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

MAURO LÚCIO AFONSO PAULINO DOS SANTOS FILHO
Orientador: Prof. Dr. Marco Antonio Moreira de Carvalho

# REVENIMENTO PARALELO APLICADO AO PROBLEMA DE MINIMIZAÇÃO DE PILHAS ABERTAS

Ouro Preto, MG
2025

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

MAURO LÚCIO AFONSO PAULINO DOS SANTOS FILHO

# REVENIMENTO PARALELO APLICADO AO PROBLEMA DE MINIMIZAÇÃO DE PILHAS ABERTAS

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

**Orientador:** Prof. Dr. Marco Antonio Moreira de Carvalho

Ouro Preto, MG
2025

# Agradecimentos

Agradeço aos meus pais Mauro e Soraya, pelo apoio incondicional; aos amigos mais próximos Arthur, Caio e Marcella, pela companhia e incentivo ao longo da jornada; ao meu orientador Marco, pela orientação precisa e atenciosa; e à minha namorada Júlia, pela paciência, carinho e presença constante. A todos, minha sincera gratidão.

# Resumo

O cenário atual exige das indústrias uma maior eficiência na gestão da matéria-prima, dos produtos semiacabados e acabados, impulsionando a busca por processos produtivos otimizados e pelo uso racional dos recursos. Em face desse desafio, esta monografia propõe a aplicação do método revenimento paralelo (*parallel tempering* – PT) para resolver o problema de minimização de pilhas abertas (*minimization of open stack problem* - MOSP). Este problema combinatório consiste em determinar a sequência ideal de processamento de padrões de corte, de modo a reduzir o número máximo de pilhas abertas simultaneamente. Uma pilha é considerada aberta desde a produção da primeira peça de um determinado tipo até que a última seja fabricada, condição que impacta significativamente o aproveitamento do espaço físico, o fluxo produtivo e os custos operacionais. Reconhecendo a natureza NP-difícil do problema, torna-se inviável a utilização de métodos exatos para instâncias de grande porte, típicas em ambientes industriais. Assim, a estratégia adotada baseia-se no PT, que se destaca na amostragem eficiente de distribuições de probabilidade complexas e no enfrentamento de problemas de otimização combinatória. A implementação paralela do PT visa alcançar soluções de alta qualidade em tempo compatível com as demandas do setor. Os resultados obtidos com o PT-MOSP confirmam essa expectativa, demonstrando a capacidade do método em encontrar soluções de qualidade competitiva para o problema, superando o método de referência (*PieceRank*) e consolidando-se como o novo estado da arte na resolução do MOSP, reforçando sua aplicabilidade prática em cenários industriais.

**Palavras-chave**: Minimização de Pilhas Abertas. Revenimento Paralelo.

# Abstract

The current scenario demands greater efficiency in the management of raw materials, as well as semi-finished and finished products in the industrial sector, driving the pursuit of optimized production processes and rational resource usage. In light of this challenge, this monograph proposes the application of the parallel tempering (PT) method to solve the minimization of open stacks problem (MOSP). This combinatorial problem involves determining the optimal sequence for processing cutting patterns to reduce the maximum number of open stacks simultaneously. A stack is defined as open when the first piece of a given type is produced and remains in that state the last piece is manufactured – a condition that significantly impacts the utilization of physical space, production flow, and operational costs. Recognizing the NP-hard nature of the problem, the use of exact methods becomes unfeasible for large-scale instances typically found in industrial environments. Therefore, the proposed strategy is based on PT, which excels at efficiently sampling complex probability distributions and addressing combinatorial optimization problems. The parallel implementation of PT aims to achieve high-quality solutions within a timeframe compatible with industry demands. The results obtained with PT-MOSP confirm this expectation, demonstrating the method's ability to find high-quality solutions for the problem, outperforming the reference method (PieceRank) and establishing itself as the new state of the art in solving the MOSP, thereby reinforcing its practical applicability in industrial settings.

**Keywords**: Minimization of Open Stacks Problem. Parallel Tempering.

# Lista de Ilustrações

# Lista de Tabelas

# Lista de Algoritmos

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| AS | *Ashikaga-Soma* |
| BRKGA | *biased random-key genetic algorithm* |
| MCMC | *Monte Carlo Markov chain* |
| MCNH | *minimum cost node heuristic* |
| MOSP | *minimization of open stack problem* |
| PT | *parallel tempering* |
| SA | *simulated annealing* |
| TPSA | *temperature parallel simulated annealing* |

# Sumário

# 1 Introdução

No meio industrial e fabril, a eficiência na manipulação de matéria-prima, produtos semiacabados e acabados é fundamental para evitar desperdícios de materiais e perdas financeiras. Para garantir a otimização dos processos de produção, é essencial adotar práticas que racionalizem o uso de recursos, assegurando a sustentabilidade e a rentabilidade da produção (Abensur, 2018). Além disso, a correta armazenagem e o transporte adequado dos produtos são igualmente cruciais, pois impactam diretamente a qualidade do produto final, os custos operacionais e as diferentes métricas de qualidade (Abensur, 2018).

Diversos setores industriais realizam operações de corte de materiais, etapa muitas vezes essencial para a otimização do processo produtivo. Indústrias como a metalúrgica, a têxtil, a moveleira e a de manipulação de vidro são exemplos comuns. Nessas indústrias, aspectos de qualidade, como fragilidade, aparência física e padronização, devem ser cuidadosamente observados. Além disso, algumas métricas de qualidade são comuns a todas elas, como pontualidade nas entregas, ritmo de expedição de produtos e aproveitamento eficiente dos materiais, sendo essenciais para garantir a eficiência e a competitividade no mercado.

Nesse contexto, são apresentados os padrões de corte, ou apenas *padrões*, que consistem em esquemas detalhados que definem a posição e as dimensões de como um objeto em estoque deve ser cortado para dar origem a produtos semiacabados e/ou acabados, referidos de maneira geral como *peças*. Esses padrões funcionam como um gabarito que busca otimizar os recursos provenientes da matéria-prima, como, por exemplo, bobinas de papel e chapas de metal, garantindo cortes com o objetivo de minimizar desperdícios e atender às especificações exigidas pela produção. Um exemplo pode ser visto na Figura 1.1, na qual há 6 padrões e 6 peças. Como se pode observar, o padrão 1 produz as peças 1 e 2; o padrão 2, por sua vez, produz as peças 3, 4 e 6; e assim sucessivamente.

Em muitas aplicações industriais, os cortes seguem restrições específicas, como o corte *guilhotinado*, no qual as peças são obtidas por cortes retos completos que atravessam todo o material em uma única direção. A definição do tipo de corte admitido influencia diretamente na estrutura dos padrões gerados e na complexidade do problema. Dessa forma, a determinação adequada dos padrões, respeitando essas restrições, pode impactar significativamente a eficiência industrial, contribuindo para a redução de custos e a melhoria da produtividade (Cherri et al., 2014).

Adicionalmente, o gerenciamento correto do estoque intermediário desempenha um papel estratégico no fluxo de produção e logística da indústria. O estoque intermediário consiste em um espaço reduzido no ambiente de produção, localizado ao redor das máquinas, em que são armazenadas as peças durante etapas do processo produtivo ou antes do envio para o cliente.

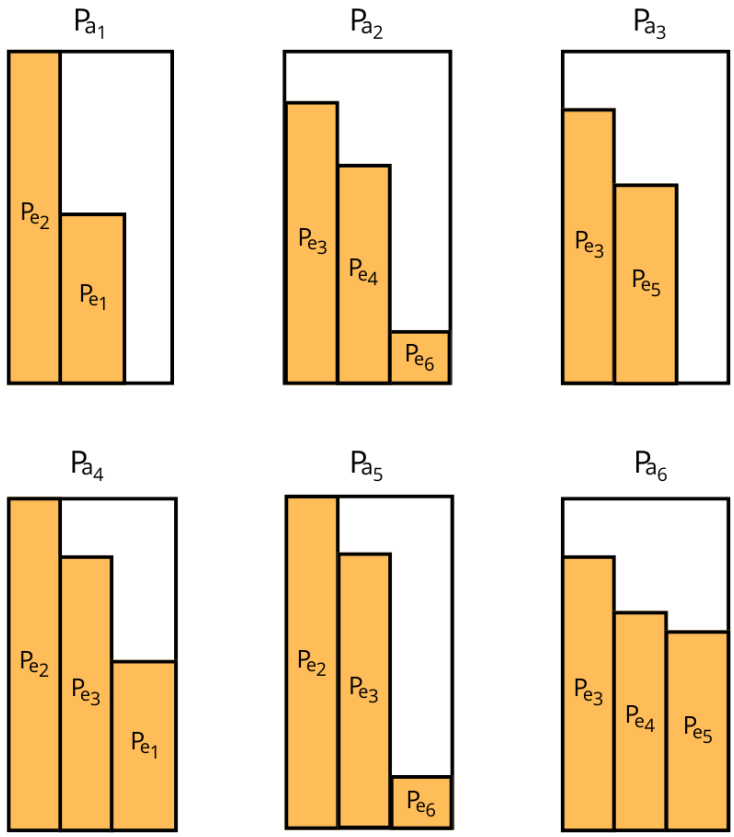


Figura 1.1 – Padrões de corte e as respectivas peças que produzem.

Dessa forma, o estoque intermediário funciona como um armazenamento temporário que absorve oscilações de demanda e compensa atrasos na produção (Ito, 2018). Esse tipo de armazenamento temporário acelera a expedição de pedidos, reduzindo o tempo de separação e transporte, pois os materiais ficam mais próximos das áreas de produção e carregamento, eliminando etapas de movimentação que seriam necessárias no caso do estoque final, que geralmente está localizado em áreas mais distantes e menos acessíveis.

Em especial, é necessário reservar um espaço no estoque intermediário para armazenar as peças já produzidas, mas que ainda não podem ser transportadas ao cliente. Esse espaço é organizado em *pilhas*, que correspondem a um agrupamento organizado que contém apenas um determinado tipo de peça. Uma nova pilha é *aberta* assim que a primeira peça de um determinado tipo é produzida e permanece aberta até que a última cópia deste tipo de peça seja fabricada, momento em que a pilha é *fechada*. Durante esse período, idealmente, a pilha deve permanecer no ambiente de produção sem ser transportada. Isso evita atrasos e interrupções no processo produtivo, reduz os riscos associados à manipulação e elimina custos adicionais com manuseio e equipamentos necessários para tal manipulação, contribuindo, assim, para a redução dos custos de produção.

É importante ressaltar que, para atender à demanda específica de cada peça, diferentes padrões de corte precisam ser processados em uma quantidade específica de vezes. Dessa forma, o

sequenciamento desses padrões pode gerar um número variável de pilhas abertas simultaneamente, pois a ordem de produção das peças pode resultar em períodos em que várias pilhas precisam ser mantidas abertas ao mesmo tempo. Dessa forma, menos pilhas abertas simultaneamente significam menos espaço ocupado, melhorando assim o fluxo de produção e gerando um melhor aproveitamento de espaço no ambiente industrial, já que, quanto maior o número de pilhas abertas simultaneamente, mais espaço será ocupado, o que confere uma característica combinatória ao problema (Yanasse, 1997).

Levando-se em consideração que o espaço físico disponível para pilhas é limitado e que não é possível abrir pilhas simultaneamente para todos os tipos de peças, caracteriza-se o problema de minimização de pilhas abertas (ou MOSP, do inglês *minimization of open stacks problem*). Esse problema foi formalizado e definido por Yanasse (1997), com o objetivo de determinar a sequência ideal de processamento dos padrões, de modo a minimizar o número máximo de pilhas abertas simultaneamente. O MOSP é o problema de otimização em foco nesta monografia.

Para abordar o MOSP, utiliza-se o método conhecido como revenimento paralelo (*parallel tempering*, PT), que possibilita a amostragem eficiente de distribuições de probabilidade complexas. Por essa razão, o PT é bastante popular na área de simulações químicas e físicas (Earl; Deem, 2005), porém, ainda não é tão difundido na área de pesquisa operacional, o que abre espaço para novas pesquisas e implementações do método (Almeida; de Castro Lima; Carvalho, 2025). Este trabalho utiliza uma implementação paralela do PT com o objetivo de determinar boas soluções em tempo praticável em ambientes reais para o problema abordado.

## 1.1 Justificativa

No Brasil, as indústrias de transformação, aquelas que convertem substâncias e componentes em produtos novos, possuíram uma participação de 14,4% no PIB nacional no ano de 2024, de acordo com a Confederação Nacional da Indústria (2025). O MOSP está presente em diversos setores industriais, principalmente nas indústrias de transformação, como a metalurgia, a indústria têxtil e a vidreira, conforme mencionado anteriormente. Ademais, sua aplicação prática, por exemplo, em projetos de circuitos de integração em escala muito grande, evidencia a relevância do MOSP para desafios reais de engenharia, conectando-o a áreas práticas e multidisciplinares (Linhares; Yanasse, 2002).

É importante destacar que o MOSP é um problema de característica NP-difícil (Yanasse, 1997), o que implica que, até o momento, não existem algoritmos conhecidos capazes de determinar soluções ótimas em tempo viável para instâncias maiores. Por isso, o desenvolvimento e a implementação de novos algoritmos que sejam capazes de gerar boas soluções de forma eficiente tornam-se essenciais para lidar com o MOSP.

