um processo produtivo, minimizando o tempo ocioso e os custos operacionais. Em termos de eficiência computacional, os resultados demonstraram que o PT superou o método estado da arte reduzindo o tempo de execução em todos os doze grupos de instâncias consideradas, com reduções que chegaram a 92,98%. Em algumas situações, o método comparado chegou a demandar sete horas e trinta minutos para resolver uma instância específica, tornando-se inviável para aplicações industriais que exigem respostas rápidas. Em contraste, o PT foi capaz de resolver a mesma instância em apenas trinta minutos.

O segundo problema abordado foi o IPMTC, que consiste na alocação eficiente de tarefas em um conjunto de máquinas paralelas idênticas. A avaliação experimental demonstrou a eficácia do PT, com a obtenção de novas melhores soluções em aproximadamente 88,4% das instâncias consideradas, totalizando 1.274 melhores soluções entre as 1.440 instâncias avaliadas no conjunto.

Por fim, o RCPMS aborda a ordenação e alocação de tarefas em um ambiente de produção com restrições adicionais de recursos. Os resultados dos experimentos computacionais indicam que o PT superou o método estado da arte em termos de qualidade da solução, obtendo 144 novas melhores soluções em 180 instâncias consideradas.

A partir desses resultados, a API desenvolvida por Almeida, de Castro Lima e Carvalho (2025) será utilizada como base para a implementação do PT na resolução do TIP, viabilizando a aplicação eficiente do método sem a necessidade de reescrever a infraestrutura paralelizada. No entanto, a adoção da API não dispensa o desenvolvimento de estruturas de dados, implementações de componentes e ajustes de parâmetros, uma vez que será necessário adaptar o PT às especificidades do problema tratado.

Para garantir um desempenho eficiente do método, é essencial definir adequadamente os parâmetros, que impactam significativamente a qualidade das soluções geradas. Entre os parâmetros a serem definidos, tem-se:

- Os valores das temperaturas inicial e final das réplicas;
- O número total de réplicas utilizadas no algoritmo;
- O comprimento das cadeias de Markov associadas a cada réplica;
- O número de trocas entre temperaturas durante a execução do algoritmo;
- O espaçamento das temperaturas entre os valores mínimo e máximo.

Para adaptar e compatibilizar a API à formulação do TIP considerada neste estudo, é necessário desenvolver componentes específicos que permitam a integração eficiente do método ao problema. Embora o PT seja um algoritmo generalista aplicado a diferentes domínios, o TIP apresenta particularidades que exigem ajustes estruturais para garantir uma implementação eficaz. Assim, os seguintes componentes devem ser desenvolvidos:

- **Codificação/decodificação**: A codificação determina como uma solução do problema é estruturada e armazenada, enquanto a decodificação converte essa estrutura em um formato compreensível pelo modelo do problema;
- **Função de avaliação**: Refere-se à função utilizada para medir a qualidade de cada solução gerada;
- **Solução inicial**: Representa a distribuição inicial das ferramentas para cada réplica no algoritmo. Pode ser gerada aleatoriamente ou baseada em alguma estratégia predefinida;
- **Estruturas de vizinhança**: Definem como novas soluções são geradas a partir da solução atual. Determinam as regras de modificação, permitindo a exploração do espaço de busca;
- **Critério de parada**: Define a condição para encerrar a execução do algoritmo.

Uma vez definido formalmente o problema e aprofundado o conhecimento sobre o funcionamento do PT, torna-se possível delinear de maneira mais precisa a adaptação do método

para sua aplicação ao TIP. O próximo capítulo indica esta etapa e outras como parte das atividades restantes para a conclusão desta monografia.

4 Desenvolvimento

A construção de um método eficaz para o TIP exige a definição cuidadosa de componentes e estruturas. Este capítulo apresenta as estratégias desenvolvidas para gerar soluções iniciais, avaliar sua qualidade, explorar o espaço de busca por meio de vizinhanças e refinar progressivamente as soluções encontradas. Todos os elementos foram concebidos de forma a integrar-se à arquitetura da API paralela do método PT desenvolvida por Almeida, de Castro Lima e Carvalho (2025), tendo o foco nos aspectos heurísticos e estruturais diretamente relacionados ao TIP.

4.1 Leitura das instâncias

As instâncias utilizadas neste trabalho não apresentam duplicação de ferramentas nem casos em que o número de ferramentas exceda a quantidade de *slots* disponíveis, o que garante a viabilidade estrutural de todas as soluções geradas. Cada instância é composta por três informações principais:

- O número total de ferramentas (n_f) utilizadas nas operações;
- A capacidade da torreta, definida pelo número de *slots* disponíveis (C);
- Uma matriz de frequência M, de dimensão $n_f \times n_f$, que registra quantas vezes cada par de ferramentas aparece em sequência nas operações previstas.

Durante o processo de leitura, essas informações são extraídas diretamente do arquivo da instância e armazenadas em estruturas auxiliares. A matriz de frequência é lida integralmente e serve de base para a construção da função de avaliação do modelo. Já os parâmetros n_f e C orientam a geração da solução inicial e definem os limites da estrutura de codificação.

Esse processo garante que cada réplica inicializada no algoritmo de revenimento paralelo receba uma configuração válida, compatível com as restrições impostas pela instância e adequada para ser refinada ao longo da execução.

4.2 Codificação e decodificação

Neste trabalho, adotou-se uma representação vetorial unidimensional e com lógica circular, com comprimento n_f, que corresponde ao número total de ferramentas distintas utilizadas. Essa estrutura representa uma torreta em uma máquina CNC, onde ferramentas são organizadas fisicamente em torno de um eixo rotacional. Por exemplo, em uma instância com 5 ferramentas e

8 *slots*, uma codificação possível é

$$[f_3,\ f_2,\ f_5,\ f_1,\ f_4].$$

A decodificação é o processo de interpretação da codificação de uma solução para fins de avaliação. De tal forma, é necessário, antes de calcular o custo, identificar as posições atuais dessas ferramentas na torreta. Considere, por exemplo, uma codificação atual dada por $[f_3,\ f_2,\ f_5,\ f_1,\ f_4]$. Para essas ferramentas, a decodificação produz o seguinte mapeamento:

$$\begin{aligned}
f_3 &\rightarrow \text{posição } 1\\
f_2 &\rightarrow \text{posição } 2\\
f_5 &\rightarrow \text{posição } 3\\
f_1 &\rightarrow \text{posição } 4\\
f_4 &\rightarrow \text{posição } 5\\
vazio &\rightarrow \text{posição } 6\\
vazio &\rightarrow \text{posição } 7\\
vazio &\rightarrow \text{posição } 8
\end{aligned}$$

Esse mapeamento é o insumo necessário para a função de avaliação. A qualidade da decodificação impacta diretamente a robustez e a eficiência da avaliação, especialmente porque a torreta possui simetria rotacional, ou seja, diferentes representações podem corresponder a comportamentos operacionais idênticos. A estrutura da decodificação deve, portanto, ser capaz de lidar com essas simetrias de forma consistente.

