UNIVERSIDADE FEDERAL DE OURO PRETO

INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS

DEPARTAMENTO DE COMPUTAÇÃO

MATEUS FILIPE MOREIRA SILVA

Orientador: Marco Antonio Moreira de Carvalho

METAHEURÍSTICA APLICADA AO PLANEJAMENTO DA PRODUÇÃO EM SISTEMAS DE MANUFATURA FLEXÍVEL: PERÍODOS DE PRODUÇÃO NÃO SUPERVISIONADA, PROCESSAMENTO PRIORITÁRIO DE TAREFAS E CUSTOS RELACIONADOS A TROCAS DE FERRAMENTAS

Ouro Preto, MG

2024

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

MATEUS FILIPE MOREIRA SILVA

METAHEURÍSTICA APLICADA AO PLANEJAMENTO DA PRODUÇÃO EM SISTEMAS DE MANUFATURA FLEXÍVEL: PERÍODOS DE PRODUÇÃO NÃO SUPERVISIONADA, PROCESSAMENTO PRIORITÁRIO DE TAREFAS E CUSTOS RELACIONADOS A TROCAS DE FERRAMENTAS

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: Marco Antonio Moreira de Carvalho

Ouro Preto, MG
2024

Resumo

Com o aumento da demanda dos mercados consumidores por uma maior variedade de produtos em quantidades menores, as empresas têm adotado modelos de manufatura flexível, como o sistema *high-mix*, *low-volume*, para se adaptar às exigências do mercado. Esse cenário exige soluções inovadoras para otimizar o planejamento da produção e reduzir custos. Na presente monografia, propõe-se a aplicação de uma metaheurística, especificamente o revenimento paralelo (Parallel Tempering - PT), para o planejamento da produção em sistemas de manufatura flexível. O problema abordado, conhecido como *Job Sequencing and Tool Switching Problem* (SSP), envolve a otimização da sequência de tarefas, com o objetivo de minimizar o número de trocas de ferramentas, considerando também custos associados e períodos de produção não supervisionada. As características da versão do SSP abordada neste estudo incluem múltiplas máquinas, tarefas com diferentes prioridades e reentrância de operações, o que aumenta a complexidade do problema. A abordagem proposta foi testada em novas instâncias baseadas em dados reais de uma indústria de manufatura, com adaptações necessárias para lidar com inconsistências nos dados. Resultados preliminares indicam que o método PT é eficaz para produzir soluções viáveis de boa qualidade para estas instâncias. Conclui-se que a aplicação do PT pode ser uma estratégia viável para otimização em sistemas de manufatura flexível e, após aprimorações no método, os resultados se tornarão mais competitivos.

Palavras-chave: Manufatura Flexível. Job Sequencing and Tool Switching Problem. Revenimento Paralelo.

Abstract

With the increasing demand from consumer markets for a greater variety of products in smaller quantities, companies have adopted flexible production models, such as the high-mix, low-volume system, to adapt to the criteria of the market. This scenario requires innovative solutions to optimize production planning and reduce costs. In this monograph, we propose the application of a metaheuristic, precisely parallel tempering (PT), for production planning in flexible production systems. The problem addressed, the Job Sequencing and Tool Switching Problem (SSP), involves optimizing the task sequence to minimize the number of tool changes and consider associated costs and unsupervised production periods. The characteristics of the SSP version addressed in this study include multiple machines, tasks with different priorities, and reentrant operations, increasing the problem's complexity. The proposed approach was tested on new instances based on real data from a production industry, with adaptations allowed to deal with inconsistencies in the data. Preliminary results indicate that the PT method effectively produces good quality viable solutions for these instances. It is concluded that applying PT can be a viable strategy for optimization in flexible production systems. After improvements in the method, the results will become more competitive.

Keywords: Flexible Manufacturing. Job Sequencing and Tool Switching Problem. Parallel Tempering.

Lista de Ilustrações

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

PT	*parallel tempering*
SSP	job sequencing and tool switching problem
HMLV	high-mix, low-volume
FMS	flexible manufacturing system
KTNS	Keep Tool Neeeded Soonest
GPCA	Greedy Pipe Construction Algorithm
MCMC	Monte Carlo Markov Chain

Sumário

1 Introdução

Com o advento da indústria 4.0, também referida como quarta Revolução Industrial, emergem novos paradigmas para os processos produtivos, desencadeando uma série de desafios e oportunidades (MORGAN et al., 2021). Adicionalmente, as crescentes demandas do mercado, tais como a necessidade de diversificação de produtos e a pressão por redução de custos, impulsionam a busca incessante por soluções inovadoras e disruptivas (FOGLIATTO; da Silveira; BORENSTEIN, 2012). Como consequência, a interdependência entre processos digitais e físicos assume um papel central, procurando otimizar a interação entre sistemas computacionais e componentes de controladores físicos da melhor forma possível (SPLUNK, 2024). Nesse contexto, as empresas enfrentam o desafio de se adaptar rapidamente às mudanças tecnológicas e às demandas do mercado, a fim de se manterem competitivas e sustentáveis a longo prazo.