Além disso, sobre o método de solução selecionado, o PT, é importante ressaltar também

que sua implementação paralela estado da arte é bastante recente e tem apresentado bons resultados, principalmente ao abordar problemas NP-difíceis relacionados à manufatura flexível. Dessa forma, visando aproveitar uma abordagem paralela do PT, assim como realizado em Almeida, de Castro Lima e Carvalho (2025), nesta monografia utiliza-se o PT para abordar o MOSP.

## 1.2 Objetivos

De maneira geral, este trabalho tem o objetivo de aplicar o PT à solução do MOSP. Para além do objetivo geral, são os objetivos específicos deste trabalho:

- Realizar pesquisa que gere embasamento teórico e revisão bibliográfica sobre o MOSP; e
- Realizar pesquisa que gere embasamento teórico e revisão bibliográfica sobre o método PT.
- Avaliar o método utilizado a partir de problemas de teste disponíveis publicamente;
- Realizar comparativos de desempenho de acordo com a base de dados proposta em Frinhani, Carvalho e Soma (2018) e o método considerado o estado da arte.

## 1.3 Organização do Trabalho

A organização do trabalho segue a seguinte estrutura: o Capítulo 2 é dedicado a uma revisão bibliográfica do problema abordado nesta monografia. O Capítulo 3 trata da fundamentação teórica do MOSP e também do PT. Em seguida, no Capítulo 4 é exemplificada a metodologia empregada e o seu desenvolvimento, incluindo o detalhamento do método escolhido. Logo após, o Capítulo 5 relata os experimentos computacionais preliminares e os resultados obtidos. Concluindo, no Capítulo 6 é apresentada a conclusão deste estudo.

# 2 Trabalhos relacionados

A literatura sobre MOSP se inicia ao fim da década de 1970 com Madsen (1979). Entretanto, a definição do problema abordado é diferente daquela do MOSP. Especificamente, tratou-se de um problema de corte e estoque aplicado à indústria de vidros. O trabalho de Madsen (1979) serviu como inspiração para a área durante seus anos seguintes e principalmente para Yuen (1991) que, já na década de 1990, definiu o MOSP e propôs duas heurísticas práticas para sequenciamento de padrões de corte. Logo após, Yuen (1995) propôs mais quatro heurísticas para o mesmo problema. O resultado obtido da comparação geral foi que a melhor entre elas é a denominada *Yuen3*. Em um trabalho complementar, Yuen e Richardson (1995) estabeleceram critérios de otimalidade e propuseram um limite inferior trivial para o valor das soluções geradas para o MOSP, baseado no número máximo de itens distintos por padrão, além de um método exato de enumeração de permutações.

O marco teórico definitivo desta década foi a demonstração de Yanasse (1997) de que o MOSP é NP-difícil. O autor também propõe um modelo em grafos, denominado grafo MOSP, em que os nós representam padrões de corte e as arestas indicam semelhanças entre padrões, i.e., padrões que compartilham pelo menos um mesmo tipo de peça, compartilhando uma mesma pilha. A partir dessa representação, é introduzido um método *branch and bound*, que simplifica a árvore de busca ao realizar a poda de nós não promissores, reduzindo progressivamente a complexidade do problema. Além disso, o artigo estabelece limites teóricos para o número máximo de pilhas abertas, vinculando-os à estrutura dos padrões e à topologia do grafo. Yanasse, Becceneri e Soma (1999) avançaram ao utilizar a heurística de contração de arcos para derivar limites inferiores e superiores, fundamentando abordagens futuras.

No final dos anos 1990 e início dos anos 2000, a diversificação de métodos ganhou destaque. Faggioli e Bentivoglio (1998) combinaram busca tabu com técnicas exatas, enquanto Fink e Voß (1999) exploraram algoritmos genéticos e *simulated annealing*. Logo após, Linhares e Yanasse (2002) apresentaram resultados teóricos sobre o MOSP e outros problemas de sequenciamento de padrões. O estudo confirma que o MOSP é um problema NP-difícil e demonstra que ele é significativamente diferente de problemas relacionados, como o problema de minimização do espalhamento de pedidos, o problema de minimização de descontinuidades e o problema de minimização de trocas de ferramentas, tornando inviável a busca por soluções ótimas simultaneamente para todos eles.

Avanços em modelagem matemática surgiram com Yanasse e Limeira (2004), que propuseram esquemas de enumeração polinomiais. Já Becceneri, Yanasse e Soma (2004), utilizaram a heurística de contração de arcos para gerar limitantes inferiores para os valores das soluções, um dos parâmetros de comparação dos métodos nos experimentos realizados. Como resultado,

foi possível superar a heurística *Yuen3* ao desenvolver a heurística de nó de custo mínimo (ou MCNH, do inglês *minimum cost node heuristic*) em grafos MOSP, que usa como critério a menor quantidade de arcos necessários para *fechar* o nó. Um nó é definido como aberto assim que o primeiro arco incidente é percorrido e como fechado quando todos os seus arcos incidentes já foram percorridos pelo método.

Ainda nos anos 2000, o MOSP foi tema do desafio de modelagem por restrições (*Constraint Modeling Challenge* 2005). O desafio era desenvolver métodos capazes de encontrar soluções ótimas para o MOSP que, embora já tivesse sido estudado em outros trabalhos, até então não havia sido abordado com técnicas de programação por restrições (Smith; Gent, 2005).

Ressaltam-se alguns destaques da competição, como Petrie (2005) que realizou uma análise que comprovou a equivalência de uma reformulação do MOSP com uma abordagem baseada em *branch-and-bound*, além de utilizar uma implementação baseada em programação por restrições; Laborie (2005) que fez uso de uma busca local para melhorar as soluções que apresentou bom desempenho em instâncias médias e pequenas, porém não conseguiu resolver as maiores instâncias do desafio; e Prestwich (2005) que estabeleceu regras de simetria e poda do espaço de soluções para o MOSP.

O método vencedor do Challenge foi posteriormente publicado em Banda e Stuckey (2007) e se destacou por apresentar uma abordagem inovadora baseada em programação dinâmica, utilizando o algoritmo A*. Essa estratégia permitiu uma redução significativa do espaço de busca, diminuindo-o de $|P|!$ para $2^{|P|}$, o que resultou em um desempenho superior em relação aos demais competidores. Além de resolver um maior número de instâncias, o método demonstrou uma execução significativamente mais rápida, atingindo uma melhoria de até duas ordens de magnitude em termos de velocidade. Além disso, o método obteve soluções de alta qualidade, consolidando-se como a melhor abordagem avaliada. Todos os experimentos foram conduzidos nas instâncias fornecidas para o Challenge, garantindo uma comparação justa e padronizada entre os diferentes métodos.

Ao fim dos anos 2000, Ashikaga e Soma (2009) propuseram uma heurística denominada ashikaga-soma, ou somente AS, que fazia uso tanto de uma heurística gulosa baseada no grau dos vértices de grafos MOSP quanto de uma heurística de extensão rotação para obtenção de ciclos hamiltonianos. O AS superou o tradicional método *Yuen3* durante os experimentos computacionais propostos. No mesmo ano, Chu e Stuckey (2009) apresentaram um novo método exato que estende a ideia de Banda e Stuckey (2007), combinando o registro de soluções ruins com uma estratégia de *branch and bound* baseada no fechamento de pilhas, tornando-o drasticamente mais rápido do que o estado da arte anterior – cerca de 5 a 6 ordens de magnitude para instâncias altamente densas e 2 a 3 ordens de magnitude para instâncias menores. O método foi testado com as instâncias de Smith e Gent (2005). Porém, poucas execuções ultrapassavam poucos milissegundos; desta forma, foi criado também um gerador de instâncias aleatórias difíceis.

Em 2010, Yanasse; Senne publicaram uma revisão sobre propriedades do MOSP e técni-

cas de pré-processamento, como eliminação de padrões redundantes e decomposição de instâncias. Este artigo tornou-se referência para métodos posteriores, ao demonstrar como simplificar problemas complexos antes da aplicação de algoritmos. Após isso, algoritmos adaptativos e modelos baseados em grafos dominaram a pesquisa. Giovanni, Massi e Pezzella (2013) desenvolveram um algoritmo genético adaptativo para instâncias grandes, enquanto Lopes e Carvalho (2015) modelaram o MOSP via grafos de intervalo, associando intervalos de tempo a pilhas e propondo uma nova formulação em programação linear inteira.

Também em 2015, Carvalho; Soma propuseram o método $HBF_{2r}$, uma heurística baseada em busca em largura com duas regras heurísticas aplicadas ao grafo MOSP, que foi provado ser equivalente a MCNH e Chu & Stuckey em termos de tempo de execução em instâncias pequenas. Em instâncias grandes, o método exato de Chu & Stuckey não foi capaz de encontrar soluções, enquanto MCNH e $HBF_{2r}$ foram capazes de fazê-lo em tempo viável. Embora MCNH tenha apresentado maior precisão, $HBF_{2r}$ demonstrou ser mais robusto, pois seus erros de solução foram mais consistentes e menos variados em comparação a MCNH. Gonçalves, Resende e Costa (2016) propuseram uma variação de algoritmo genético com representação por chaves aleatórias viciadas (ou BRKGA, do inglês *biased random-key genetic algorithm*), alcançando soluções próximas ao ótimo quando comparadas com o método exato de Chu & Stuckey, o A* de Banda e Stuckey (2007), e a *Yuen3*.

Por último, meta-heurísticas voltaram a se tornar a tendência para solução do MOSP. Lima e Carvalho (2017) aplicaram busca em vizinhança variável para refinar soluções, alcançando bons resultados independentemente de qual vizinhança seja escolhida, produzindo soluções de boa qualidade com curto tempo de execução, sem uma grande influência do tamanho de suas instâncias. No ano seguinte, Santos e Carvalho (2018) utilizaram *adaptive large neighborhood search* para otimização do leiaute de circuitos eletrônicos. Apesar de ser testado em um menor número de instâncias, aparentou ser competitivo, alcançando 89% de soluções ótimas e melhorando a melhor solução conhecida de 14 instâncias ao ser comparado com o BRKGA e Chu & Stuckey.

O estado da arte atual é representado por Frinhani, Carvalho e Soma (2018), que adaptou a heurística *PageRank*, para redes complexas, ao grafo MOSP, além de propor instâncias maiores para o problema. O método proposto, *PieceRank*, superou todas as abordagens anteriores para o MOSP em termos de qualidade de soluções no menor tempo de execução, incluindo as heurísticas *Yuen3*, MCNH, Chu & Stuckey e $HBF_{2r}$, visto que essas se provaram inviáveis nas novas instâncias, que são cerca de 5 vezes maiores que as anteriores. Estabeleceu-se, portanto, uma nova referência para a resolução do MOSP até os dias de hoje. Destaca-se que o trabalho mencionado justifica a necessidade de instâncias maiores devido à grande variedade de configurações de veículos e de peças na indústria automobilística brasileira. Este método e as instâncias propostas são considerados nos experimentos computacionais e na análise comparativa dos resultados desta monografia.

# 3 Fundamentação teórica

Neste capítulo, apresenta-se a fundamentação teórica necessária para o entendimento do problema abordado nesta monografia, bem como da abordagem computacional adotada. Dessa forma, são detalhados o MOSP e os princípios gerais do funcionamento do PT.

## 3.1 O problema de minimização de pilhas abertas

O MOSP pode ser definido formalmente da seguinte maneira: considere $M$ como o número total de peças a serem fabricadas e $N$ como o número total de padrões de corte disponíveis. O conjunto de padrões de corte é representado por $\mathcal{P}_a = \{P_{a_1}, P_{a_2}, \ldots, P_{a_N}\}$, em que cada padrão $P_{a_i}$ produz peças do conjunto $\mathcal{P}_e = \{P_{e_1}, P_{e_2}, \ldots, P_{e_M}\}$.

Cada peça está associada a uma pilha que é aberta no momento em que a produção dessa peça é iniciada e somente fechada quando todas as unidades necessárias dessa peça são produzidas. A relação entre os padrões de corte e as peças pode ser representada por uma matriz binária de incidência $A \in \{0,1\}^{M \times N}$, em que:

$$a_{ij} = \begin{cases} 1, & \text{se a peça } P_{e_i} \text{ é produzida pelo padrão } P_{a_j}, \\ 0, & \text{caso contrário.} \end{cases} \tag{3.1}$$

A ordem de processamento dos padrões é representada por uma permutação $\pi$ das $N$ colunas da matriz $A$, em que $\pi_k$ denota a coluna posicionada no $k$-ésimo estágio de produção. Isso significa que a $k$-ésima coluna da matriz é dada por $[a_{1,\pi_k}, a_{2,\pi_k}, \ldots, a_{M,\pi_k}]^\top$. É importante ressaltar que, em um único estágio, o padrão é processado quantas vezes for necessário, de forma a não retornar à linha de produção em estágios futuros. Além disso, uma pilha se torna *ociosa* caso o próximo padrão não produza a peça que estava sendo produzida.