4.3 Solução inicial

A solução inicial representa o ponto de partida de cada réplica no algoritmo de revenimento paralelo. No contexto do TIP, cada réplica parte de uma configuração distinta da torreta, permitindo uma exploração mais ampla do espaço de busca e reduzindo a chance de convergência prematura para mínimos locais. A seguir, são descritas as duas abordagens utilizadas neste trabalho para gerar soluções iniciais: uma aleatória e outra baseada em uma heurística gulosa.

4.3.1 Geração aleatória

Nesta abordagem, os índices das ferramentas identificadas na instância são embaralhados por meio de uma permutação aleatória. O vetor resultante é utilizado para preencher os primeiros *slots* da torreta, garantindo que cada ferramenta apareça exatamente uma vez. Essa permutação é distinta para cada réplica, promovendo diversidade no processo de busca.

Embora essa construção não busque soluções de custo otimizado, ela evita configurações artificiais e fornece uma base robusta para a aplicação da cadeia de Markov. O objetivo é oferecer uma configuração viável, compacta e estruturalmente coerente com a operação física da torreta.

Por exemplo, para uma instância com sete ferramentas e dez *slots*, uma solução inicial possível gerada por permutação seria

$$[f_4,\ f_1,\ f_6,\ f_7,\ f_3,\ f_2,\ f_5]$$

4.3.2 Geração inicial mista: aleatoriedade e heurística guiada por frequência

Além da geração aleatória, este trabalho adota uma abordagem complementar baseada em uma heurística gulosa para a criação da solução inicial de réplicas selecionadas. Essa heurística tem como objetivo aproximar as ferramentas mais frequentemente utilizadas na instância, com base na matriz de frequência M.

Durante o pré-processamento, a parte superior de M é analisada para identificar os pares de ferramentas com maior frequência de ocorrência. As ferramentas que apresentam alta interação são posicionadas lado a lado na torreta, formando um bloco com expectativa de menor custo. A ordem é definida de forma a maximizar a proximidade entre os pares mais recorrentes.

Essa abordagem é aplicada a duas réplicas específicas: a de temperatura mais baixa (responsável por explorar soluções de melhor qualidade) e a réplica intermediária. As demais continuam sendo inicializadas com permutações aleatórias, garantindo a manutenção da variabilidade entre as réplicas, um aspecto essencial para a efetividade do revenimento paralelo.

Com isso, busca-se aliar duas estratégias: a convergência rápida proporcionada pela heurística estruturada e a capacidade de diversificação promovida pela aleatoriedade. A coexistência dessas abordagens potencializa a eficiência da busca e o equilíbrio entre exploração e intensificação.

Na Figura 4.1, é apresentada uma simulação da alocação inicial de sete ferramentas entre cinco réplicas distintas, denotadas por R1 até R5, ordenadas por temperatura crescente. As réplicas R1 (mais fria) e R3 (intermediária) recebem configurações iniciais construídas pela heurística gulosa, com ferramentas organizadas de acordo com suas maiores frequências de ocorrência. As demais réplicas (R2, R4 e R5) são inicializadas com permutações aleatórias, contribuindo para a diversidade da população de soluções.

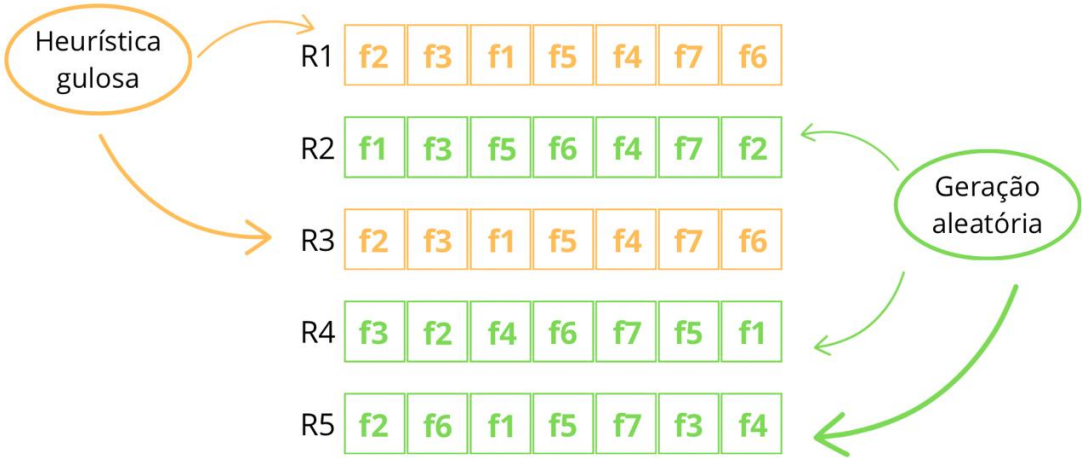

Figura 4.1 – Distribuição de soluções iniciais por heurística e aleatoriedade entre as réplicas.

4.4 Estruturas de vizinhança

Para possibilitar a exploração do espaço de soluções por meio de cadeias de Markov, é necessário definir movimentos que, a partir de uma configuração atual, gerem novas soluções com pequenas alterações. Tais movimentos formam diferentes estruturas de vizinhança, como as descritas a seguir.

4.4.1 Troca

Uma das estruturas de vizinhanças utilizadas neste trabalho é o movimento do tipo troca. Esse operador seleciona duas posições distintas no vetor de codificação e troca os conteúdos dos *slots* correspondentes, gerando uma nova configuração da torreta.