A emergência da demanda do mercado consumidor por uma ampla gama de produtos tem impulsionado a adoção de um modelo de manufatura conhecido como *high-mix*, *low-volume* (HMLV) (GAN; MUSA; YAP, 2023). Esse paradigma responde à necessidade de produzir uma variedade cada vez maior de produtos em quantidades menores. No HMLV, a fabricação ocorre em lotes pequenos, permitindo a flexibilidade necessária para atender às demandas do mercado em constante mudança. Cada produto único implica em tempos de produção distintos e ajustes específicos de maquinário, tornando a programação da produção um desafio complexo. Gerenciar a alocação de recursos e determinar a sequência de produção ideal para uma variedade de itens distintos representa um gargalo de produtividade para as empresas que operam em um regime HMLV, tais como produção de semicondutores, eletrônicos, peças usinadas e indústria aeroespacial (GAN; MUSA; YAP, 2023).

A fim de lidar com uma grande variedade de produtos e requisitos complexos dos clientes, as fábricas fazem uso do modelo produtivo denominado *flexible manufacturing system* (FMS) (CALMELS, 2019). Um FMS consiste em uma sequência de máquinas flexíveis em uma linha de produção. Cada máquina flexível possui um *magazine* que pode ser configurado pela instalação de um conjunto de *ferramentas*, cada uma delas projetada para uma função como lixamento, corte, entre outros. A capacidade do *magazine* para ferramentas instaladas em cada máquina é limitada e, na maior parte das aplicações reais, é muito menor que a quantidade total de ferramentas disponível para processar todas as tarefas da produção.

No contexto do problema abordado nesta monografia, uma *tarefa* é definida como uma ou mais operações para fabricação de um produto. Cada operação possui um tempo de processamento e um conjunto de ferramentas necessárias para ser iniciada. A produção é divida em *estágios* em que cada tipo de operação é processada sem interrupções. O conjunto de ferramentas pressente no *magazine* de uma máquina em um estágio pode conter mais do que somente as ferramentas

necessárias para processar a dada operação de uma tarefa, porém, caso alguma ferramenta necessária para o processamento não esteja presente no *magazine* da máquina, é preciso realizar uma troca de ferramentas entre estágios consecutivos da produção.

Assim, se faz necessário determinar a sequência ótima para o processamento das tarefas, a fim de minimizar as trocas de ferramentas, uma vez que trocas resultam em custos adicionais. Na literatura, o custo da troca é representado tanto por interrupções na produção ou por uma penalidade concreta no lucro após o processamento de um conjunto de tarefas (CALMELS, 2019). Esse desafio é referido na literatura como *job sequencing and tool switching problem* (SSP). O SSP envolve determinar a sequência ideal de processamento das tarefas que minimize o número de trocas de ferramentas, além de especificar o plano de troca de ferramentas em si, ou seja, quais ferramentas serão carregadas no *magazine* da máquina flexível a cada estágio de produção.

Existem diversas versões para o SSP, desde sua versão mais básica (TANG; DENARDO, 1988) que considera uma sequência de tarefas a serem processadas em apenas uma máquina, até outras versões variando número de máquinas, objetivos únicos e diversos, ferramentas de tamanhos homogêneos e heterogêneos, lista completa de tarefas conhecida *a priori* ou não, capacidade do *magazine* e desgaste das ferramentas. (CALMELS, 2019). A presente monografia trata de uma versão específica do SSP que consiste em minimizar o custo em máquinas paralelas idênticas. Tarefas são classificadas entre *prioritárias* e *não prioritárias*, e o custo é associado ao número de trocas de ferramentas e ao processamento ou não de tarefas prioritárias. Além disso, as tarefas podem ser *reentrantes*, o que significa que pode haver a necessidade processá-las em mais de um estágio para sua fabricação. Outra característica do problema é a existência de um *período não supervisionado* de produção, de maneira que trocas de ferramentas não podem ser realizadas sem acompanhamento humano, embora a produção possa continuar dado que não sejam necessárias tais trocas. Essa recente versão do problema foi abordada uma única vez na literatura, recentemente por (DANG et al., 2023).

A fim de tratar o problema descrito, será utilizado o método revenimento paralelo (ou *parallel tempering*, PT) o qual se assemelha ao processo de reaquecimento e resfriamento de materiais visando melhorias em suas propriedades físicas. O PT não é popular na área de otimização, ao contrário do que ocorre na área de simulações físico-químicas. Isto não se deve ao desempenho do método, porém a questões históricas e de implementação (ALMEIDA, 2024). Vale ressaltar que implementações paralelas do PT, apesar do nome, não são usuais na literatura, apesar de promissoras. A presente monografia empregará uma implementação paralela do PT a fim de produzir soluções próximas do ótimo para o problema abordado.