Uma pilha associada a uma peça $P_{e_i} \in \mathcal{P}_e$ está aberta no estágio $k$ da sequência $\pi$ se,

$$\sum_{j=1}^{k} a_{i,\pi_j} > 0 \quad \text{e} \quad \sum_{j=k}^{N} a_{i,\pi_j} > 0, \tag{3.2}$$

em que $\sum_{j=1}^{k} a_{i,\pi_j} > 0$ indica que a produção da peça $P_{e_i}$ foi iniciada antes ou no estágio $k$, e $\sum_{j=k}^{N} a_{i,\pi_j} > 0$ indica que a produção de $P_{e_i}$ será finalizada no estágio $k$ ou posteriormente. O número de pilhas abertas no estágio $k$ é, portanto, dado por,

$$O(k) = \left| \{P_{e_i} \in \mathcal{P}_e : \sum_{j=1}^{k} a_{i,\pi_j} > 0 \text{ e } \sum_{j=k}^{N} a_{i,\pi_j} > 0\} \right|. \tag{3.3}$$

O MOSP é um problema NP-difícil cujo objetivo é determinar a permutação $\pi^*$, dentre todas as $N!$ possíveis, que minimize o número máximo de pilhas abertas simultaneamente ao longo do processamento, conforme apresentado na Equação 3.4.

$$\min \max_{1 \leq k \leq N} O(k). \tag{3.4}$$

Uma instância do MOSP pode ser descrita por meio de matrizes. A Tabela 3.1 representa a relação entre os tipos de peças $P_{e_1}, P_{e_2}, P_{e_3}, P_{e_4}, P_{e_5}, P_{e_6}$ e os padrões $P_{a_1}, P_{a_2}, P_{a_3}, P_{a_4}\ P_{a_5}, P_{a_6}$. Neste caso, o padrão $P_{a_1}$ produz as peças $P_{e_1}$ e $P_{e_2}$ (primeira e segunda linhas da coluna $P_{a_1}$), e assim por diante.

Tabela 3.1 – Matriz de incidência ($A$).

| Peça/Padrão | $P_{a_1}$ | $P_{a_2}$ | $P_{a_3}$ | $P_{a_4}$ | $P_{a_5}$ | $P_{a_6}$ |
|---|---|---|---|---|---|---|
| $P_{e_1}$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $P_{e_2}$ | 1 | 0 | 0 | 1 | 1 | 0 |
| $P_{e_3}$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $P_{e_4}$ | 0 | 1 | 0 | 0 | 0 | 1 |
| $P_{e_5}$ | 0 | 0 | 1 | 0 | 0 | 1 |
| $P_{e_6}$ | 0 | 1 | 0 | 0 | 1 | 0 |

A partir da matriz $A$, é possível gerar uma nova matriz $B$ que representa as pilhas abertas ao longo dos estágios de produção. A matriz $B$ é gerada conforme as seguintes regras. Cada linha de $B$ corresponde a uma pilha $P_{e_1}, P_{e_2}, P_{e_3}, P_{e_4}, P_{e_5}, P_{e_6}$, e cada coluna corresponde a um estágio de processamento. Dessa forma, essa relação pode ser representada por $B \in \{0,1\}^{M \times N}$, em que:

$$b_{ij} = \begin{cases} 1, & \text{se } P_{e_i} \text{ foi produzida a partir de algum padrão no estágio } j \textit{ ou} \\ & \text{se } P_{e_i} \text{ foi produzida a partir de algum padrão até o estágio } j \\ & \textit{e ainda será produzida novamente em algum estágio posterior} \\ & \text{(condição para cobrir pilhas ociosas),} \\ 0, & \text{caso contrário.} \end{cases} \tag{3.5}$$

Com base na permutação $\pi_1 = [P_{a_1}, P_{a_2}, P_{a_3}, P_{a_4}, P_{a_5}, P_{a_6}]$, a matriz $B^{\pi_1}$ resultante é exibida na Tabela 3.2. Na matriz $B^{\pi_1}$, é possível observar que nos estágios $j = 3, 4$, todas as pilhas estão abertas, e que no estágio $j = 2$ as pilhas $P_{e_1}$ e $P_{e_2}$ estão abertas, porém, de maneira ociosa, pois o padrão processado nesse estágio não produz as peças $P_{e_1}$ e $P_{e_2}$. O número máximo de pilhas abertas pode ser calculado somando os valores em cada coluna da matriz $B^{\pi_1}$ e encontrando o valor máximo. Para este exemplo, as somas de cada uma das colunas de $B^{\pi_1}$ são [2, 5, 6, 6, 5, 3], respectivamente. Portanto, o número máximo de pilhas abertas é 6.

Tabela 3.2 – Matriz de pilhas abertas ($B^{\pi_1}$).

| Pilha/Estágio | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $P_{e_1}$ | 1 | 1 | 1 | 1 | 0 | 0 |
| $P_{e_2}$ | 1 | 1 | 1 | 1 | 1 | 0 |
| $P_{e_3}$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $P_{e_4}$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $P_{e_5}$ | 0 | 0 | 1 | 1 | 1 | 1 |
| $P_{e_6}$ | 0 | 1 | 1 | 1 | 1 | 0 |

Com base na permutação $\pi_2 = [P_{a_3}, P_{a_6}, P_{a_2}, P_{a_5}, P_{a_4}, P_{a_1}]$, a matriz $B^{\pi_2}$ resultante é obtida visando otimizar a distribuição dos padrões de produção ao longo dos estágios, conforme apresentado na Tabela 3.3.

Tabela 3.3 – Matriz de pilhas abertas ($B^{\pi_2}$).

| Pilha/Estágio | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $P_{e_1}$ | 0 | 0 | 0 | 0 | 1 | 1 |
| $P_{e_2}$ | 0 | 0 | 0 | 1 | 1 | 1 |
| $P_{e_3}$ | 1 | 1 | 1 | 1 | 1 | 0 |
| $P_{e_4}$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $P_{e_5}$ | 1 | 1 | 0 | 0 | 0 | 0 |
| $P_{e_6}$ | 0 | 0 | 1 | 1 | 0 | 0 |

Na nova matriz $B^{\pi_2}$, é possível observar que, a partir do estágio $j = 2$ até o estágio $j = 5$, o número de pilhas abertas permanece constante, embora as pilhas abertas variem. Para este exemplo, as somas de cada uma das colunas são [2, 3, 3, 3, 3, 2], respectivamente. Portanto, o número máximo de pilhas abertas é 3.

Como ilustrado nos exemplos, uma permutação adequada dos padrões pode reduzir significativamente o número máximo de pilhas abertas, otimizando assim o uso de recursos e a eficiência do processo de produção. No entanto, o MOSP é um problema NP-difícil (Yanasse, 1997), o que implica que a busca por uma solução ótima torna-se computacionalmente inviável à medida que o número de padrões e peças aumenta. Essa complexidade justifica a importância de desenvolver métodos heurísticos e meta-heurísticos para abordar o problema de forma eficiente em cenários práticos.

## 3.2 Revenimento paralelo

O PT é um método amplamente utilizado para a amostragem eficiente de distribuições de probabilidade complexas. Também conhecido como *replica exchange* e originário da física estatística, sua ideia inicial foi proposta por Swendsen e Wang (1986), que sugeriram a simulação de múltiplas réplicas de um sistema em diferentes temperaturas com trocas parciais de informações.

Posteriormente, Geyer et al. (1991) aprimoraram essa abordagem, introduzindo a troca completa de dados entre réplicas, estabelecendo a base teórica do método. De forma independente, Kimura e Taki (1991) adaptaram conceitos semelhantes no contexto da pesquisa operacional, dando origem ao *temperature parallel simulated annealing* (TPSA), um precursor direto das técnicas do PT aplicadas fora do domínio tradicional da física.

Para se entender o funcionamento do PT é necessário voltar atrás alguns passos e compreender alguns conceitos estruturantes. Dentre eles, o *método de Monte Carlo*, uma técnica baseada em amostragem aleatória utilizada para resolver problemas complexos que seriam inviáveis por abordagens determinísticas. Surgido como uma ferramenta na física estatística, o método permite estimar valores aproximados para distribuições cuja solução exata é de difícil obtenção e, à medida que o número de amostras aumenta, os resultados apresentam tendência à convergência, indo em direção ao ótimo (Metropolis et al., 1953). Na área de otimização, sua aplicação se dá pela simulação de diversas possíveis soluções de problemas combinatórios, sendo sua eficiência diretamente relacionada à quantidade de soluções analisadas.

Visando melhorar a eficiência do método de Monte Carlo, foi desenvolvido o método que combina Monte Carlo e cadeias de Markov (ou MCMC, do inglês *Markov chain Monte Carlo*) que introduz a dependência entre as amostras geradas para acelerar a convergência e melhorar a exploração do espaço de amostragem. Diferentemente do método de Monte Carlo tradicional, no qual cada amostra é gerada de forma independente, o MCMC constrói uma sequência de estados interligados, em que cada novo estado é obtido a partir de uma modificação do estado anterior seguindo regras probabilísticas. Esse processo é guiado por uma cadeia de Markov, garantindo que, após um número de iterações, a distribuição amostral converge para a distribuição de interesse. Dessa forma, o MCMC se torna particularmente útil em problemas de alta dimensionalidade, nos quais a amostragem direta é inviável.

De maneira análoga à abordagem estatística do MCMC, no contexto de otimização cada amostra pode ser interpretada como uma solução, e cada transição entre estados corresponde a uma modificação ou melhoria na solução atual – realizada ao gerar uma solução vizinha e aceitá-la com base em um critério probabilístico. Assim como o MCMC constrói uma cadeia de estados interligados que, com o tempo, converge para uma distribuição de interesse. Esse processo permite explorar o espaço de soluções, amostrando progressivamente regiões com melhores resultados. Nesse paralelo, os termos “estado” e “transição” correspondem, respectivamente, à solução atual e à geração de uma solução vizinha, enquanto a “distribuição de interesse” remete à avaliação que orienta a busca pela solução ótima.

Dentro dessa abordagem, um dos algoritmos mais conhecidos é o *algoritmo de Metropolis*. O algoritmo segue um processo iterativo no qual um novo estado $s'$ é proposto com base no estado atual $s$, sendo sua aceitação determinada por um critério probabilístico. Esse critério é baseado na *distribuição de Boltzmann*, que define a probabilidade de aceitação do novo estado levando em conta a diferença de energia entre os estados. Nesse contexto, o termo energia refere-se a uma

medida numérica que quantifica a qualidade ou o custo associado a um estado – em sistemas físicos, está relacionada à estabilidade. A distribuição de Boltzmann, que define a probabilidade de aceitação do novo estado considerando a diferença de energia entre os estados, é expressa pela Equação 3.6.

$$P(E,T) = e^{-\frac{(\Delta E)}{T}} \tag{3.6}$$

Em que:

- $\Delta E = f(s') - f(s)$ representa a variação da energia entre o novo estado e o atual,
- $T$ é o parâmetro de temperatura que regula a aceitação de estados com maior nível de energia.

Se o novo estado possui um menor nível de energia, ele é sempre aceito. Caso contrário, ele ainda pode ser aceito com uma probabilidade que depende de $\Delta E$ e da temperatura $T$, conforme a Equação 3.6 demonstra. Para um valor de $T$ alto, há maior probabilidade de aceitar estados com maior energia, permitindo uma ampla exploração do espaço de busca. Para um valor de $T$ baixo, reduz-se a aceitação de estados menos favoráveis e favorece-se uma busca mais refinada em regiões promissoras. O valor da temperatura é um mecanismo essencial para o balanceamento entre exploração e intensificação da amostragem.

Embora o algoritmo de Metropolis represente um avanço sobre os métodos tradicionais de Monte Carlo, ele pode se estagnar ao lidar com funções que possuem múltiplos ótimos locais separados por barreiras de energia, o que torna a exploração do espaço amostral ineficiente em alguns cenários. Numa tentativa de amenizar essa limitação, o *algoritmo de Metropolis-Hastings* foi desenvolvido como um MCMC, introduzindo um mecanismo mais flexível para a seleção de estados propostos. Esse mecanismo permite a utilização de *distribuições de proposta* arbitrárias – não restritas a serem simétricas – de modo que a escolha dos novos estados possa ser adaptada às características específicas da distribuição.

Diferentemente do algoritmo de Metropolis original, que impõe distribuições simétricas para as transições, essa flexibilidade acelera a convergência e melhora a capacidade de explorar espaços com múltiplos ótimos (Hastings, 1970), embora ainda possua limitações do mesmo tipo. De maneira análoga à abordagem estatística do algoritmo de Metropolis e do algoritmo Metropolis-Hastings, no contexto de otimização, a escolha de aceitação de soluções segue a mesma lógica apresentada anteriormente pela Equação 3.6, essencial para equilibrar a exploração de diferentes áreas do espaço de busca e a intensificação da busca por ótimos globais.