A Figura 4.2 ilustra uma aplicação do operador troca. A linha superior apresenta a solução s, com sete ferramentas organizadas sequencialmente. Após a troca entre as posições de f_3 e f_5, obtém-se a nova configuração s'.

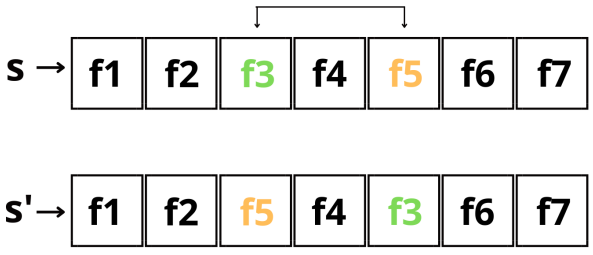

Figura 4.2 – Estrutura de vizinhança troca.

4.4.2 Inserção

Outra estrutura de vizinhança considerada neste trabalho é o movimento do tipo *inserção*. Nesse operador, uma ferramenta é selecionada aleatoriamente e removida de sua posição atual na torreta e inserida em outra posição válida, deslocando as demais ferramentas entre essas posições para manter a integridade da solução.

Considere, por exemplo, a Figura 4.3, em que s representa a configuração inicial das ferramentas na torreta. Nessa estrutura, aplica-se a operação de inserção removendo a ferramenta f_3 da posição 3 e inserindo-a na posição 5. A nova configuração resultante é representada por s' na própria figura, evidenciando a realocação de f_3 e o deslocamento subsequente das demais ferramentas para acomodar a nova ordem.

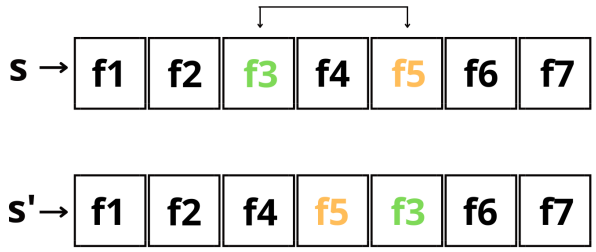

Figura 4.3 – Estrutura de vizinhança inserção.

4.4.3 Inversão

O operador de inversão é uma estrutura de vizinhança que inverte a ordem de um segmento contínuo de ferramentas dentro do vetor de codificação. Ao selecionar duas posições aleatórias no vetor, todos os elementos entre essas posições são revertidos. Esse tipo de movimento é especialmente útil para realinhar blocos inteiros de ferramentas, o que pode levar a alterações substanciais no padrão de deslocamentos.

Considere, por exemplo, a configuração original s da Figura 4.4. Selecionando as posições 1 e 5 para aplicação da inversão, o segmento intermediário é invertido, resultando na nova configuração s'.

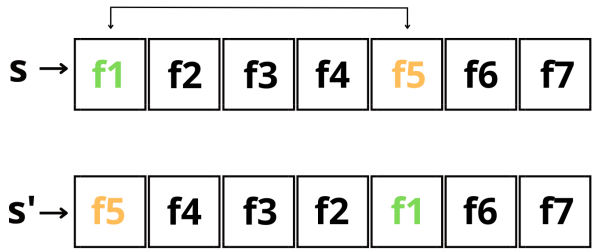

Figura 4.4 – Estrutura de vizinhança inversão.

4.4.4 Combinação estocástica de vizinhanças

Além da aplicação individual dos operadores descritos anteriormente, este trabalho também propõe uma abordagem híbrida, em que os movimentos *troca*, *inserção* e *inversão* são combinados de forma estocástica. A cada iteração, o algoritmo seleciona aleatoriamente um dos três operadores disponíveis, atribuindo a todos a mesma probabilidade de escolha.

Essa estratégia visa ampliar a diversidade das soluções geradas ao longo da busca, evitando que o algoritmo se restrinja a padrões repetitivos de modificação. A aleatoriedade na escolha do operador promove variações tanto de curta quanto de longa escala nas configurações da torreta, permitindo que diferentes regiões do espaço de soluções sejam exploradas de maneira mais abrangente.

Essa combinação de estruturas de vizinhança é particularmente benéfica em um contexto de execução paralela com múltiplas réplicas, pois contribui para que cada réplica percorra trajetórias distintas no espaço de busca. Como resultado, aumenta-se a chance de descoberta de soluções de melhor qualidade ao longo do processo de revenimento paralelo.

Como pode ser visto na Figura 4.5, essa escolha aleatória entre operadores é modelada como um processo uniforme, no qual qualquer um dos três movimentos pode ser aplicado em cada iteração, com probabilidade igual.

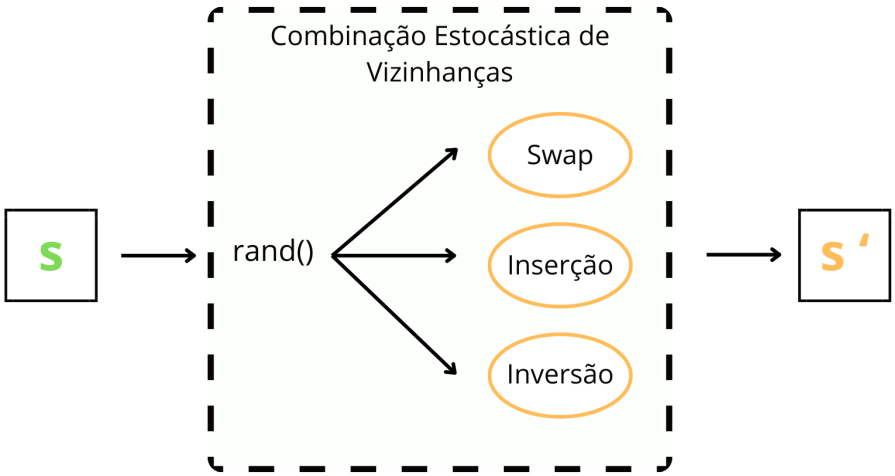

Figura 4.5 – Escolha estocástica entre operadores de vizinhança.