1.1 Justificativa

As aplicações do SSP no âmbito industrial são diversas, sendo uma das mais notórias em metalúrgicas. Essa indústria possui um mercado de mais de 200 bilhões de dólares, uma vez que esse tipo de manipulação de metal é usada em diversas outras indústrias, como aeroespacial e fabricação de automóveis, por exemplo (CHCOMPANY, 2024). Nesse contexto, tornos ou máquinas de corte seriam as máquinas do problema e as ferramentas são acopladas as mesmas para processar as operações das tarefas. Dessa forma, abordar o SSP se mostra extremamente vantajoso para minimizar desperdícios e custos.

Outra aplicação do SSP ocorre na montagem de *printed circuit boards* (PCBs) (GHRAYEB; FINCH, 2003), nesse problema, máquinas são alimentadas com componentes (análogos às ferramentas) a serem montados nas placas dos circuitos. Cada placa possui uma configuração específica de componentes, logo pode ser considerada uma tarefa. A indústria de montagem de PCBs tem grande relevância no mercado mundial e um mercado previsto em 73,1 bilhões de dólares até 2027 (INDUSTRYARC, 2024).

Uma vez mostrada a importância do SSP para indústria, é importante ressaltar que o problema em questão pertence à classe NP-Difícil (CRAMA et al., 1994). Isto implica em, à medida que o tamanho do problema aumenta, encontrar uma solução ótima torna-se cada vez mais impraticável em um prazo razoável. Portanto, explorar abordagens alternativas para lidar com o SSP é crucial para o desenvolvimento de algoritmos eficientes que possam fornecer soluções satisfatórias dentro de restrições práticas.

1.2 Objetivos

Em linhas gerais, este trabalho visa empregar o método *revenimento paralelo* (ou *parallel tempering*, PT) para abordagem da versão do SSP abordada com o intuito de obter resultados superiores em relação ao estado da arte atual. Os objetivos específicos são:

1. Realizar uma revisão bibliográfica abrangente sobre o SSP. O foco estará na análise das características dos problemas similares ao tratado na presente monografia;
2. Gerar pesquisa e embasamento teórico a respeito SSP e suas variações;
3. Realizar pesquisa para geração de embasamento teórico a respeito do método *parallel tempering*;
4. Gerar instâncias desafiadoras para o problema a partir de dados reais, disponíveis publicamente por outros estudos.

1.3 Organização do Trabalho

A organização do trabalho segue a seguinte estrutura: o Capítulo 2 é dedicado a revisão dos trabalhos relacionados. O Capítulo 3 trata da fundamentação teórica tanto da versão do SSP em questão quanto do método a ser utilizado para sua abordagem. Em seguida, o Capítulo 4 aborda a metodologia empregada e o seu desenvolvimento, incluindo o detalhamento do método escolhido. O Capítulo 5 relata os experimentos computacionais preliminares e os resultados obtidos considerando instâncias baseadas em dados reais da literatura. O Capítulo 6 apresenta o plano de atividades restantes para conclusão da monografia e, por fim, a conclusão parcial deste estudo é apresentada no Capítulo 7.

2 Trabalhos relacionados

O SSP uniforme serve como base para vários tipos de problemas derivados, incluindo o tema desta monografia. O SSP uniforme destaca-se como o problema mais prevalente na pesquisa sobre problemas de otimização em sistemas de manufatura flexível. Originalmente introduzido por Tang e Denardo (1988), este problema específico envolve determinar a sequência ideal do processamento das tarefas e o plano de trocas de ferramentas correspondente com o objetivo de minimizar o número de trocas de ferramentas. A versão original do SSP é referida como "uniforme" devido aos tempos de configuração constantes e independentes da sequência, ao sequenciamento das tarefas em uma única máquina e aos tamanhos uniformes das ferramentas. O Capítulo 3 apresenta uma descrição detalhada deste problema.

No estudo de Tang e Denardo (1988) é realizada a conjectura de que o SSP é um problema pertencente à classe NP-Difícil, considerando que ele pode ser modelado como um problema do caixeiro viajante com arestas de comprimentos variados. Entretanto, não é apresentada uma prova formal para tal conjectura. Algum tempo depois, o estudo de Crama et al. (1994) prova formalmente que o SSP pertence de fato à classe de problemas NP-Difícil.