O método PT funciona criando várias cópias do sistema simulado, ou seja, coordenando várias constantes $T$ do algoritmo de Metropolis. Dessa forma, são criadas $k$ réplicas do sistema ($R = \{r_1, r_2, \ldots, r_k\}$) em que cada réplica $r_i$ possui uma temperatura associada $T_i$ e um estado

inicial. Assim, cada réplica em uma temperatura fixa executa um MCMC de maneira independente e, após esse processo, são propostas *trocas de réplicas entre temperaturas adjacentes* $r_i$ e $r_j$. A probabilidade de troca é definida pela Equação 3.7.

$$P(r_i \rightarrow r_j) = \min\left[1, \exp\left(\Delta\beta\Delta E\right)\right] \tag{3.7}$$

Na Equação 3.7, $\Delta\beta$ pode ser expandido como

$$\Delta\beta = \frac{1}{T_j} - \frac{1}{T_i},$$

com $T_i$ e $T_j$ correspondentes às temperaturas de cada réplica $r_i$ e $r_j$ envolvidas na proposta de troca. Ainda, $\Delta E$ indica a diferença entre o valor da energia em cada réplica, ou seja,

$$\Delta E = f(r_j) - f(r_i),$$

em que $f(r)$ representa a qualidade do estado na réplica correspondente.

Se a troca for aceita, a réplica $r_i$ passa a considerar o valor da temperatura $T_j$ e a réplica $r_j$ passa a considerar o valor da temperatura $T_i$. Caso contrário, as temperaturas permanecem inalteradas. Esse mecanismo permite que o sistema explore o espaço amostral de forma eficiente, equilibrando a diversificação, em temperaturas altas – em que estados menos favoráveis têm maior probabilidade de serem aceitos, expandindo a busca – com a intensificação, em temperaturas baixas – em que estados piores são aceitos mais raramente, refinando a busca em torno dos melhores estados encontrados.

A Figura 3.1 ilustra o processo de troca de temperaturas no algoritmo PT. As setas diagonais representam as propostas de troca entre réplicas adjacentes aceitas, enquanto as setas horizontais indicam a manutenção da temperatura ao longo da simulação, ou seja, propostas de trocas não aceitas. Cada réplica segue uma trajetória no espaço de temperaturas fixas, potencialmente alternando entre fases de maior e menor diversificação e exploração, conforme descrito anteriormente. Esse mecanismo reduz o tempo de autocorrelação do algoritmo, permitindo que as réplicas superem barreiras de energia e escapem de mínimos locais. Como resultado, o sistema consegue explorar o espaço amostral de maneira mais eficiente.

Assim como o Algoritmo de Metropolis, o PT também pode ser interpretado como um algoritmo de otimização. Dessa forma, a troca de temperaturas entre réplicas, definida de maneira probabilística com base na diferença das qualidades das soluções, promove um equilíbrio entre diversificação e intensificação da busca, facilitando a superação de mínimos locais e a aproximação de ótimos globais. Para uma compreensão mais detalhada do funcionamento básico do PT, o Algoritmo 1 fornece uma ilustração mais abrangente do processo. Como entrada, são fornecidos um vetor de temperaturas $T$, um vetor de soluções $S$, um número inteiro $qT_{emp}$ que representa a quantidade de trocas de temperatura e um inteiro $N_{kmax}$ que define o comprimento da cadeia de Markov utilizada pelo algoritmo de Metropolis.

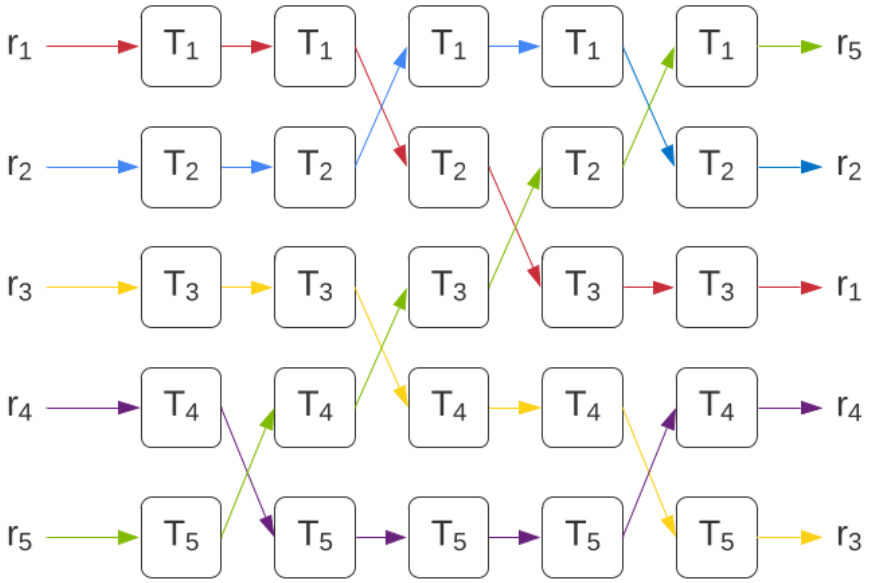


Figura 3.1 – Diagrama das trocas de temperaturas do algoritmo PT. Extraído de Almeida, de Castro Lima e Carvalho (2025).

O laço de repetição principal, entre as linhas 1 e 20, é executado até que o critério de parada seja atendido. Dentro desse laço, o algoritmo percorre todas as réplicas entre as linhas 3 e 16. Para cada réplica, o algoritmo inicializa sua solução e temperatura, e o método de Metropolis é executado dentro do laço de repetição entre as linhas 7 e 14. Durante essa etapa, novas soluções vizinhas são geradas e avaliadas. Se uma nova solução apresentar um valor de energia menor ou for aceita com base no critério probabilístico, ela substitui a solução atual da réplica na linha 11. Em seguida, verifica-se se essa nova solução é a melhor encontrada até o momento; caso positivo, ela é armazenada na linha 13.

Após a execução do algoritmo de Metropolis para todas as réplicas, o algoritmo realiza a etapa de trocas de réplicas entre temperaturas. Essa etapa, especificamente entre as linhas 17 e 20, avalia a possibilidade de troca entre réplicas adjacentes com base na diferença de energia ajustada pelo fator de temperatura. A aceitação da troca segue uma condição determinística (melhoria da solução) ou probabilística, seguindo a Equação 3.6 (linha 18). Por fim, o resultado final do algoritmo corresponde à melhor solução encontrada durante sua execução, armazenada em $S^*$.

Apesar de sua eficácia em simulação, o PT teve um emprego tardio e restrito em problemas de otimização dentro da pesquisa operacional. Apenas alguns estudos exploraram seu potencial nessa área. Yamamoto e Hashimoto (2000) aplicaram o TPSA a um problema de gerenciamento de combustível em reatores, obtendo resultados promissores frente ao SA (*simulated annealing*). Posteriormente, Miki et al. (2002) desenvolveram uma versão do TPSA para problemas contínuos, enquanto Miki et al. (2003) introduziram um algoritmo genético para ajustar dinamicamente a temperatura no TPSA. Em um estudo mais detalhado, Jun e Mizuta (2005) analisaram uma

**Algoritmo 3.1:** Algoritmo básico PT

**Entrada:** Vetor de temperaturas $T$, vetor de soluções $S$, quantidade de trocas de temperaturas $qT_{emp}$ e comprimento da cadeia de Markov $N_{kmax}$.
**Saída:** Melhor solução encontrada $S^*$.

**enquanto** *critério de parada não atendido* **faça**
  $indice\_replica \leftarrow 0$;
  **enquanto** $indice\_replica < tamanho(T) - 1$ **faça**
    $s \leftarrow S[indice\_replica]$;
    $t \leftarrow T[indice\_replica]$;
    $k \leftarrow 0$;
    **enquanto** $k \leq N_{kmax}$ **faça**
      $s' \leftarrow vizinho(s)$;
      $\Delta E \leftarrow f(s') - f(s)$;
      **se** $\Delta E \leq 0$ ***ou*** *aceita(s')* **então**
        $s \leftarrow s'$;
      **se** $s$ *é melhor que* $S^*$ **então**
        $S^* \leftarrow s$;
      $k \leftarrow k + 1$;
    $S[indice\_replica] \leftarrow s$;
    $indice\_replica \leftarrow indice\_replica + 1$;
  **para** $indice\_replica \leftarrow 1$ ***até*** $tamanho(T)$ **faça**
    $\Delta\beta\Delta E \leftarrow \left(\frac{1}{T[indice\_replica-1]} - \frac{1}{T[indice\_replica]}\right) \times (f(S[indice\_replica - 1]) - f(S[indice\_replica]))$;
    **se** $\Delta\beta\Delta E \geq 0$ ***ou*** *aceita troca(indice_replica, indice_replica-1)* **então**
      $troca(S[indice\_replica], S[indice\_replica - 1])$;

versão modificada do TPSA aplicada ao problema do caixeiro viajante com 33 cidades na China, investigando o impacto dos parâmetros em relação ao SA.

Outros estudos buscaram aprimorar o PT e sua aplicação em otimização. Wang et al. (2009) propuseram uma abordagem híbrida entre PT e SA para o problema do caixeiro viajante, utilizando múltiplas réplicas com temperaturas iniciais diferentes. No contexto da tomografia hiperespectral, Cai, Ewing e Ma (2011) estudaram a determinação de temperaturas no PT com base na *temperatura crítica*, otimizando a exploração do espaço de busca. Além disso, Kataoka et al. (2012) modificaram o TPSA e o aplicaram ao problema de empacotamento de retângulos, enquanto Zhu, Fang e Katzgraber (2016) desenvolveram o algoritmo borealis – inspirado no PT combinado com o algoritmo de *cluster* isoenergético – para otimização de problemas NP com variáveis booleanas. Esses estudos, apesar de pontuais, demonstram o potencial do PT como alternativa para problemas complexos de otimização.

Embora o método possua forte afinidade com arquiteturas paralelas – sendo amplamente utilizado para a amostragem eficiente de distribuições de probabilidade complexas e adaptando-se

bem a plataformas multiprocessador modernas, potencializando o paralelismo e acelerando a busca por soluções (Swendsen; Wang, 1986), a literatura apresenta poucas implementações que exploram esse potencial. Mesmo sendo um algoritmo intrinsecamente paralelo, a maioria das implementações reportadas é sequencial, o que limita a plena utilização de seus benefícios. Entretanto, nesta monografia, será empregada uma implementação paralela do PT, ampliando a eficiência e a escalabilidade na resolução do MOSP.

Almeida, de Castro Lima e Carvalho (2025) revisitaram o PT, explorando seu potencial em problemas de otimização. O estudo apresenta uma nova implementação paralela baseada em CPU para o PT e a avalia em três estudos de caso envolvendo problemas desafiadores de pesquisa operacional. Os resultados demonstram um desempenho significativo em relação aos métodos considerados estado da arte em cada estudo de caso, com uma redução notável no tempo de execução e a obtenção de novas soluções aprimoradas. Além disso, o artigo propõe uma API contendo a implementação paralela *multi-core* do PT para facilitar seu uso em futuras implementações, beneficiando tanto a área de pesquisa operacional quanto a de simulação.

A implementação paralela mencionada utiliza o modelo de programação paralela baseado em fluxo de dados (*dataflow*). Na implementação paralela adotada, o laço de repetição que processa as réplicas (linhas 3 a 16 do Algoritmo 1) é executado de forma concorrente, explorando o potencial das arquiteturas multiprocessadas. Embora a execução paralela em CPU não seja uma abordagem comum para o PT, Almeida, de Castro Lima e Carvalho (2025) demonstraram seu potencial em problemas de otimização complexos.

Nesta monografia, é adotada a implementação paralela disponibilizada por meio da API mencionada acima. Apesar da API fornecer uma base estruturada para a execução do PT, ainda é necessário configurar elementos específicos do problema e componentes auxiliares que garantam o funcionamento adequado do algoritmo. É importante ressaltar que, embora o PT seja um método generalista, sua eficácia depende do acoplamento adequado entre o método e o problema abordado. Isso se dá pela personalização dos componentes definidos pela API, que devem ser ajustados conforme as particularidades do problema em questão. Dentre esses componentes, específicos para cada problema tratado, destacam-se:

**Codificação:** Estrutura de dados responsável por representar a solução para o problema;

**Decodificação:** Função que converte a solução codificada em uma solução completa e válida para o problema;

**Solução inicial:** Solução de partida para cada réplica do PT, seja gerada de forma aleatória ou com base em um critério estruturado;

**Soluções vizinhas:** Componente responsável por transformar uma solução $s$ em uma solução $s'$, introduzindo pequenas modificações que possibilitam a exploração do espaço de busca;

**Função de avaliação:** Função que recebe uma solução codificada e retorna um valor numérico que indica a sua qualidade;

**Critério de parada:** Condição que determina quando o algoritmo deve encerrar a busca. Pode ser baseada em um número máximo de iterações, na obtenção de uma solução com qualidade satisfatória; em um limite de tempo ou em uma combinação lógica de diferentes condições.