4.5 Função de avaliação

A função de avaliação tem papel central na resolução do TIP, sendo responsável por mensurar a qualidade de uma dada configuração de ferramentas na torreta. Essa avaliação consiste em calcular o custo total associado às trocas de ferramentas, com base na sequência de operações da instância e na posição atual das ferramentas no vetor codificado.

Para que a função de avaliação represente fielmente as exigências da instância, emprega-se a matriz de frequência M, na qual cada elemento $M_{i,j}$ indica o número de vezes que o par de ferramentas (f_i, f_j) ocorre em sequência ao longo das operações. A matriz é simétrica por construção, de modo que a frequência de transição entre duas ferramentas é registrada tanto em $M_{i,j}$ quanto em $M_{j,i}$.

Essa duplicação estrutural garante que a frequência total de um par possa ser obtida independentemente da ordem de indexação. Para otimizar o cálculo, apenas a metade superior da matriz é percorrida durante a avaliação; ao final, o somatório parcial é multiplicado por 2, assegurando que todas as transições entre pares distintos sejam contabilizadas integralmente. Essa abordagem reduz o custo computacional durante as milhares de vezes que a função de avaliação é calculada por diferentes réplicas, contribuindo para a eficiência do algoritmo sem comprometer a precisão dos resultados.

Considere a matriz de frequência M, de dimensão 5×5, que corresponde a uma instância com 5 ferramentas distintas (f_1 a f_5).

$$M = \begin{bmatrix} 0 & 2 & 10 & 10 & 14 \\ 2 & 0 & 16 & 6 & 8 \\ 10 & 16 & 0 & 2 & 10 \\ 10 & 6 & 2 & 0 & 14 \\ 14 & 8 & 10 & 14 & 0 \end{bmatrix}$$

Neste exemplo, o valor $M_{1,3} = 10$ indica que a ferramenta f_1 é seguida 10 vezes pela ferramenta f_3 ao longo da sequência de operações. Como M é simétrica, temos $M_{3,1} = M_{1,3}$.

A partir da matriz de frequência M, o custo total da solução é obtido considerando, para cada par (f_i, f_j) com frequência $f_{i,j} > 0$, a menor distância circular $d_{i,j}$ entre as posições dessas ferramentas na codificação atual, calculada conforme a Equação (3.1). Essa distância é medida no sentido horário ou anti-horário ao longo dos C *slots*, e o custo parcial do par resulta do produto $f_{i,j} \times d_{i,j}$. O somatório desses custos parciais fornece o custo total, refletindo diretamente a eficiência da configuração analisada.

$$\text{Custo total} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij} \cdot d(f_i, f_j)$$

Esse modelo de avaliação torna o problema altamente sensível a pequenas mudanças estruturais, que podem alterar significativamente o custo ao aproximar ou afastar ferramentas com alta frequência de uso conjunto. Por exemplo, considere uma torreta já decodificada com $C = 10$ *slots* e a seguinte ordem de ferramentas:

$$[f_1,\ f_4,\ f_2,\ f_3,\ f_5,\ \_,\ \_,\ \_,\ \_,\ \_]$$

O cálculo da distância entre f_1 e f_3 é, portanto,

$$d(f_1, f_3) = \min(|3 - 0|,\ 10 - |3 - 0|) = \min(3,\ 7) = 3$$

Usando o exemplo da matriz F , em que a frequência entre f_1 e f_3 é 10, e sabendo que f_1 está na posição 0 e f_3 na posição 3, temos:

$$f_{13} \cdot d(f_1, f_3) = 10 \cdot 3 = 30$$

Esse valor é então somado ao custo total da solução. O mesmo processo se repete para todos os pares com frequência positiva.

5 Experimentos Computacionais

Neste capítulo, são reportados os experimentos realizados para avaliar o desempenho da abordagem proposta na resolução do TIP. Foram conduzidos testes sobre um conjunto padronizado de instâncias, com o objetivo de validar a eficácia do método, analisar a variabilidade entre execuções independentes e comparar os resultados com os valores de referência da literatura. Para a execução, foi utilizada a API paralela do PT desenvolvida por Almeida, de Castro Lima e Carvalho (2025).

Os experimentos foram executados em uma máquina com processador Intel Core i7-4790 CPU @ 3.60GHz, equipada com 4 cores, 16 GB de memória RAM e sistema operacional Ubuntu 18.04.6 LTS. O código foi implementado em C++, utilizando o compilador g++ 7.5.0 com suporte ao padrão C++17, paralelismo via -pthread e otimizações ativadas com a flag -O3. O código-fonte utilizado nos experimentos está disponível publicamente no GitHub[1] para consulta e reprodução dos resultados.

5.1 Instâncias

As instâncias utilizadas neste trabalho são as mesmas propostas por Baykasoglu, Yoruk e Yildiz (2024), considerado a referência estado da arte. A Tabela 5.1 define, para cada conjunto de instâncias, o número de ferramentas utilizadas (n_f), o número de *slots* disponíveis na torreta (C), e uma matriz de frequência F de dimensão $n_f \times n_f$, que registra quantas vezes cada par de ferramentas aparece em sequência nas operações.

Tabela 5.1 – Resumo das instâncias utilizadas nos experimentos

Conjunto	C	n_f	**instâncias**
`anjos-60`	100	60	5
`anjos-70`	100	70	5
`anjos-75`	100	75	6
`anjos-80`	100	80	5
`o-*`	5–30	5–20	13
`s-*`	20–30	12–25	15
`sko-*`	60–100	36–100	8
`y-*`	6–100	6–60	25

[1] <https://github.com/LorrayneCristine/Monografia-II_Parallel-tempering-applied-to-the-problem-of-tool-indexing/tree/main>

5.2 Ajuste de parâmetros

Para a configuração do PT destacam-se dois parâmetros adicionais, ambos definidos a partir da implementação disponível na API de Almeida, de Castro Lima e Carvalho (2025): a distribuição inicial das temperaturas e o ajuste dinâmico. A primeira conta com quatro alternativas — linear e linear inversa (espaçamento constante direto ou invertido), exponencial (maior densidade nas temperaturas frias) e progressão geométrica (balanceamento ao longo do espectro). Já o ajuste dinâmico pode seguir três estratégias: equalização da probabilidade de troca (uniformiza taxas de aceitação), taxa-alvo (mantém a aceitação em torno de 23,4%) e *feedback-optimized* (maximiza a eficiência dos percursos térmicos).