Atualmente, para o SSP uniforme, o estado da arte é definido por Mecler, Subramanian e Vidal (2020). Uma meta-heurística denominada busca genética híbrida (HGS) é aplicada à solução do SSP. O mesmo método já havia obtido bons resultados quando aplicado ao problema de roteamento de veículos. O estudo em questão gerou resultados melhores que abordagens anteriores por meio da utilização de um objetivo secundário, além da aplicação inédita do método HGS.

O estudo de Calmels (2019) oferece uma revisão ampla sobre o SSP e suas variantes. Foram analisados estudos considerando um período abrangente, desde 1988 até 2018, considerando cerca de 61 artigos no total. A análise é segmentada conforme as diferentes versões do problema, variando diversos parâmetros conforme a aplicação industrial, tais como os elencados a seguir.

Número de máquinas: Ambientes produtivos que lidam com uma máquina, máquinas paralelas idênticas (PROFESSOR, 1988; BEEZãO et al., 2017) e máquinas paralelas não relacionadas (ÖZPEYNIRCI; GöKGüR; HNICH, 2016).

Tempo de configuração do *magazine*: O tempo de configuração é constante para todas as ferramentas (PROFESSOR, 1988), ou dependente de outras características das ferramentas, como tamanho (PRIVAULT; FINKE, 1995).

Objetivos: Minimizar o número de trocas de ferramentas (CRAMA et al., 1994) ou uma combinação de objetivos envolvendo a redução do tempo total de conclusão das tarefas e a minimização dos instantes de troca de ferramentas (KEUNG; IP; LEE, 2001).

Tamanho das ferramentas e capacidade do *magazine*: Ferramentas que necessitam de mais de um espaço no *magazine* (RUPE; KUO, 1997) e/ou tarefas que necessitam de mais ferramentas do que o *magazine* suporta simultaneamente (MATZLIACH; TZUR, 2000; TZUR; ALTMAN, 2004).

Desgaste de ferramentas: Ferramentas que não se desgastam (PROFESSOR, 1988) ou desgaste de ferramentas representado por distribuição determinística ou estocástica (HIRVI-KORPI et al., 2006; FARUGHI et al., 2017).

A versão específica do SSP a ser abordada na presente monografia foi tratada em apenas um artigo, proposto por Dang et al. (2023), salvo melhor juízo. O problema se caracteriza por uma junção de diversas variações do SSP em um único problema adicionados de novas características. O mesmo se caracteriza pela utilização de múltiplas máquinas para processamento das tarefas, períodos de produção não supervisionada em que tarefas podem ser processadas, mas ferramentas não podem ser retiradas ou adicionadas no *magazine*, distinção entre tarefas prioritárias e não prioritárias, e, por fim, tarefas podem ter reentrância, exigindo que sejam processadas mais de uma vez para serem concluídas. O objetivo inédito desta versão do problema é maximizar o lucro, composto pela diferença entre o prejuízo, definido pela quantidade de tarefas prioritárias não concluídas e o custo das trocas de ferramentas, e o bônus gerado pela conclusão das tarefas regulares e prioritárias dentro do horizonte de planejamento.

A característica de períodos não supervisionados já foi abordada na literatura por Noël, Sodhi e Lamond (2007), visando selecionar a melhor velocidade de corte para o processamento de tarefas em máquinas flexíveis durante um período não supervisionado, porém, os aspectos de trocas de ferramentas não são considerados. Agnetis et al. (2009) abordam um período não supervisionado com probabilidade de falhas de tarefas durante esse período, impedindo que tarefas subsequentes sejam finalizadas. Contudo, o artigo não aborda trocas de ferramentas. As demais características, como tarefas reentrantes, tarefas prioritárias e horizonte de planejamento limitado não são encontrados em estudos relacionados a manufatura flexível.

O estudo de Dang et al. (2023) propõe três abordagens para o problema considerado nesta monografia: um modelo de programação linear inteira mista (MILP), uma heurística de particionamento e um algoritmo genético. Conforme ocorre comumente em modelos de programação linear, apesar da garantia de otimalidade das soluções, o crescimento rápido do tempo de execução à medida que o tamanho das instâncias aumenta limita sua aplicação. Dessa maneira, a heurística de particionamento e o algoritmo genético obtiveram desempenho melhor para instâncias contendo mais do que 25 tarefas.

A pesquisa foi realizada com parceria de uma indústria real. Dessa maneira, as instâncias utilizadas para realização dos experimentos computacionais foram baseadas em dados reais, os quais foram disponibilizados publicamente pelos autores. Contudo, os dados disponibilizados não são suficientes para reproduzir as instâncias utilizadas no artigo, que não foram disponibilizadas.

Desta forma, é necessário realizar algumas suposições sobre as mesmas para tentar oferecer algum tipo de reprodutibilidade e comparabilidade. Além da criação de instâncias inéditas, também foi proposta a adaptação das instâncias disponibilizadas por Beezão et al. (2017) por meio da adição de informações como período não supervisionado, tarefas prioritárias e horizonte de planejamento.