Além da definição de componentes, foram realizados experimentos preliminares para ajustar os parâmetros do PT, que influenciam diretamente o desempenho do método. Esse ajuste de parâmetros foi feito utilizando o *iRace*, uma ferramenta de configuração automática de algoritmos baseada em procedimentos iterativos de seleção dos melhores parâmetros por meio de competições entre configurações (López-Ibáñez et al., 2016). Entre os principais parâmetros configurados, incluem-se:

**Temperaturas inicial e final:** Definem os limites do intervalo de temperaturas a ser distribuído entre as réplicas.

**Número de réplicas do algoritmo:** Determina quantas réplicas serão processadas em diferentes temperaturas;

**Comprimento das cadeias de Markov:** Define quantas soluções vizinhas serão geradas em cada réplica;

**Quantidade de tentativas de troca entre réplicas:** Controla o número de vezes em que trocas de temperaturas são propostas para réplicas vizinhas;

**Estratégia de distribuição inicial das temperaturas:** Define como as temperaturas do intervalo são distribuídas entre as réplicas;

**Método para atualização automática das temperaturas durante a execução:** Determina de que maneira as temperaturas das réplicas podem ser ajustadas dinamicamente durante a execução do algoritmo;

**Taxa de ajuste das temperaturas:** Define o intervalo de ajuste das temperaturas quando um método de atualização automática é utilizado.

No capítulo seguinte é detalhado o processo de adaptação do PT para a abordagem do MOSP. Isso envolve a definição e implementação dos componentes e a definição dos parâmetros, descritos anteriormente, essenciais para a execução eficiente do algoritmo.

# 4 Desenvolvimento

Meta-heurísticas são inerentemente generalistas, oferecendo abordagens flexíveis e aplicáveis a uma ampla variedade de problemas de otimização, exigindo modelagens específicas para cada cenário. O PT pode ser considerado uma meta-heurística; portanto, essa característica permite que a técnica encontre soluções de alta qualidade mesmo em problemas complexos e com restrições diversificadas. Neste capítulo, são apresentadas as adaptações realizadas na API para integrá-la ao MOSP, detalhando as modificações necessárias para o correto acoplamento dos componentes.

## 4.1 Codificação e decodificação

Uma solução do MOSP pode ser abordada a partir de diferentes estratégias de modelagem e resolução, sendo as principais o *sequenciamento de padrões*, em que se estabelece explicitamente a permutação dos padrões a serem processados, ou o *sequenciamento de peças*, abordagem na qual se determina uma ordem de prioridade para as peças, a partir da qual é derivada uma sequência de padrões compatível. Ambas as abordagens visam minimizar o número máximo de pilhas abertas simultaneamente durante o processo produtivo, mas partem de perspectivas distintas sobre o problema.

### 4.1.1 Sequenciamento de padrões

Neste tipo de abordagem, o objetivo é encontrar uma sequência de *padrões* $\mathcal{P}_a = \{P_{a_1}, P_{a_2}, \ldots, P_{a_N}\}$ a fim de minimizar o número máximo de pilhas abertas simultaneamente durante a produção. A solução do problema consiste, portanto, em uma permutação $\pi$ das colunas da matriz de incidência $A$, definindo a ordem em que os padrões serão executados.

Como exemplo, considere a permutação $\pi = [3, 6, 2, 5, 4, 1]$, representada na Figura 4.1. Essa permutação indica que o padrão $3$ é processado primeiro, seguido pelo padrão $6$, depois o padrão $2$, e assim por diante.

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 3 | 6 | 2 | 5 | 4 | 1 |

Figura 4.1 – Exemplo de solução codificada com base no sequenciamento de padrões.

A decodificação consiste em transformar o arranjo com a permutação de padrões na matriz de incidência correspondente, como mostrado na Tabela 4.1, em que cada coluna representa

um padrão e cada linha representa uma peça. O valor 1 indica que o padrão produz a peça correspondente; caso contrário, o valor é 0. A permutação define a ordem de leitura das colunas.

Tabela 4.1 – Matriz de incidência gerada a partir da permutação de padrões.

| Peça/Padrão | $P_{a_3}$ | $P_{a_6}$ | $P_{a_2}$ | $P_{a_5}$ | $P_{a_4}$ | $P_{a_1}$ |
|---|---|---|---|---|---|---|
| $P_{e_1}$ | 0 | 0 | 0 | 0 | 1 | 1 |
| $P_{e_2}$ | 0 | 0 | 0 | 1 | 1 | 1 |
| $P_{e_3}$ | 1 | 1 | 1 | 1 | 1 | 0 |
| $P_{e_4}$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $P_{e_5}$ | 1 | 1 | 0 | 0 | 0 | 0 |
| $P_{e_6}$ | 0 | 0 | 1 | 1 | 0 | 0 |

Do ponto de vista computacional, o sequenciamento de padrões, por ser uma manipulação indireta das pilhas, oferece vantagens importantes. Como cada movimento no espaço de busca pode ser definido como uma troca entre dois padrões, ou uma inserção de um padrão em uma nova posição, a definição de vizinhanças é natural e de fácil implementação. Além disso, a avaliação de uma solução pode ser feita de forma eficiente ao percorrer a sequência de padrões, atualizando as pilhas abertas com vetores auxiliares que monitoram, para cada peça, a quantidade acumulada de vezes em que ela foi produzida.

Contudo, essa simplicidade estrutural pode levar a desafios. Alterações pontuais na ordem dos padrões podem ter impactos complexos no comportamento das pilhas, dificultando a previsão dos efeitos de uma modificação local. Isso pode resultar em soluções presas em ótimos locais, especialmente em instâncias com forte sobreposição de peças entre padrões.

### 4.1.2 Sequenciamento de peças

O sequenciamento de peças, proposto inicialmente por Becceneri, Yanasse e Soma (2004), é uma abordagem frequente na literatura, sendo usado inclusive pelo método *PieceRank* proposto por Frinhani, Carvalho e Soma (2018). Ao contrário da abordagem anterior, o sequenciamento de peças busca definir uma ordem de prioridade entre as *peças* $\mathcal{P}_e = \{P_{e_1}, P_{e_2}, \ldots, P_{e_M}\}$, com base na qual é construída uma sequência de padrões compatível. A solução consiste, portanto, em uma permutação $\pi$ das linhas da matriz $A$, definindo a ordem em que as peças devem ser atendidas.

A sequência de padrões é construída a partir dessa permutação de peças por meio de um procedimento de varredura: percorre-se a lista de peças na ordem definida, e, a cada nova peça analisada, identificam-se todos os padrões que a produzem. Cada vez que um padrão tiver todas as suas peças já analisadas, ele é inserido na sequência final. Esse processo continua até que todos os padrões sejam posicionados.

Como exemplo, considere a permutação de peças $\pi = [3, 5, 4, 6, 2, 1]$, representada na Figura 4.2. Essa permutação indica que a peça 3 é a primeira a ser produzida, seguida pela peça

5, depois a peça 4, e assim por diante.

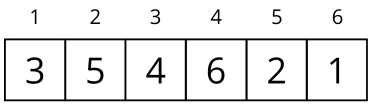


Figura 4.2 – Exemplo de solução codificada com base no sequenciamento de peças.

Durante a varredura, após a análise das peças 3 e 5, o padrão 3 pode ser inserido na solução. Ao incluir a peça 4, o padrão 6 também passa a estar disponível para o sequenciamento. Ao final do processo, obtém-se a mesma sequência de padrões $\pi = [3, 6, 2, 5, 4, 1]$, mas agora derivada indiretamente da ordem das peças priorizadas.

Tabela 4.2 – Matriz de incidência gerada a partir da permutação de peças.

| Peça/Padrão | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $P_{e_3}$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $P_{e_5}$ | 0 | 0 | 1 | 0 | 0 | 1 |
| $P_{e_4}$ | 0 | 1 | 0 | 0 | 0 | 1 |
| $P_{e_6}$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $P_{e_2}$ | 1 | 0 | 0 | 1 | 1 | 0 |
| $P_{e_1}$ | 1 | 0 | 0 | 1 | 0 | 0 |

Essa abordagem permite uma maior precisão na manipulação de pilhas por ter um controle direto na abertura e fechamento delas, também sendo vantajosa em contextos onde o número de peças $M$ é significativamente menor que o número de padrões $N$, reduzindo o espaço de busca. Além disso, pequenas alterações na permutação de peças podem induzir mudanças mais amplas na sequência de padrões, facilitando a exploração do espaço de busca e a fuga de ótimos locais. Em contrapartida, a avaliação de uma permutação de peças é mais custosa. A cada iteração, é necessário reconstruir a sequência de padrões indiretamente, o que demanda verificações adicionais para garantir a viabilidade e a cobertura das demandas.

## 4.2 Solução inicial

A solução inicial em uma primeira abordagem é gerada de maneira aleatória, ou seja, uma permutação aleatória da sequência de padrões de uma instância. Dessa forma, cada execução tende a iniciar de um ponto diferente no espaço de busca, favorecendo a diversidade das soluções e aumentando as chances de encontrar resultados de melhor qualidade.

Além da abordagem aleatória, também é utilizado o método MCNH, visando gerar soluções iniciais mais promissoras. Nesse contexto, adota-se uma estratégia mista: na qual algumas soluções utilizam o MCNH e outras são totalmente aleatórias. Essa combinação de abordagens — aleatória e heurística — permite balancear a exploração ampla do espaço de busca

com uma exploração informada, potencializando o desempenho do algoritmo PT na obtenção de boas soluções.

## 4.3 Função de avaliação

A função de avaliação tem como objetivo quantificar a qualidade de uma solução baseada na permutação dos padrões de corte. Essa avaliação é feita por meio da interpretação da matriz de incidência, para gerar a matriz de pilhas abertas. A matriz de incidência representa a relação entre os tipos de peças e os padrões. Já a matriz de pilhas abertas permite calcular a métrica de avaliação da solução, que é o número máximo de pilhas abertas em qualquer estágio do processamento. Isso é feito somando os valores de cada coluna da matriz de pilhas abertas apresentada na Tabela 4.3 e identificando o valor máximo obtido.

Tabela 4.3 – Matriz de pilhas abertas.

| Pilha/Estágio | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $P_{e_1}$ | 0 | 0 | 0 | 0 | 1 | 1 |
| $P_{e_2}$ | 0 | 0 | 0 | 1 | 1 | 1 |
| $P_{e_3}$ | 1 | 1 | 1 | 1 | 1 | 0 |
| $P_{e_4}$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $P_{e_5}$ | 1 | 1 | 0 | 0 | 0 | 0 |
| $P_{e_6}$ | 0 | 0 | 1 | 1 | 0 | 0 |

A matriz de pilhas abertas apresentada na Tabela 4.3 é exatamente igual à matriz de incidência apresentada na Tabela 4.1, isso acontece porque na matriz de incidência ordena-se os padrões na ordem em que são processados e porque não existem pilhas ociosas durante o processamento dos padrões. Para este exemplo, as somas de cada uma das colunas são [2, 3, 3, 3, 3, 2], respectivamente. Portanto, o número máximo de pilhas abertas é 3.

## 4.4 Estruturas de vizinhança

A definição de estruturas de vizinhança é um aspecto fundamental para a eficiência do PT, pois determina como a busca se movimenta no espaço de soluções. Diferentes operadores de vizinhança podem ser empregados para modificar a permutação da sequência de padrões, permitindo explorar novas soluções e escapar de ótimos locais. Nesta implementação, serão considerados alguns tipos de operadores, dentre eles incluem-se:

***swap*:** este operador seleciona aleatoriamente duas posições na permutação e troca os padrões nelas contidos, gerando uma nova solução vizinha. Considera-se a permutação $\pi = [3, 6, 2, 5, 4, 2]$. Se os padrões nas posições 2 e 5 – destacadas em amarelo – forem

trocados, a nova permutação resultante será $\pi = [3, 4, 2, 5, 6, 2]$, conforme ilustrado na Figura 4.3.

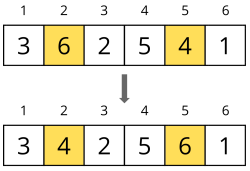


Figura 4.3 – Exemplo de movimento *swap* entre as posições 2 e 5.

**2-opt:** este operador seleciona duas posições na permutação e inverte a ordem dos padrões contidos no segmento delimitado por essas posições, gerando uma nova solução vizinha. Considera-se novamente a permutação $\pi = [3, 6, 2, 5, 4, 2]$. Se forem escolhidas as posições 2 e 5 – destacadas em amarelo – o segmento $[6, 2, 5, 4]$ é invertido para $[4, 5, 2, 6]$ – também destacado em amarelo – resultando na nova permutação $\pi = [3, 4, 5, 2, 6, 2]$, conforme ilustrado na Figura 4.4.

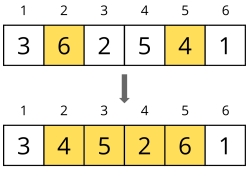


Figura 4.4 – Exemplo de movimento 2-opt entre as posições 2 e 5.

**inserção aleatória:** este operador seleciona aleatoriamente uma posição da permutação, remove o padrão correspondente e o reinsere em uma posição diferente, selecionada aleatoriamente, gerando uma nova solução vizinha. Considera-se mais uma vez a permutação $\pi = [3, 6, 2, 5, 4, 2]$. Se o padrão na posição 2 (padrão 6) for removido e inserido na posição 5, a nova permutação resultante será $\pi = [3, 2, 5, 4, 6, 2]$, conforme ilustrado na Figura 4.5 e destacado em amarelo.

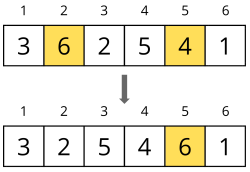


Figura 4.5 – Exemplo de movimento de inserção do elemento da posição 2 na posição 5.