Neste trabalho, os valores finais foram definidos por meio da ferramenta *irace* (López-Ibáñez et al., 2016), um pacote em linguagem R amplamente utilizado para ajuste automático de parâmetros em algoritmos estocásticos. A ferramenta realiza uma busca orientada por instâncias de treino, selecionando combinações promissoras com base no desempenho médio obtido. A Tabela 5.2 apresenta o conjunto de parâmetros considerados durante esse processo.

Tabela 5.2 – Parâmetros explorados no ajuste com *irace*.

Parâmetro	**Opções testadas**
Temperatura inicial (`TEMP_INIT`)	{0.01, 0.02, 0.03}
Temperatura final (`TEMP_FIM`)	{10.0, 20.0, 30.0}
Comprimento da cadeia de Markov (`MCL`)	{300, 400, 500, 600}
Distribuição de temperatura[a] (`TEMP_DIST`)	{1, 2, 3, 4}
Tipo de movimento[b] (`MOV_TYPE`)	{1, 2, 3, 4}
Tipo de solução inicial[c] (`MODE`)	{1, 2}
Tipo de ajuste de temperatura[d] (`TYPE_UPDATE`)	{1, 2, 3}
Frequência de ajuste[e] (`TEMP_UPDATE`)	{3, 4, 5}

[a] 1 - Linear, 2 - Linear inversa, 3 - Exponencial, 4 - Progressão geométrica.
[b] 1 - *Swap*, 2 - *Insertion*, 3 - Inversão, 4 - Combinação estocástica.
[c] 1 - Geração inicial aleatória, 2 - Geração inicial mista.
[d] 1 - 23%, 2 - Igualar taxas de aceitação, 3 - Otimização com feedback.
[e] Valor que divide o MCL para definir o intervalo de atualização térmica.

Particularmente, o intervalo entre atualizações de temperatura é calculado com base na frequência de atualização selecionada, utilizando a fórmula:

$$\texttt{tempUpdate} = \frac{\texttt{PTL}}{\text{frequência de atualização}}$$

Além dos parâmetros definidos via *irace*, o algoritmo também utiliza argumentos internos fixos. O número de réplicas é determinado automaticamente como o número de núcleos físicos da máquina menos um; neste experimento, foram utilizadas três réplicas, explorando o paralelismo

disponível. O número total de propostas de troca entre temperaturas é fixado em 2200, pela constante `PTL`. Esses valores complementam os parâmetros otimizados e asseguram a execução estável e controlada do método ao longo das iterações em todas as réplicas.

Após a execução do processo de calibração, o *irace* identificou as combinações mais promissoras com base nos desempenhos médios obtidos. A Tabela 5.3 apresenta as quatro melhores configurações selecionadas, ordenadas de acordo com o peso atribuído pela ferramenta, após diversas iterações de avaliação em instâncias de treino. A coluna "Peso" representa a probabilidade relativa com que cada configuração foi considerada promissora pelo *irace*, com a configuração de ID 298 sendo a mais favorecida e a utilizada para realização dos experimentos computacionais.

Tabela 5.3 – Melhores configurações identificadas pelo *irace* após o ajuste de parâmetros.

ID	MOV_TYPE	TEMP_INIT	TEMP_FIM	MCL	TEMP_DIST	TYPE_UPDATE	TEMP_UPDATE	MODE	PESO
298	4	0,01	20,0	400	2	2	4	2	0,4
53	4	0,02	10,0	300	2	3	5	2	0,3
339	4	0,02	10,0	300	1	2	4	2	0,2
393	4	0,03	20,0	400	4	2	3	2	0,1

5.3 Comparação com o estado da arte

Os experimentos desta seção têm como objetivo comparar os resultados obtidos pelo PT com aqueles reportados na literatura por Baykasoglu, Yoruk e Yildiz (2024), considerado o estado da arte. Embora o conjunto de testes completo tenha incluído um número maior de instâncias, para a análise comparativa foram consideradas apenas as instâncias para as quais o artigo de referência disponibilizou resultados, de modo a permitir uma comparação direta.

Cada instância foi resolvida 10 vezes de forma independente, com os mesmos parâmetros de configuração. Essa abordagem tem dois objetivos principais: (1) mitigar a influência de fatores estocásticos inerentes ao método e (2) permitir a análise de estabilidade do desempenho. Para cada instância, são apresentadas três métricas principais:

- **gap (%)**: diferença percentual entre o melhor valor de solução obtido nas 10 execuções e o valor reportado na literatura. Um *gap* nulo indica que o método alcançou o mesmo valor de referência, enquanto valores negativos indicam que foram encontradas novas soluções melhores que as reportadas na literatura. O *gap* é calculado conforme a Equação (5.1).

$$gap(\%) = \frac{Z_{\text{pt}} - Z_{\text{ref}}}{Z_{\text{ref}}} \times 100 \tag{5.1}$$

  onde Z_{PT} representa o custo obtido neste trabalho e Z_{ref} o custo reportado por Baykasoglu, Yoruk e Yildiz (2024).

- **Desvio médio (%)**: representa a média das diferenças percentuais entre cada execução e o melhor resultado obtido para a mesma instância. Esse indicador avalia a consistência do

método dentro de um conjunto de execuções. Quanto menor o valor, menor a dispersão dos resultados em relação ao melhor encontrado. O cálculo do desvio médio interno é dado pela Equação (5.2):

$$D_{\text{médio-int}}(\%) = \frac{1}{n}\sum_{i=1}^{n}\frac{Z_i - Z_{best}}{Z_{best}} \times 100 \tag{5.2}$$

onde n é o número de execuções (neste caso, $n = 10$), Z_i é o custo obtido na execução i e Z_{best} é o menor valor encontrado entre as n execuções.

- **Tempo médio (s)**: tempo de execução médio, em segundos, considerando as 10 execuções da instância. Essa métrica evidencia a relação entre o esforço computacional e a complexidade da instância.