O estudo realizado por Dang et al. (2023) estabelece o estado da arte para a versão do SSP em questão, porém, não é disponibilizado o código-fonte para nenhum dos três métodos utilizados no estudo. Os dados das tarefas disponibilizados pelos autores não contêm informação sobre quais tarefas são prioritárias ou não, sendo essa informação gerada dinamicamente para cada instância proposta. O mesmo ocorre com as informações sobre reentrância de cada tarefa. Esse fato, justamente com a não disponibilização das instâncias, dificulta a reprodutibilidade dos experimentos.

Uma análise preliminar dos dados disponibilizados indicou potenciais falhas, pois os mesmos contêm conjuntos de ferramentas maiores que a capacidade predefinida de 80 *slots* do *magazine*, mencionada no artigo, além de conjuntos de ferramentas que são subconjuntos de outros – o que implicaria na não necessidade de trocas de ferramentas. É válido ressaltar que é possível que parte dos dados redundantes não tenha sido utilizada na criação das instâncias. No entanto, sem acesso às instâncias, não é possível constatar ou não essa suspeita.

Outro aspecto problemático é que, após analisar os resultados publicados no artigo, os três métodos utilizados pelos autores conseguiram obter o valor ótimo para a função objetivo do problema em 38,4% das instâncias. Para 61,5% das instâncias, os resultados indicam que não houve penalidades associadas a trocas de ferramentas ou a tarefas prioritárias não finalizadas. Portanto, há indícios de que parte das instâncias pode não incluir dados relevantes sobre trocas de ferramentas e horizonte de planejamento, reduzindo assim o desafio para se obter bons resultados.

Levando em consideração as potenciais falhas apresentadas anteriormente, tornou-se necessária a criação de novas instâncias disponíveis publicamente para estudos futuros. Esta tarefa é descrita na Seção 5. As novas instâncias são baseadas nos dados reais disponibilizados por Dang et al. (2023), porém, evitando-se as deficiências identificadas nesta revisão da literatura.

3 Fundamentação Teórica

Nesse capitulo são formalizados os conceitos do SSP, da sua versão mais simples até sua versão específica tratada nesta monografia. Adicionalmente, é apresentada a base conceitual do método de resolução, o revenimento paralelo, com descrição de suas etapas de funcionamento.

3.1 O Problema de Minimização de Troca de Ferramentas

O SSP foi primeiramente abordado por Tang e Denardo (1988). Em sua versão uniforme, o problema pode ser descrito seguindo as seguintes especificações, de acordo com Calmels (2019):

- O conjunto de tarefas deve ser processado em uma única máquina;
- Os espaços reservados para cada ferramenta individualmente (*slots*) no *magazine* da máquina são idênticos e cada ferramenta requer apenas um espaço;
- Apenas uma ferramenta pode ser trocada por vez e os tempos de troca de ferramentas são constantes e iguais para todas as ferramentas;
- O conjunto de tarefas e o subconjunto de ferramentas necessárias para cada tarefa são conhecidos antecipadamente;
- O número de ferramentas exigido por qualquer tarefa é menor ou igual à capacidade do *magazine* de ferramentas; e
- As ferramentas não quebram nem se desgastam.

Dessa forma, o SSP pode ser dividido em dois subproblemas: determinar uma permutação das tarefas na qual seja minimizada a quantidade de trocas de ferramentas no *magazine* da máquina. Esse subproblema pertence à classe NP-Difícil. O segundo subproblema consiste em, dada uma permutação das tarefas, determinar o plano de trocas de ferramentas, ou seja, determinar quais ferramentas estarão no *magazine* da máquina em cada estágio da produção. Esse problema pode ser resolvido em tempo determinístico polinomial pelo algoritmo *Keep Tool Neeeded Soonest* (KTNS) (TANG; DENARDO, 1988) ou *Greedy Pipe Construction Algorithm* (GPCA) (CHERNIAVSKII; GOLDENGORIN, 2022).

Formalmente, o SSP uniforme pode ser descrito da seguinte maneira: dada uma máquina flexível com capacidade para armazenar C ferramentas em seu *magazine*, um conjunto de tarefas $T = \{1, \ldots, n\}$, um conjunto de ferramentas $F = \{1, \ldots, m\}$, o subconjunto de ferramentas

F_i $(F_i \in F)$ necessário para processar a tarefa i $(i \in T)$, o objetivo é determinar uma permutação dos elementos de T no qual o número de trocas de ferramentas seja minimizado. Na presente monografia, o problema abordado expande o SSP original com as seguintes especificidades, determinadas por Dang et al. (2023):

Múltiplas máquinas: As tarefas podem ser processadas por qualquer uma das múltiplas máquinas idênticas e cada máquina possui uma cópia própria do conjunto de ferramentas.