## 4.5 Critério de parada

O critério de parada é baseado no número máximo de propostas de trocas de temperaturas entre réplicas. Durante a execução do PT, cada réplica realiza vários ciclos de cadeias de Markov.

Ao final de cada ciclo, o algoritmo avalia a possibilidade de troca de temperaturas entre réplicas adjacentes. A quantidade de trocas efetuadas é monitorada ao longo da execução. Quando o número de trocas propostas atinge um limite pré-estabelecido, o laço principal do algoritmo é interrompido. Esse mecanismo de parada atua de forma dupla:

- Impedindo execuções excessivamente longas e, consequentemente, preservando recursos computacionais.
- Equilibrando a exploração e a intensificação da busca, já que as trocas de temperatura facilitam a superação de barreiras energéticas e a fuga de mínimos locais, direcionando a busca para regiões mais promissoras.

Dessa maneira, o critério de parada baseado no número máximo de trocas de temperaturas assegura que o algoritmo finalize sua execução em um tempo razoável, mantendo a eficácia na obtenção de soluções de alta qualidade. Após realizada a implementação descrita neste capítulo, serão conduzidos experimentos abrangentes para avaliar a performance do método, tanto quanto à qualidade das soluções obtidas quanto à sua eficiência computacional.

# 5 Experimentos Computacionais

Neste capítulo, são apresentados os experimentos computacionais realizados com o objetivo de avaliar o desempenho do PT-MOSP (PT aplicado ao MOSP) em comparação ao *PieceRank*, proposto por Frinhani, Carvalho e Soma (2018). Inicialmente, descrevem-se as instâncias utilizadas. Em seguida, são discutidos os experimentos preliminares conduzidos para definir os principais parâmetros e estratégias do algoritmo. Por fim, testaram-se duas abordagens de sequenciamento – por padrões e por peças – previamente discutidas.

Os experimentos computacionais foram realizados em um computador com processador Intel Core i7-10700 CPU @ 2.90 GHz, com 8 núcleos físicos e 16 *threads*, possuindo 16 MB de cache L3 e 16 GB de memória RAM. O sistema operacional utilizado foi o Ubuntu 20.04. As implementações foram desenvolvidas na linguagem C++ e compiladas utilizando o compilador g++ na versão 11.4.0, com a opção de otimização `-O3`. Para a implementação do algoritmo, foi utilizada a API do PT desenvolvida por Almeida, de Castro Lima e Carvalho (2025) A execução do PT foi configurada para utilizar 5 *threads*, de forma a aproveitar o paralelismo do processador. Todos os códigos-fonte, bem como os resultados individuais, estão disponíveis em <https://github.com/Laps-F/TCC>.

## 5.1 Instâncias

Para a realização dos experimentos computacionais apresentados nesta monografia, empregou-se um conjunto de 610 instâncias, onde cada instância foi executada 5 vezes de forma independente, a fim de reduzir a influência da aleatoriedade. Este conjunto de dados foi especificamente proposto e gerado por Frinhani, Carvalho e Soma (2018), como parte de um esforço para suprir a necessidade de instâncias de maiores dimensões na literatura e, consequentemente, permitir uma avaliação mais robusta do comportamento de métodos heurísticos em cenários complexos.

A geração dessas instâncias foi realizada utilizando o gerador de Chu e Stuckey (2009). Um aspecto fundamental que diferencia este conjunto de dados de seus predecessores é a introdução de uma maior variedade na quantidade de peças por padrão. Enquanto conjuntos anteriores se restringiam a 2, 4, 6, 8 e 10 peças por padrão, as novas instâncias de Frinhani, Carvalho e Soma (2018) foram elaboradas para permitir a análise do desempenho das heurísticas em grafos de densidades variadas. O objetivo por trás dessa estratégia foi identificar uma possível convergência no valor das soluções obtidas pelos métodos a partir de determinadas densidades do grafo. As dimensões das instâncias criadas e utilizadas nos experimentos são detalhadas a seguir:

- **Instâncias 400x400:** Compostas por dez subconjuntos de instâncias, cada um com número

médio de 2, 4, 6, 8, 10, 14, 18, 20, 24, 28, 30 e 34 peças por padrão.

- **Instâncias 600x600:** Incluem as mesmas proporções de peças por padrão que as de 400x400, adicionando instâncias com número médio de 38 e 40 peças por padrão.
- **Instâncias 800x800:** Mantêm as proporções da categoria anteior, estendendo para um número médio de 44, 48 e 50 peças por padrão.
- **Instâncias 1000x1000:** Representam as maiores instâncias do conjunto, com as mesmas proporções de peças por padrão da categoria anterior, incluindo também um número médio de 54 peças por padrão.

A geração de uma associação peça-padrão é feita de forma aleatória, com base em uma densidade-alvo $d = \frac{\bar{N}}{N}$, em que $\bar{N}$ é o número médio de peças por padrão e $N$ é o total de peças da instância. Para cada par peça-padrão, há uma probabilidade $d$ de se estabelecer uma relação de pertinência. O processo de geração garante que todas as peças estejam associadas a pelo menos um padrão e que cada padrão contenha no mínimo duas peças, evitando colunas ou linhas vazias. Além disso, assegura-se a conectividade do grafo implícito da instância ao garantir que cada padrão compartilhe peças com os demais.

Ao analisar as instâncias, verificou-se um comportamento esperado em relação à densidade dos grafos gerados. Como o algoritmo de geração de instâncias busca garantir a conectividade entre os padrões por meio do compartilhamento de peças, à medida que se aumenta o número médio de peças por padrão – isto é, a densidade $d$, o grafo resultante tende a se aproximar de um grafo completo. Essa característica implica que os conjuntos de instâncias com maiores valores de $\bar{N}$ possuem estruturas mais densas e, consequentemente, segundo (Yanasse; Senne, 2010), valores de solução únicos e próximos a *N*.

## 5.2 Experimentos preliminares

Antes da execução dos experimentos principais, foram realizados testes preliminares com o objetivo de definir as melhores estratégias de configuração do algoritmo PT-MOSP. Essa etapa envolveu três aspectos fundamentais: a escolha do método de geração de soluções iniciais, a definição de um formato adequado de codificação das soluções e o ajuste dos parâmetros do algoritmo.

Inicialmente, foram comparadas duas abordagens distintas para a geração de soluções iniciais: uma completamente aleatória e outra baseada no método MCNH. A análise dos resultados demonstrou que o PT-MOSP, utilizando soluções geradas a partir do MCNH, produzia soluções significativamente melhores e mais consistentes, o que levou à sua escolha como gerador padrão nas demais etapas experimentais.

Antes de realizar a calibração formal dos parâmetros, foi realizada uma execução de teste do PT-MOSP com uma configuração inicial básica. Essa execução teve como objetivo apenas obter valores de referência, servindo como ponto de partida para a calibração. Esses resultados preliminares foram posteriormente superados pelas implementações ajustadas obtidas nas etapas de calibração e otimização subsequentes.

O próximo passo foi realizar a calibração dos parâmetros do PT-MOSP. Para isso, utilizou-se o pacote iRace, uma ferramenta amplamente utilizada para ajustar automaticamente os parâmetros de algoritmos de otimização. A principal vantagem do iRace é sua capacidade de testar diversas configurações em paralelo, avaliando o desempenho de cada uma com base em resultados obtidos em um conjunto de instâncias previamente definido de maneira aleatória. Ao longo das execuções, as combinações menos promissoras são descartadas, enquanto as mais eficientes são refinadas e testadas novamente (López-Ibáñez et al., 2016). Essa abordagem permitiu identificar, de forma sistemática, um conjunto de parâmetros com bom desempenho médio em diferentes cenários.

Os parâmetros que foram ajustados e seus valores finais (destacados em negrito) são descritos na Tabela 5.1.Vale ressaltar que o número de tentativas de trocas de temperaturas foi definido separadamente do processo de calibração via iRace. Essa decisão foi tomada para evitar que o valor fosse ajustado de forma a extrapolar o tempo de execução previsto para cada instância, mantendo a viabilidade computacional dos experimentos.

Tabela 5.1 – Parâmetros da calibração.

| **Parâmetro** | **Valores possíveis** |
|---|---|
| $T_0$ (temperatura inicial) | $\{\mathbf{0{,}01}; 0, 1; 1\}$ |
| $T_f$ (temperatura final) | $\{3; 5; \mathbf{10}\}$ |
| $N_{kmax}$ (comprimento da cadeia de Markov) | $\{300; 400; 500; \mathbf{600}\}$ |
| Distribuição de temperatura[a] | $\{1; \mathbf{2}; 3; 4\}$ |
| Forma de gerar soluções vizinhas[b] | $\{0; 1; \mathbf{2}\}$ |
| Tipo de atualização de temperatura[c] | $\{\mathbf{0}; 1; 2; 3\}$ |
| Taxa de ajuste das temperaturas[d] | $\{3; 4; \mathbf{5}\}$ |

[a] 1: linear, 2: linear inteira, 3: exponencial, 4: geométrica.
[b] 0: swap, 1: 2-opt, 2: inserção aleatória.
[c] 0: sem ajuste; 1: ajusta para taxa de troca $\approx 23,4\%$, 2: equaliza probabilidade de troca entre réplicas vizinhas, 3: otimiza o tempo de ida e volta entre temperaturas extremas (Almeida; de Castro Lima; Carvalho, 2025).
[d] Calculada como o número total de trocas dividido pelo valor da taxa.

Após a calibragem dos parâmetros, foram implementadas duas variações do PT-MOSP, diferenciadas pelo tipo de codificação adotado. A primeira versão utiliza sequenciamento por padrões. Embora funcional e com bons resultados, essa abordagem apresentou um gargalo significativo: para manter a qualidade das soluções, era necessário aumentar progressivamente os valores de $qT_{emp}$ (número de propostas de trocas de temperatura) e de $N_{kmax}$, o que resultou em tempos de execução mais longos, sem ganhos proporcionais na qualidade das soluções.

Como alternativa, foi utilizada a codificação baseada em sequenciamento de peças. Essa variação reduziu o crescimento da demanda pelo $qT_{emp}$ e pelo $N_{kmax}$, além de apresentar resultados mais competitivos em termos de qualidade de solução e eficiência de execução, com uma melhoria média em torno de 3% de *gap* em relação ao sequenciamento de padrões em instâncias menos densas e de 1,5% de *gap* em relação à codificação baseada no sequenciamento de padrões no geral, justificando sua adoção como abordagem principal nos experimentos comparativos. Entretanto, por ser computacionalmente mais custosa, o tempo de execução passou a se situar na ordem de minutos (ao invés de milissegundos) para cada instância.

## 5.3 Comparação com o estado da arte

As Tabelas 5.2 a 5.6 apresentam a comparação dos resultados entre o método PT-MOSP e o algoritmo *PieceRank* (Frinhani; Carvalho; Soma, 2018). A coluna Conjunto indica o subconjunto de instâncias, agrupado pelo número médio de peças por padrão. A *PieceRank* Sol. exibe a melhor solução obtida pelo *PieceRank* para cada subconjunto. As colunas seguintes apresentam, para o PT-MOSP, a melhor solução encontrada durante as execuções ($S^*$), a média das soluções obtidas ($S$), a solução inicial média gerada antes da otimização ($S_0$), o tempo médio de execução em minutos ($T(m)$) e o desvio padrão das soluções ($\sigma$). Por fim, as três últimas colunas mostram as distâncias percentuais entre valores comparados; para simplificar a notação, a solução obtida pelo método *PieceRank* será representada pela variável $PR$. Com isso, os *gaps* percentuais são calculados da seguinte forma:

$$
\begin{aligned}
gap_{S^*}(\%) &= \frac{S^* - PR}{PR} \times 100, \\
gap_{S}(\%) &= \frac{S - PR}{PR} \times 100, \\
gap_{S_0,S}(\%) &= \frac{S - S_0}{S_0} \times 100,
\end{aligned}
$$

onde $S^*$ representa a melhor solução encontrada pelo PT-MOSP, $S$ é a média das soluções obtidas e $S_0$ corresponde à solução inicial média gerada antes da otimização. Assim, $gap_{S^*}$ e $gap_S$ indicam a diferença percentual entre o PT-MOSP e o *PieceRank*, enquanto $gap_{S_0,S}$ mostra a melhoria percentual da solução média em relação à inicial. O tempo de execução do *PieceRank* foi omitido por ser muito curto, da ordem de poucos milissegundos. Observa-se que, quanto maior o número médio de peças por padrão na coluna Conjunto, mais densas são as instâncias daquele subconjunto.