A Tabela 5.4 apresenta, para cada instância selecionada, o número de *slots* (C), o número de ferramentas (n_f), o valor da melhor solução obtida no presente trabalho (Z_{PT}), o valor de referência da literatura Z_{ref}, o *gap* entre estes valores, o desvio médio e o tempo médio de execução do PT. Os melhores valores de solução são destacados em negrito.

Tabela 5.4 – Comparativo entre resultados obtidos neste trabalho e valores reportados por Baykasoglu, Yoruk e Yildiz (2024).

Instância	C	n_f	Z_{PT}	Z_{ref}	**Gap (%)**	**Desvio (%)**	**Tempo (s)**
anjos-60-01	100	60	**108106**	**108106**	0,00	0,01	8,84
anjos-60-02	100	60	**62548**	**62548**	0,00	0,01	7,87
anjos-60-03	100	60	**47018**	**47018**	0,00	0,19	6,94
anjos-60-04	100	60	23184	**23182**	0,01	0,30	5,65
anjos-60-05	100	60	**30336**	**30336**	0,00	0,11	6,15
anjos-70-01	100	70	84604	**84592**	0,01	0,17	9,29
anjos-70-02	100	70	**103446**	**103446**	0,00	0,00	10,06
anjos-70-03	100	70	**87588**	**87588**	0,00	0,08	9,67
anjos-70-04	100	70	**55402**	**55402**	0,00	0,08	8,27
anjos-70-05	100	70	**268476**	**268476**	0,00	0,03	12,29
anjos-75-01	100	75	**133260**	**133260**	0,00	0,06	11,64
anjos-75-02	100	75	**223612**	**223612**	0,00	0,01	13,05
anjos-75-03	100	75	**76302**	**76302**	0,00	0,03	9,98
anjos-75-04_2	100	75	**212682**	**212682**	0,00	0,00	13,52
anjos-75-05	100	75	**94034**	**94034**	0,00	0,06	10,70
anjos-80-01	100	80	**108926**	**108926**	0,00	0,00	12,36
anjos-80-02	100	80	**105702**	**105702**	0,00	0,05	12,47
anjos-80-03	100	80	**190182**	**190182**	0,00	0,00	14,07
anjos-80-04	100	80	**201656**	**201656**	0,00	0,00	14,03

Continua na próxima página

Tabela 5.4 – *Continuação da página anterior*

Instância	C	n_f	$Z_{\mathbf{PT}}$	$Z_{\mathbf{ref}}$	**Gap (%)**	**Desvio (%)**	**Tempo (s)**
anjos-80-05	100	80	**72428**	**72428**	0,00	0,08	13,30
o-10_t_2	12	10	**2544**	**2544**	0,00	0,00	0,76
o-10_t_3	15	10	**2756**	**2756**	0,00	0,00	0,75
o-10_t_4	20	10	**2804**	**2804**	0,00	0,00	0,75
o-10_t_5	25	10	**2804**	**2804**	0,00	0,00	0,76
o-10_t_6	30	10	**2804**	**2804**	0,00	0,00	0,75
o-10_t	10	10	**2264**	**2264**	0,00	0,00	0,75
o-15_t_2	30	15	**10268**	**10268**	0,00	0,00	1,06
o-15_t	20	15	**9636**	**9636**	0,00	0,00	1,06
o-20_t	30	20	**25200**	**25200**	0,00	0,00	1,56
o-5_t_2	10	5	**300**	**300**	0,00	0,00	0,54
o-5_t	5	5	**248**	**248**	0,00	0,00	0,54
o-6_t	10	6	**584**	**584**	0,00	0,00	0,57
o-7_t	10	7	**936**	**936**	0,00	0,00	0,58
o-8_t	10	8	**1448**	**1448**	0,00	0,00	0,63
o-9_t	10	9	**1732**	**1732**	0,00	0,00	0,70
s-12_t	20	12	**8818**	**8818**	0,00	0,00	0,89
s-13_t_2	30	13	**11794**	**11794**	0,00	0,00	0,97
s-13_t	20	13	**11562**	**11562**	0,00	0,00	0,96
sko-100	100	100	**577866**	577966	-0,02	0,19	24,95
sko-36	60	36	**20574**	**20574**	0,00	0,00	2,05
sko-42	60	42	**48812**	**48812**	0,00	0,16	3,89
sko-49	60	49	73040	**73034**	0,01	0,16	5,01
sko-56	60	56	**105330**	**105330**	0,00	0,12	6,34
sko-64	100	64	**190374**	190380	-0,00	0,11	8,30
sko-72	100	72	**265136**	**265136**	0,00	0,25	10,22
sko-81	100	81	367870	**367776**	0,03	0,56	13,32
y-10_t	10	10	**5422**	**5422**	0,00	0,00	0,75
y-11_t_2	12	11	**6700**	**6700**	0,00	0,00	0,83
y-11_t	11	11	**6166**	**6166**	0,00	0,00	0,83
y-12_t	12	12	**7354**	**7354**	0,00	0,00	0,89
y-13_t	13	13	**8392**	**8392**	0,00	0,00	0,97
y-14_t	14	14	**9852**	**9852**	0,00	0,00	1,05
y-15_t_2	30	15	**14718**	**14718**	0,00	0,00	1,06
y-20_t	30	20	**23770**	**23770**	0,00	0,00	1,56
y-30_t	60	30	**55346**	**55346**	0,00	0,03	2,93

Continua na próxima página

Tabela 5.4 – *Continuação da página anterior*

Instância	C	n_f	Z_{PT}	Z_{ref}	**Gap (%)**	**Desvio (%)**	**Tempo (s)**
y-35_t	60	35	**76028**	**76028**	0,00	0,00	3,78
y-40_t	60	40	**92652**	**92652**	0,00	0,03	4,76
y-45_t	60	45	**117620**	**117620**	0,00	0,03	5,91
y-50_t	60	50	**146598**	146630	-0,02	0,08	7,11
y-60_t	100	60	222458	**222412**	0,02	0,02	9,75
y-6_t_2	10	6	**2744**	**2744**	0,00	0,00	0,57
y-6_t	6	6	**2126**	**2126**	0,00	0,00	0,57
y-7_t	10	7	**3466**	**3466**	0,00	0,00	0,58
y-8_t	10	8	**4176**	**4176**	0,00	0,00	0,63
y-9_t	10	9	**4864**	**4864**	0,00	0,00	0,70

Analisando os resultados, observa-se que o método implementado reproduziu com elevada precisão os custos reportados na literatura. Em grande parte das instâncias, o melhor valor conhecido foi igualado, enquanto em algumas instâncias foram alcançadas soluções melhores que as reportadas por Baykasoglu, Yoruk e Yildiz (2024). Apenas para cinco instâncias não foram atingidos os melhores valores de referência, porém, mesmo nesses casos, a diferença foi mínima, com *gap* inferior a 0,03%.