Horizonte de planejamento: Representa o prazo total para o processamento de tarefas. É considerado um horizonte de planejamento de 7 dias. Tarefas não finalizadas dentro do horizonte do plano podem acarretar em penalidades.

Período de produção não supervisionado: É determinado um período na produção em que as máquinas operam sem supervisão humana. Trocas de ferramentas não são permitidas nesse período, assim, todas as ferramentas requeridas devem estar presentes no *magazine* antes do início do período não supervisionado. É relevante evidenciar que o período não supervisionado ocorre sempre no fim do dia, ou seja, considerar um período não supervisionado de 12 horas significa reservar as 12 últimas horas do dia como não supervisionadas.

Trocas de ferramentas: As trocas não causam atraso na produção, mas possuem um custo associado. Quando ocorrem trocas de ferramentas em um determinado estágio da produção, elas pertencem a uma chamada *instância* de troca de ferramentas. Para cada instância de trocas existe um custo fixo e um custo variável. O custo fixo é associado a ocorrência de uma instância, e o custo variável é associado ao número de ferramentas trocadas em determinada instância. De acordo com Dang et al. (2023) o valor ideal para o custo fixo e variável são \$1 e \$10, respectivamente. É importante ressaltar que uma troca de ferramentas só acontece quando uma ferramenta necessária para uma operação não está presente na mesma máquina que processa a operação anterior. Dessa forma o carregamento inicial de ferramentas na máquina não representa uma troca.

Tarefas prioritárias: Cada tarefa pode ser *prioritária* ou *regular*. As tarefas prioritárias possuem um custo associado caso não sejam finalizadas dentro do horizonte de planejamento; para cada tarefa prioritária ou regular finalizada é atribuído um prêmio. De acordo com Dang et al. (2023) o prêmio por finalizar tarefas e o custo por não finalizar uma tarefa prioritária é de \$30.

Tarefas reentrantes: Tarefas podem possuir reentrância, ou seja, podem ter que ser processada mais de uma vez para serem finalizadas, i.e., possuem mais do que uma operação. Em termos práticos, significa que a tarefa é essencialmente duplicada, com o mesmo rótulo de prioridade e conjunto de ferramentas. Entretanto, o tempo de processamento pode ser diferente a cada operação. No problema abordado, tarefas podem ter reentrância de no máximo duas vezes.

Este é um problema NP-difícil por se tratar de um caso especial do SSP, o qual já é reconhecido como NP-difícil (CRAMA et al., 1994). O objetivo desta variante do SSP é a maximização do lucro, ou seja, a diferença entre o prêmio obtido pelo processamento das tarefas prioritárias, o custo das trocas de ferramentas (fixo mais variável) e a penalidade das tarefas prioritárias não finalizadas. A Figura 3.1 apresenta a relação entre prêmio e penalidade pela conclusão ou não das tarefas.

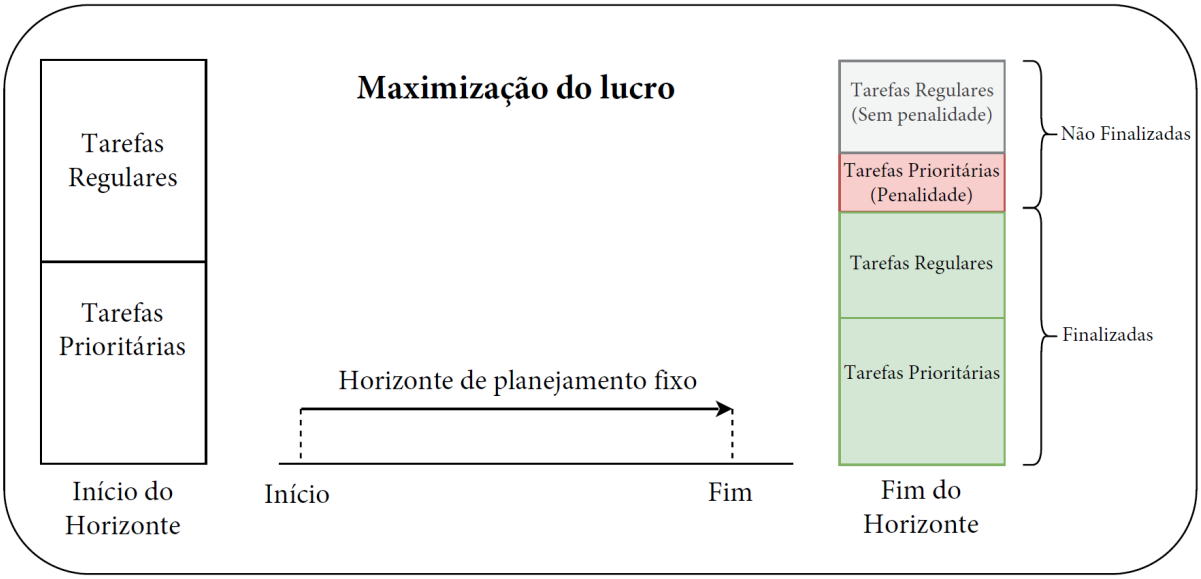

Figura 3.1 – Maximização do lucro com penalidades. Adaptado de (DANG et al., 2023)