Tabela 5.2 – Comparação entre o PT-MOSP e o *PieceRank* para instâncias 400x400.

| Conjunto | *PieceRank* | PT-MOSP | | | | | $gap_{S^*}$(%) | $gap_S$(%) | $gap_{S_0,S}$(%) |
|---|---|---|---|---|---|---|---|---|---|
| | Sol. | $S^*$ | $S$ | $S_0$ | $T(m)$ | $\sigma$ | | | |
| 2 | 82,60 | **70,70** | 71,40 | 99,70 | 0,61 | 0,45 | -14,41 | -13,56 | -28,39 |
| 4 | 177,60 | **162,90** | 164,20 | 181,90 | 0,78 | 0,91 | -8,28 | -7,55 | -9,73 |
| 6 | 254,80 | **241,80** | 243,24 | 254,80 | 0,97 | 0,96 | -5,10 | -4,54 | -4,54 |
| 8 | 302,80 | **292,60** | 294,12 | 303,10 | 1,19 | 1,17 | -3,37 | -2,87 | -2,96 |
| 10 | 331,40 | **323,40** | 324,70 | 330,10 | 1,37 | 0,91 | -2,41 | -2,02 | -1,64 |
| 14 | 361,70 | **357,00** | 357,76 | 361,40 | 1,76 | 0,63 | -1,30 | -1,09 | -1,01 |
| 18 | 375,60 | **372,90** | 373,68 | 376,30 | 2,15 | 0,35 | -0,72 | -0,51 | -0,70 |
| 20 | 380,40 | **378,20** | 378,80 | 380,30 | 2,30 | 0,50 | -0,58 | -0,42 | -0,39 |
| 24 | 386,70 | **384,80** | 385,34 | 386,70 | 2,70 | 0,50 | -0,49 | -0,35 | -0,35 |
| 28 | 390,10 | **389,00** | 389,50 | 390,20 | 3,00 | 0,17 | -0,28 | -0,15 | -0,18 |
| 30 | 391,70 | **390,60** | 391,04 | 391,60 | 3,26 | 0,29 | -0,28 | -0,17 | -0,14 |
| 34 | 393,90 | **393,10** | 393,24 | 393,80 | 3,60 | 0,12 | -0,20 | -0,17 | -0,14 |

É possível observar que o PT-MOSP apresenta melhorias significativas, especialmente nos subconjuntos com menor número médio de peças por padrão, ou seja, para os conjuntos com média de peças por padrão até 14, onde o *gap* percentual entre a melhor solução do PT-MOSP e a solução do PieceRank é consideravelmente maior. Essa tendência é consistente ao longo das quatro dimensões de instâncias analisadas. Conforme o tamanho dos subconjuntos cresce, a diferença relativa entre os métodos tende a diminuir, indicando que a vantagem do PT-MOSP se torna menos expressiva em instâncias mais densas. Ainda assim, o método mantém uma vantagem, ainda que mais sutil, em praticamente todos os subconjuntos apresentados.

Esse comportamento pode ser explicado pela própria característica das instâncias. Devido ao modo como são geradas, os grafos correspondentes ao MOSP são sempre conexos, garantindo que exista pelo menos um caminho entre quaisquer dois vértices (padrões). À medida que o número médio de peças por padrão aumenta, o grafo se torna mais denso, aproximando-se estruturalmente de um grafo completo. Em um grafo completo, no entanto, a solução ótima é única (Yanasse, 1997) , pois todas as conexões possíveis já estão presentes. Dessa forma, quanto maior a densidade, mais restrito se torna o espaço de busca e menores são as margens para melhoria, o que justifica a redução da diferença percentual observada entre o PT-MOSP e o *PieceRank* nas instâncias mais densas.

Tabela 5.3 – Comparação entre o PT-MOSP e o *PieceRank* para instâncias 600x600.

| Conjunto | *PieceRank* | PT-MOSP | | | | | $gap_{S^*}$(%) | $gap_S$(%) | $gap_{S_0,S}$(%) |
|---|---|---|---|---|---|---|---|---|---|
| | Sol. | $S^*$ | $S$ | $S_0$ | $T(m)$ | $\sigma$ | | | |
| 2 | 125,40 | **105,90** | 106,94 | 145,80 | 0,98 | 0,79 | -15,55 | -14,72 | -26,65 |
| 4 | 266,40 | **242,30** | 244,38 | 270,30 | 1,31 | 1,09 | -9,05 | -8,27 | -9,59 |
| 6 | 379,20 | **361,30** | 363,76 | 382,40 | 1,64 | 1,37 | -4,72 | -4,07 | -4,87 |
| 8 | 452,60 | **438,50** | 439,86 | 453,00 | 1,90 | 1,47 | -3,12 | -2,81 | -2,90 |
| 10 | 494,50 | **485,10** | 486,52 | 494,60 | 2,24 | 1,26 | -1,90 | -1,61 | -1,63 |
| 14 | 540,10 | **533,20** | 534,30 | 538,60 | 2,80 | 0,94 | -1,28 | -1,07 | -0,80 |
| 18 | 562,00 | **557,80** | 558,96 | 562,10 | 3,41 | 0,72 | -0,75 | -0,54 | -0,56 |
| 20 | 569,90 | **566,40** | 567,14 | 569,40 | 3,67 | 0,50 | -0,61 | -0,48 | -0,40 |
| 24 | 577,30 | **576,50** | 577,10 | 577,30 | 4,27 | 0,53 | -0,14 | -0,03 | -0,03 |
| 28 | 584,20 | **582,80** | 583,46 | 584,80 | 4,83 | 0,45 | -0,24 | -0,13 | -0,23 |
| 30 | 586,60 | **585,00** | 585,74 | 586,10 | 5,09 | 0,33 | -0,27 | -0,15 | -0,06 |
| 34 | 590,00 | **588,80** | 589,22 | 590,30 | 5,69 | 0,52 | -0,20 | -0,13 | -0,18 |
| 38 | 591,60 | **591,20** | 591,64 | 591,80 | 6,31 | 0,40 | -0,07 | 0,01 | -0,03 |
| 40 | 592,90 | **592,10** | 592,54 | 592,70 | 6,48 | 0,35 | -0,13 | -0,06 | -0,03 |

O tempo médio de execução do PT-MOSP aumenta conforme o tamanho das instâncias cresce, o que é esperado dada a maior complexidade computacional, variando de alguns segundos para as menores instâncias até cerca de quinze minutos para as maiores. Apesar do aumento no tempo, o ganho em qualidade das soluções justifica o custo computacional adicional, principalmente para os subconjuntos menores, onde a melhoria é mais expressiva. Ressalta-se, contudo, que é possível reduzir ainda mais esse tempo, caso se julgue necessário, por meio da adoção de um gatilho dinâmico que interrompa a execução do método quando estagnado. Em média, o método converge quando o valor de $qT_{emp}$ atinge 65% de seu valor total.

Outro ponto importante a destacar é a melhoria significativa do PT-MOSP em relação à sua própria solução inicial média. Seguindo o mesmo padrão para conjuntos com média de peças por padrão até 14, observa-se uma redução expressiva entre a solução inicial ($S_0$) e a média das soluções obtidas após otimização ($S$), evidenciada pelos valores negativos do $gap_{S_0,S}$. Isso indica que o processo de otimização aplicado pelo PT-MOSP é eficaz em aprimorar as soluções iniciais geradas que possuem espaço para melhora, contribuindo para a qualidade dos resultados finais.

Tabela 5.4 – Comparação entre o PT-MOSP e o *PieceRank* para instâncias 800x800.

| Conjunto | *PieceRank* | PT-MOSP | | | | | $gap_{S^*}(\%)$ | $gap_S(\%)$ | $gap_{S_0,S}(\%)$ |
|---|---|---|---|---|---|---|---|---|---|
| | Sol. | $S^*$ | $S$ | $S_0$ | $T(m)$ | $\sigma$ | | | |
| 2 | 161,50 | **136,60** | 138,08 | 190,40 | 1,45 | 1,13 | -15,42 | -14,50 | -27,48 |
| 4 | 354,70 | **323,10** | 324,96 | 364,00 | 1,88 | 1,26 | -8,91 | -8,38 | -10,73 |
| 6 | 504,20 | **480,80** | 483,18 | 605,30 | 2,31 | 2,01 | -4,64 | -4,17 | -20,18 |
| 8 | 596,60 | **581,70** | 584,10 | 599,60 | 2,73 | 1,50 | -2,50 | -2,10 | -2,59 |
| 10 | 658,50 | **647,10** | 649,18 | 658,40 | 3,21 | 1,53 | -1,73 | -1,42 | -1,40 |
| 14 | 720,50 | **712,80** | 714,70 | 720,30 | 3,91 | 1,08 | -1,07 | -0,80 | -0,78 |
| 18 | 749,80 | **744,20** | 745,72 | 748,70 | 4,76 | 0,88 | -0,75 | -0,54 | -0,40 |
| 20 | 758,40 | **755,40** | 756,18 | 757,80 | 5,19 | 0,71 | -0,40 | -0,29 | -0,21 |
| 24 | 770,60 | **768,20** | 769,24 | 770,50 | 5,96 | 0,78 | -0,31 | -0,18 | -0,16 |
| 28 | 778,00 | **776,80** | 777,86 | 778,40 | 6,82 | 0,64 | -0,15 | -0,02 | -0,07 |
| 30 | 781,20 | **780,20** | 781,14 | 781,10 | 7,14 | 0,50 | -0,13 | -0,01 | 0,01 |
| 34 | 785,30 | **784,80** | 785,46 | 785,30 | 7,87 | 0,49 | -0,06 | 0,02 | 0,02 |
| 38 | 788,90 | **787,70** | 788,40 | 788,80 | 8,53 | 0,40 | -0,15 | -0,06 | -0,05 |
| 40 | 789,80 | **789,50** | 789,98 | 789,90 | 9,21 | 0,39 | -0,04 | 0,02 | 0,01 |
| 44 | 791,90 | **791,10** | 791,74 | 792,10 | 9,85 | 0,40 | -0,10 | -0,02 | -0,05 |
| 48 | 793,20 | **792,80** | 793,22 | 793,11 | 10,41 | 0,43 | -0,05 | 0,00 | 0,01 |
| 50 | 793,80 | **793,50** | 793,84 | 794,18 | 11,16 | 0,40 | -0,04 | 0,01 | -0,04 |

Quanto à variabilidade dos resultados, os valores do desvio padrão ($\sigma$) apresentados são consistentemente baixos, geralmente inferiores a 2, o que indica que as soluções obtidas em múltiplas execuções independentes são bastante estáveis e reproduzíveis. Essa baixa variabilidade reforça a confiabilidade do método, sugerindo que ele fornece resultados consistentes para as mesmas instâncias.

Para verificar a significância estatística das diferenças, foi aplicado inicialmente o teste de normalidade de Shapiro–Wilk a cada grupo de instâncias. Nenhum dos grupos apresentou distribuição normal, com $p$-valores inferiores a 0,05 em todas as dimensões, o que justificou a utilização do teste não paramétrico de Wilcoxon para dados pareados.

Tabela 5.5 – Resultados dos testes estatísticos aplicados às instâncias do MOSP.

| **Dimensão** | **$p$-valor Shapiro–Wilk** | **$p$-valor Wilcoxon** |
|---|---|---|
| 400×400 | $4.22 \times 10^{-2}$ | $4.88 \times 10^{-4}$ |
| 600×600 | $1.09 \times 10^{-2}$ | $9.79 \times 10^{-4}$ |
| 800×800 | $5.31 \times 10^{-4}$ | $2.92 \times 10^{-4}$ |
| 1000×1000 | $1.56 \times 10^{-4}$ | $2.75 \times 10^{-4}$ |

Conforme mostra a Tabela 5.5, os $p$-valores do teste de Wilcoxon foram todos inferiores a 0,001, indicando que as diferenças entre o PT-MOSP e o *PieceRank* são estatisticamente significativas em todas as dimensões analisadas. Esses resultados reforçam que as melhorias obtidas pelo PT-MOSP não se devem ao acaso, mas refletem sua superioridade sistemática sobre o método de referência.