A convergência para a melhor solução ocorreu, em média, no ciclo 195,3 do PTL, considerando todas as execuções e instâncias, enquanto o caso mais tardio foi observado no ciclo 2164. Nas instâncias menores, uma parcela expressiva dos ótimos surgiu já no ciclo 0 (primeira avaliação), indicando alta efetividade da construção inicial e menor complexidade do espaço de busca.

O desvio médio entre as execuções, que mede a estabilidade do método, manteve-se baixo. Em praticamente todas as instâncias, esta variação não ultrapassou 0,2%, confirmando que múltiplas execuções independentes do PT não resultaram em variação significativa nos resultados obtidos. Isto demonstra a robustez da abordagem e a baixa influência de fatores aleatórios na obtenção das soluções finais.

As instâncias dos conjuntos `o` e `s`, com n_f reduzido e valores baixos de C, exigiram tempos de execução bem menores, quase sempre inferiores a 1,3 segundos, mantendo *gap* nulo e o desvio médio zero. Nos conjuntos `anjos` e `y`, que apresentam instâncias de tamanho moderado, os tempos de execução variaram entre 5 e 14 segundos, com comportamento previsível em função do número de ferramentas. Observa-se que o tempo cresce de forma proporcional ao aumento de n_f, enquanto o *gap* permanece praticamente estável.

O conjunto `sko`, que concentra instâncias maiores, exigiu tempos médios mais elevados, chegando a 24,95 segundos para a instância `sko-100`. Apesar desse valor ser superior aos tempos observados nas demais instâncias, ele ainda é considerado bastante baixo em termos de custo

computacional, especialmente levando em conta a dimensão do problema. A qualidade das soluções manteve-se alinhada, com *gap* e o desvio médio controlado, reforçando que o aumento do tamanho das instâncias impactou o esforço computacional, mas não comprometeu a qualidade dos resultados.

Neste trabalho optou-se por não realizar uma comparação direta com os valores reportados por Baykasoglu, Yoruk e Yildiz (2024). Essa decisão decorre do fato de que os experimentos foram realizados em máquinas com características distintas e o código-fonte não foi disponibilizado, o que inviabiliza a reprodução exata do ambiente de testes. Ainda assim, observa-se que ambos os trabalhos apresentaram tempos bastante reduzidos, mesmo nas maiores instâncias.

De maneira geral, a comparação com o estado da arte confirmou que o método foi capaz de reproduzir com alta fidelidade os resultados reportados por Baykasoglu, Yoruk e Yildiz (2024), alcançando ou superando o desempenho da literatura na maioria dos casos. O baixo desvio médio, a proximidade do *gap* com zero e o comportamento estável do tempo de execução reforçam a confiabilidade e robustez da implementação do PT.

6 Conclusão

A otimização da indexação de ferramentas é um fator crítico para a eficiência dos processos industriais em máquinas numericamente controladas, especialmente em contextos de manufatura flexível, pois contribui diretamente para a redução do tempo de operação e, por consequência, dos custos produtivos. Diante dessa relevância, este trabalho investigou o problema de indexação de ferramentas (ou *tool indexing problem*, TIP), abordando sua formalização, importância prática e as principais estratégias encontradas na literatura. Este estudo implementou e avaliou o método revenimento paralelo (ou *parallel tempering*, PT) aplicado ao TIP, buscando posicioná-lo como uma alternativa para a resolução do problema. Os experimentos computacionais demonstraram que o método proposto foi capaz de reproduzir com elevada precisão os resultados da literatura, alcançando ou superando os valores de referência em grande parte das instâncias testadas. Um aspecto relevante foi a robustez do método, confirmada pelo baixo desvio médio entre execuções. Além disso, o tempo de execução manteve-se competitivo, alcançando no máximo 24,95 segundos, considerado baixo para o porte do problema. Os resultados obtidos evidenciam que o método proposto apresenta boa relação entre desempenho computacional e qualidade de solução, configurando-se como uma alternativa viável e escalável frente aos métodos existentes na literatura. Como próximos passos, pretende-se (i) expandir o conjunto de instâncias avaliadas; (ii) compilar o estudo no formato de artigo científico; (iii) versão multiobjetivo da abordagem proposta; (iv) tratar cenários com mais ferramentas do que *slots* disponíveis; e (v) incorporar a duplicação de ferramentas, avaliando seus impactos em custo e robustez.

Referências

ALMEIDA, A. L. B.; de Castro Lima, J.; CARVALHO, M. A. M. Revisiting the parallel tempering algorithm: High-performance computing and applications in operations research. *Computers & Operations Research*, v. 178, p. 107000, 2025. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054825000280>.

AMOUZGAR, K.; NOURMOHAMMADI, A.; NG, A. H. Multi-objective optimisation of tool indexing problem: a mathematical model and a modified genetic algorithm. *International Journal of Production Research*, Taylor & Francis, v. 59, n. 12, p. 3572–3590, 2021. Disponível em: <https://doi.org/10.1080/00207543.2021.1897174>.

ATTA, S.; MAHAPATRA, P. R. S.; MUKHOPADHYAY, A. Solving tool indexing problem using harmony search algorithm with harmony refinement. *Soft Computing*, v. 23, p. 7407–7423, 2018. Disponível em: <https://doi.org/10.1007/s00500-018-3385-5>.

BAYKASOGLU, A.; DERELI, T. Heuristic optimization system for the determination of index positions on cnc magazines with the consideration of cutting tool duplications. *International Journal of Production Research*, Taylor & Francis, v. 42, n. 7, p. 1281–1303, 2004.