A Tabela 3.1 apresenta um exemplo de uma instância para o problema abordado. Tarefas possuem tempo de processamento, conjunto de ferramentas e rótulo de prioridade ou regularidade. Uma tarefa pode possuir uma ou duas operações, o que representa a reentrância. O conjunto de ferramentas faz referência a Tabela 3.2, na qual há um identificador para cada conjunto de ferramentas e a composição do conjunto de ferramentas em si, indicada pelos índices das ferramentas. No problema abordado, existe exatamente um exemplar de cada ferramenta por máquina, ou seja, mesmo que dois conjuntos contenham a mesma ferramenta, há apenas uma cópia disponível em cada máquina.

Tabela 3.1 – Exemplo instância SSP.

Tarefa	Operação	Tempo de Processamento	Conjunto de Ferramentas	Prioridade
1	1	3	F_1	Prioritário
1	2	5	F_1	Prioritário
2	1	7	F_2	Regular
3	1	6	F_3	Regular
3	2	8	F_3	Regular
4	1	4	F_2	Prioritário
4	2	9	F_2	Prioritário
5	1	6	F_4	Regular
6	1	10	F_5	Regular
7	1	5	F_1	Prioritário

Na instância em questão, o conjunto de todas as ferramentas é dado por $F = \{1, \ldots, 20\}$. É importante ressaltar a interseção entre os conjunto de ferramentas: os conjuntos F_2 e F_5, por exemplo, possuem o seguinte conjunto de ferramentas em comum: $F_2 \cap F_5 = \{15, 16, 17, 18\}$. Esse fato indica um potencial benefício em sequenciar estas tarefas próximas uma das outras a fim de diminuir o número de trocas de ferramentas.

Tabela 3.2 – Conjuntos de ferramentas para exemplo de instância SSP.

Conjunto de Ferramentas	Ferramentas	Tamanho
F_1	{1,2,3,4,5}	5
F_2	{12, 13, 14, 15, 16, 17, 18}	7
F_3	{4, 5, 8, 9, 10, 11, 12, 13}	8
F_4	{5, 6, 7}	3
F_5	{15, 16, 17, 18, 19, 20}	6

A Figura 3.2 representa o gráfico de Gantt de uma possível solução para o exemplo considerando um tempo não supervisionado de 12 horas (720 minutos) e um horizonte de planejamento de 2 dias (2880 minutos). São consideradas duas máquinas na produção e a capacidade do *magazine* para ambas é dada pelo tamanho do maior conjunto de ferramentas, ou seja, 8 ferramentas. A fim de identificar as tarefas, considera-se um par ordenado (*id*, *op*) em que *id* identifica a tarefa e *op* identifica a operação. A permutação $S_1 = [(1,1), (1,2), (2,1), (3,1), (3,2)]$ determina a sequência das tarefas na máquina 1 e a permutação $S_2 = [(4,1), (4,2), (5,1), (6,1)]$ determina a sequência de tarefas na máquina 2.

A Tabela 3.3 apresenta as ferramentas no *magazine* de cada máquina durante cada estágio da produção, bem como o número de trocas de ferramentas. É importante ressaltar que a inserção das ferramentas do primeiro estágio não são contadas como trocas. Considerando um custo de \$1 para o custo variável e \$10 para o custo fixo, o custo total das trocas de ferramentas para

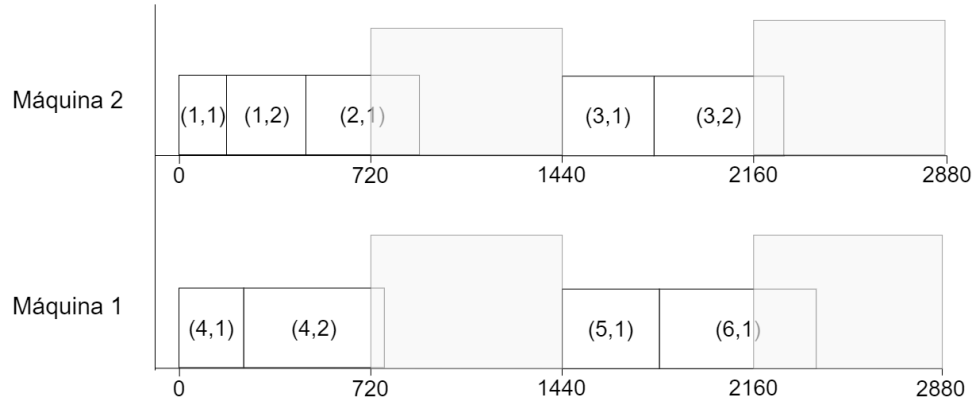

Figura 3.2 – Gráfico de Gantt para a primeira solução do exemplo.

o processamento das tarefas na máquina 1 é dado por $(4 \times\ 1 + 10) + (5 \times\ 1 + 10) = 29$. O mesmo pode ser feito para a máquina 2: $(2 \times\ 1 + 10) + (2 \times\ 1 + 10) = 24$.