Tabela 5.6 – Comparação entre o PT-MOSP e o *PieceRank* para instâncias 1000x1000.

| Conjunto | *PieceRank* | PT-MOSP | | | | | $gap_{S^*}$(%) | $gap_S$(%) | $gap_{S_0,S}$(%) |
|---|---|---|---|---|---|---|---|---|---|
| | Sol. | $S^*$ | $S$ | $S_0$ | $T(m)$ | $\sigma$ | | | |
| 2 | 200,80 | **171,40** | 173,12 | 237,30 | 1,96 | 1,37 | -14,64 | -13,78 | -27,05 |
| 4 | 439,30 | **399,50** | 402,68 | 449,40 | 2,50 | 2,30 | -9,06 | -8,34 | -10,40 |
| 6 | 630,90 | **605,00** | 607,98 | 635,80 | 3,05 | 2,35 | -4,11 | -3,63 | -4,38 |
| 8 | 748,60 | **727,70** | 731,14 | 748,20 | 3,63 | 1,68 | -2,79 | -2,33 | -2,28 |
| 10 | 821,20 | **809,10** | 811,22 | 822,30 | 4,13 | 1,91 | -1,47 | -1,22 | -1,35 |
| 14 | 899,10 | **890,50** | 892,74 | 899,20 | 5,21 | 1,18 | -0,96 | -0,71 | -0,72 |
| 18 | 935,10 | **931,00** | 932,22 | 936,00 | 6,24 | 0,95 | -0,44 | -0,31 | -0,40 |
| 20 | 945,90 | **942,40** | 944,02 | 945,80 | 6,68 | 0,98 | -0,37 | -0,20 | -0,19 |
| 24 | 962,90 | **960,60** | 961,64 | 962,50 | 7,81 | 0,90 | -0,24 | -0,13 | -0,09 |
| 28 | 972,10 | **971,00** | 971,98 | 972,40 | 8,76 | 0,72 | -0,11 | -0,01 | -0,04 |
| 30 | 975,30 | **974,50** | 975,76 | 976,10 | 9,19 | 0,55 | -0,08 | 0,05 | -0,03 |
| 34 | 981,60 | **980,80** | 981,68 | 981,80 | 10,44 | 0,66 | -0,08 | 0,01 | -0,01 |
| 38 | 985,00 | **984,50** | 985,22 | 985,10 | 11,23 | 0,62 | -0,05 | 0,02 | 0,01 |
| 40 | 986,70 | **986,20** | 986,98 | 986,90 | 11,76 | 0,39 | -0,05 | 0,03 | 0,01 |
| 44 | 989,20 | **988,70** | 989,32 | 989,30 | 12,65 | 0,46 | -0,05 | 0,01 | 0,00 |
| 48 | 991,40 | **990,80** | 991,26 | 991,40 | 13,60 | 0,50 | -0,06 | -0,01 | -0,01 |
| 50 | **991,60** | 991,80 | 992,14 | 991,90 | 14,00 | 0,45 | 0,02 | 0,05 | 0,02 |
| 54 | 992,90 | **992,80** | 993,30 | 993,30 | 15,23 | 0,46 | -0,01 | 0,04 | 0,00 |

Os resultados dos testes estatísticos indicaram que, para todas as instâncias avaliadas, houve diferença estatisticamente significativa entre os métodos comparados. Esses resultados confirmam que o PT-MOSP supera consistentemente o *PieceRank*, tendo um desempenho superior e consolidando-se como o novo estado da arte na resolução do MOSP.

# 6 Conclusão

Esta monografia abordou o *minimization of open stack problem* (MOSP), um problema combinatório NP-difícil de relevância indiscutível no cenário industrial, e propôs a aplicação da meta-heurística *parallel tempering* (PT) para sua resolução. A inviabilidade de métodos exatos em instâncias de grande porte justifica o uso de abordagens heurísticas, e o PT demonstrou potencial significativo para esta classe de problemas.

O PT-MOSP evidenciou eficácia consistente na melhoria da qualidade das soluções, destacando-se em instâncias de menor densidade, onde superou o método de referência (*Piece-Rank*) e explorou regiões do espaço de busca menos acessíveis a outras abordagens. Em instâncias mais densas, a alta conectividade dos grafos MOSP tende a aproximar todos os métodos de uma solução praticamente única, reduzindo as diferenças de desempenho, mas o PT-MOSP manteve competitividade e robustez. Essa alta conectividade reduz as diferenças de desempenho e resulta, em muitos casos, em um empate técnico entre as abordagens.

Em termos de custo computacional, o PT-MOSP demandou tempos de execução de alguns segundos para instâncias menores até cerca de quinze minutos para as maiores. Esse investimento mostrou-se compensatório, considerando os ganhos obtidos em qualidade de solução, representando um *trade-off* adequado para contextos industriais que priorizam otimização fina e redução de custos a médio e longo prazo. Outro aspecto relevante foi a robustez do método. O PT-MOSP aprimorou de forma consistente as soluções iniciais geradas heuristicamente e apresentou baixa variabilidade entre execuções, característica essencial para aplicações em ambiente produtivo, onde estabilidade e reprodutibilidade são fatores críticos para adoção tecnológica.

Os experimentos computacionais e testes estatísticos confirmaram que as diferenças observadas entre o PT-MOSP e o *PieceRank* são estatisticamente significativas, consolidando o PT-MOSP como o atual estado da arte na resolução do MOSP. Em síntese, o método mostrou-se uma abordagem competitiva, robusta e adaptável, com potencial para futuras melhorias em implementações paralelas, estratégias de vizinhança e aplicação a problemas combinatórios correlatos em cenários industriais desafiadores.

# Referências


ABENSUR, E. O. *Pesquisa operacional para cursos de engenharia de produção*. São Paulo, Brasil: Editora Blucher, 2018.

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>.

ASHIKAGA, F. M.; SOMA, N. Y. A heuristic for the minimization of open stacks problem. *Pesquisa Operacional*, SciELO Brasil, v. 29, p. 439–450, 2009.

BANDA, M. G. D. L.; STUCKEY, P. J. Dynamic programming to minimize the maximum number of open stacks. *INFORMS Journal on Computing*, v. 19, n. 4, p. 607–617, 2007.

BECCENERI, J. C.; YANASSE, H. H.; SOMA, N. Y. A method for solving the minimization of the maximum number of open stacks problem within a cutting process. *Computers & operations research*, Elsevier, v. 31, n. 14, p. 2315–2332, 2004.

CAI, W.; EWING, D. J.; MA, L. Investigation of temperature parallel simulated annealing for optimizing continuous functions with application to hyperspectral tomography. *Applied Mathematics and Computation*, Elsevier, v. 217, n. 12, p. 5754–5767, 2011.

CARVALHO, M. A. M. D.; SOMA, N. Y. A breadth-first search applied to the minimization of the open stacks. *Journal of the Operational Research Society*, Taylor & Francis, v. 66, n. 6, p. 936–946, 2015.

CHERRI, A. C.; ARENALES, M. N.; YANASSE, H. H.; POLDI, K. C.; VIANNA, A. C. G. The one-dimensional cutting stock problem with usable leftovers–a survey. *European Journal of Operational Research*, Elsevier, v. 236, n. 2, p. 395–402, 2014.

CHU, G.; STUCKEY, P. J. Minimizing the maximum number of open stacks by customer search. In: SPRINGER. *Principles and Practice of Constraint Programming-CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September 20-24, 2009 Proceedings 15*. [S.l.], 2009. p. 242–257.

CONFEDERAçãO NACIONAL DA INDúSTRIA. *Perfil da Indústria Brasileira - Indústria de Transformação*. 2025. [Online; accessed 25-março-2024]. Disponível em: <https://industriabrasileira.portaldaindustria.com.br/grafico/total/producao/#/industria-transformacao>.

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.

FAGGIOLI, E.; BENTIVOGLIO, C. A. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, Elsevier, v. 110, n. 3, p. 564–575, 1998.

FINK, A.; VOSS, S. Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, Elsevier, v. 26, n. 1, p. 17–34, 1999.

FRINHANI, R. d. M. D.; CARVALHO, M. A. M.; SOMA, N. Y. A pagerank-based heuristic for the minimization of open stacks problem. *Plos one*, Public Library of Science San Francisco, CA USA, v. 13, n. 8, p. e0203076, 2018.

GEYER, C. J. et al. Markov chain monte carlo maximum likelihood. In: NEW YORK. *Computing science and statistics: Proceedings of the 23rd Symposium on the Interface*. [S.l.], 1991. v. 156163.

GIOVANNI, L. D.; MASSI, G.; PEZZELLA, F. An adaptive genetic algorithm for large-size open stack problems. *International Journal of Production Research*, Taylor & Francis, v. 51, n. 3, p. 682–697, 2013.

GONÇALVES, J. F.; RESENDE, M. G.; COSTA, M. D. A biased random-key genetic algorithm for the minimization of open stacks problem. *International Transactions in Operational Research*, Wiley Online Library, v. 23, n. 1-2, p. 25–46, 2016.

HASTINGS, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, v. 57, n. 1, p. 97–109, 04 1970. ISSN 0006-3444. Disponível em: <https://doi.org/10.1093/biomet/57.1.97>.

ITO, A. S. *Gestão de Estoques de Produtos Intermediários–Estudo de caso de uma Empresa de Petróleo*. Tese (Doutorado) — PUC-Rio, 2018.

JUN, Y.; MIZUTA, S. Detailed analysis of uphill moves in temperature parallel simulated annealing and enhancement of exchange probabilities. *Complex Systems*, [Champaign, IL, USA: Complex Systems Publications, Inc., c1987-, v. 15, n. 4, p. 349, 2005.

KATAOKA, M.; TSUKIYAMIA, S.; KAMBE, T.; FUKUI, M. An effective method to use gpu for rectangle packing. In: *10th IEEE International NEWCAS Conference*. Montreal, QC, Canada: IEEE, 2012. p. 129–132.

KIMURA, K.; TAKI, K. *Time-homogeneous Parallel Annealing Algorithm: (extended Abstract)*. ICOT, 1991. (ICOT technical report). Disponível em: <https://books.google.com.br/books?id=DYkazwEACAAJ>.

LABORIE, P. S. P. A constraint programming approach to the min-stack problem. *IJCAI05 Constraint Modelling Challenge entry*, v. 19, n. 4, 2005.

LIMA, J. R.; CARVALHO, M. A. M. Descent search approaches applied to the minimization of open stacks. *Computers & Industrial Engineering*, Elsevier, v. 112, p. 175–186, 2017.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, vlsi design, and flexible machines. *Computers & Operations Research*, Elsevier, v. 29, n. 12, p. 1759–1772, 2002.

LOPES, I. C.; CARVALHO, J. V. d. Graph properties of minimization of open stacks problems and a new integer programming model. *Pesquisa Operacional*, SciELO Brasil, v. 35, n. 2, p. 213–250, 2015.

LóPEZ-IBáñEZ, M.; DUBOIS-LACOSTE, J.; Pérez Cáceres, L.; BIRATTARI, M.; STüTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016. ISSN 2214-7160. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214716015300270>.

MADSEN, O. B. Glass cutting in a small firm. *Mathematical Programming*, Springer, v. 17, p. 85–90, 1979.

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.

MIKI, M.; HIROYASU, T.; KASAI, M.; ONO, K.; JITTA, T. Temperature parallel simulated annealing with adaptive neighborhood for continuous optimization problem. In: CITESEER. *Second international workshop on Intelligent systems design and application*. MIT, Boston, USA, 2002. p. 149–154.

MIKI, M.; HIROYASU, T.; WAKO, J.; YOSHIDA, T. Adaptive temperature schedule determined by genetic algorithm for parallel simulated annealing. In: COMPUTATIONAL INTELLIGENCE AND APPLICATIONS. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.* Canberra, ACT, Australia, 2003. v. 1, p. 459–466 Vol.1.

PETRIE, N. W. K. Using customer elimination orderings to minimise the maximum number of open stacks. *IJCAI05 Constraint Modelling Challenge entry*, v. 19, n. 4, 2005.

PRESTWICH, S. Open stack minimisation by local search and reverse dominance reasoning. *IJCAI05 Constraint Modelling Challenge entry*, v. 19, n. 4, 2005.

SANTOS, V. G. M.; CARVALHO, M. A. M. de. Adaptive large neighborhood search applied to the design of electronic circuits. *Applied Soft Computing*, Elsevier, v. 73, p. 14–23, 2018.

SMITH, B.; GENT, I. Constraint modelling challenge 2005. In: *IJCAI 2005 Fifth Workshop on Modelling and Solving Problems with Constraints*. Edinburgh, Scotland: IJCAI Organization, 2005. p. 1–8.

SWENDSEN, R. H.; WANG, J.-S. Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, American Physical Society, v. 57, p. 2607–2609, Nov 1986. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevLett.57.2607>.

WANG, C.; HYMAN, J. D.; PERCUS, A.; CAFLISCH, R. Parallel tempering for the traveling salesman problem. *International Journal of Modern Physics C*, World Scientific, v. 20, n. 04, p. 539–556, 2009.

YAMAMOTO, A.; HASHIMOTO, H. Application of temperature parallel simulated annealing to loading pattern optimizations of pressurized water reactors. *Nuclear science and engineering*, Taylor & Francis, v. 136, n. 2, p. 247–257, 2000.

YANASSE, H. H. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, Elsevier, v. 100, n. 3, p. 454–463, 1997.

YANASSE, H. H.; BECCENERI, J. C.; SOMA, N. Y. Bounds for a problem of sequencing patterns. *Pesquisa Operacional*, v. 19, n. 2, p. 249–277, 1999.

YANASSE, H. H.; LIMEIRA, M. S. Refinements on an enumeration scheme for solving a pattern sequencing problem. *International Transactions in Operational Research*, Wiley Online Library, v. 11, n. 3, p. 277–292, 2004.

YANASSE, H. H.; SENNE, E. L. F. The minimization of open stacks problem: A review of some properties and their use in pre-processing operations. *European Journal of Operational Research*, v. 203, n. 3, p. 559–567, 2010. ISSN 0377-2217.

YUEN, B. J. Heuristics for sequencing cutting patterns. *European Journal of Operational Research*, Elsevier, v. 55, n. 2, p. 183–190, 1991.

YUEN, B. J. Improved heuristics for sequencing cutting patterns. *European Journal of Operational Research*, Elsevier, v. 87, n. 1, p. 57–64, 1995.

YUEN, B. J.; RICHARDSON, K. V. Establishing the optimality of sequencing heuristics for cutting stock problems. *European Journal of Operational Research*, Elsevier, v. 84, n. 3, p. 590–598, 1995.

ZHU, Z.; FANG, C.; KATZGRABER, H. G. *borealis - A generalized global update algorithm for Boolean optimization problems*. 2016. Disponível em: <https://arxiv.org/abs/1605.09399>.