BAYKASOGLU, A.; YORUK, E.; YILDIZ, S. T. Turret-index optimisation with mathematical programming and metaheuristic approaches. *International Journal of Production Research*, 2024. Disponível em: <https://doi.org/10.1080/00207543.2024.2399711>.

BAYKASOğLU, A.; OZSOYDAN, F. B. An improved approach for determination of index positions on cnc magazines with cutting tool duplications by integrating shortest path algorithm. *International Journal of Production Research*, v. 54, n. 3, p. 742–760, 2015.

BAYKASOğLU, A.; OZSOYDAN, F. B. Minimizing tool switching and indexing times with tool duplications in automatic machines. *International Journal of Advanced Manufacturing Technology*, Springer-Verlag London, 2016.

BENOV, D. M. The manhattan project, the first electronic computer and the monte carlo method. *Monte Carlo Methods and Applications*, De Gruyter, v. 22, n. 1, p. 73–79, 2016. Disponível em: <https://doi.org/10.1515/mcma-2016-0102>.

DERELI, T.; BAYKASOğLU, A.; GINDY, N.; FILIZ, I. Determination of optimal turret index positions by genetic algorithms. *International Symposium on Intelligent Manufacturing Systems*, v. 2, p. 743–750, 1998.

DERELI, T.; FILIZ, I. H. Allocating optimal index positions on tool magazines using genetic algorithms. *Robotics and Autonomous Systems*, v. 33, n. 2, p. 155–167, 2000.

DUMAN, E. Modelling the operations of a component placement machine with rotational turret and stationary component magazine. *Journal of the Operational Research Society*, v. 58, n. 3, p. 317–325, 2007. Disponível em: <https://doi.org/10.1057/palgrave.jors.2602152>.

EARL, D. J.; DEEM, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, Royal Society of Chemistry, v. 7, n. 23, p. 3910–3916, 2005.

EARL, D. J.; DEEM, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, Royal Society of Chemistry (RSC), v. 7, n. 23, p. 3910, 2005. ISSN 1463-9084. Disponível em: <http://dx.doi.org/10.1039/B509983H>.

GHOSH, D. *Allocating tools to index positions in tool magazines using tabu search*. Ahmedabad, India, 2016. Accessed: 2025-03-18. Disponível em: <https://www.iima.ac.in>.

HANSMANN, U. H. Parallel tempering algorithm for conformational studies of biological molecules. *Chemical Physics Letters*, v. 281, n. 1, p. 140–150, 1997.

HASTINGS, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, Oxford University Press, v. 57, n. 1, p. 97–109, 1970.

HUKUSHIMA, K.; NEMOTO, K. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, The Physical Society of Japan, v. 65, n. 6, p. 1604–1608, 1996.

JAVAID, M.; HALEEM, A.; SINGH, R. P.; SUMAN, R. Enabling flexible manufacturing system (fms) through the applications of industry 4.0 technologies. *Internet of Things and Cyber-Physical Systems*, v. 2, p. 49–62, 2022. ISSN 2667-3452. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2667345222000153>.

JONES, G. L.; QIN, Q. Markov chain monte carlo in practice. *Annual Review of Statistics and Its Application*, Annual Reviews, v. 9, p. 557–578, 2022. Disponível em: <https://doi.org/10.1146/annurev-statistics-040220-090158>.

KAIGHOBADI, M.; VENKATESH, K. Flexible manufacturing systems: An overview. *International Journal of Operations & Production Management*, v. 14, n. 4, p. 26–49, 1994. Disponível em: <https://doi.org/10.1108/01443579410056029>.

KRISHNA, A. G.; RAO, K. M. Optimal allocation of index positions on tool magazines using an ant colony algorithm. *International Journal of Advanced Manufacturing Technology*, Springer-Verlag London Limited, v. 30, p. 717–721, 2006.

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

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, American Institute of Physics, v. 21, n. 6, p. 1087–1092, 1953.

MONOSTORI, L.; KADAR, B.; BAUERNHANSL, T. Cyber-physical systems in manufacturing. *CIRP Annals*, v. 65, p. 621–641, 2016.

MUñOZ-BENAVENT, P.; SOLANES, J. E.; GRACIA, L.; TORNERO, J. Robust auto tool change for industrial robots using visual servoing. *International Journal of Systems Science*, v. 50, n. 2, p. 432–449, 2019.

NAGAI, T.; PANTELOPULOS, G. A.; TAKAHASHI, T.; STRAUB, J. E. On the use of mass scaling for stable and efficient simulated tempering with molecular dynamics. *Journal of Computational Chemistry*, v. 27, p. 2017–2028, 2016.

OLIVEIRA, E. S. D.; BACCI, S. C. C.; PARIS, L. R. P.; MENEGASSI, W. J. C.; NETO, J. M. F. A. Automação nos processos industriais: Processo de implementação e o papel do gestor de tecnologia da informação. *Prospectus*, v. 6, n. 1, p. 153–203, 2024.

QUDEIRI, J. E. A.; AL-MOMANI, R.; JAMALI, M. A.; YAMAMOTO, H. Optimization hole-cutting operations sequence in cnc machine tools using ga. *International Conference on Service Systems and Service Management*, v. 1, p. 501–506, 2006.

SOORI, M.; JOUGH, F. K. G.; DASTRES, R.; AREZOO, B. Robotical automation in cnc machine tools: A review. *Acta Mechanica et Automatica*, Sciendo, v. 18, n. 3, p. 434–450, 2024.

SWENDSEN, R. H.; WANG, J.-S. Replica monte carlo simulation of spin-glasses. *Physical Review Letters*, American Physical Society, v. 57, n. 21, p. 2607–2609, 1986.

WILKS, D. S. Chapter 10 - time series. In: WILKS, D. S. (Ed.). *Statistical Methods in the Atmospheric Sciences (Fourth Edition)*. Fourth edition. Elsevier, 2019. p. 485–550. ISBN 978-0-12-815823-4. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128158234000109>.

YAO, K.-C.; CHEN, D.-C.; PAN, C.-H.; LIN, C.-L. The development trends of computer numerical control (cnc) machine tool technology. *Mathematics*, MDPI, Changhua City, Taiwan, v. 12, n. 13, p. 1923, 2024. ISSN 2227-7390.