Tabela 3.3 – Sequenciamento das ferramentas para a primeira solução de exemplo.

	Estágio de Produção	Ferramentas no *magazine*	Número de Trocas
Máquina 1	(1, 1)	{1, 2, 3, 4, 5, 12, 13, 14}	-
	(1, 2)	{1, 2, 3, 4, 5, 12, 13, 14}	0
	(2, 1)	{4, 12, 13, 14, 15, 16, 17, 18}	4
	(3, 1)	{4, 5, 8, 9, 10, 11, 12, 13}	5
	(3, 2)	{4, 5, 8, 9, 10, 11, 12, 13}	0
Máquina 2	(4, 1)	{5, 12, 13, 14, 15, 16, 17, 18}	-
	(4, 2)	{5, 12, 13, 14, 15, 16, 17, 18}	0
	(5, 1)	{5, 6, 7, 12, 15, 16, 17, 18}	2
	(6, 1)	{5, 6, 15, 16, 17, 18, 19, 20}	2

Dado que 9 tarefas foram finalizadas dentro do horizonte de planejamento e que a tarefa (7,1) é prioritária e não foi finalizada, considerando um prêmio de \$30 por finalizar tarefas e um custo também de \$30 por não finalizar tarefas prioritárias, o valor da solução proposta é dado por $(9 \times\ 30) - (30 \times\ 1) - (29) - (24) = 187$.

A Figura 3.3 apresenta o gráfico de Gantt para uma segunda solução da instância de exemplo. É possível observar que, alternando a ordem das tarefas, foi possível processá-las todas dentro do horizonte de planejamento.

A Tabela 3.4 exibe a quantidade de trocas de ferramentas para cada máquina na segunda solução. Observa-se que houve um total de 10 trocas em 3 ocasiões distintas, resultando em um custo total de $\$40$. Nesta abordagem, todas as tarefas foram concluídas, resultando em um prêmio de $\$300$. Portanto, o valor da nova solução é de $\$260$, representando uma melhoria de cerca 40% em relação à solução anterior.

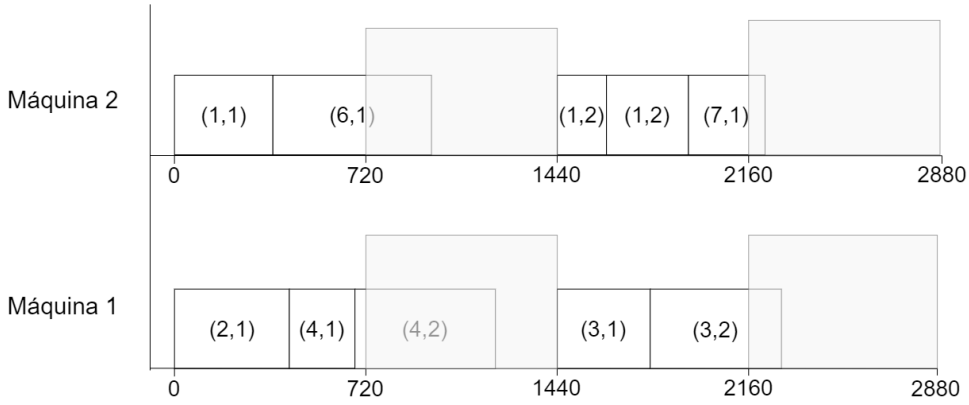

Figura 3.3 – Gráfico de Gantt para a segunda solução do exemplo.

Tabela 3.4 – Sequenciamento das ferramentas para a segunda solução do exemplo.

	Estágio de Produção	Ferramentas no *magazine*	Número de Trocas
Máquina 1	(2, 1)	{4,12,13,14,15,16,17,18}	-
	(4, 1)	{4,12,13,14,15,16,17,18}	0
	(4, 2)	{4,12,13,14,15,16,17,18}	0
	(3, 1)	{4,5,8,9,10,11,12,13}	5
	(3, 2)	{4,5,8,9,10,11,12,13}	0
Máquina 2	(5, 1)	{5,6,7,15,16,17,18,19}	-
	(6, 1)	{5,6,15,16,17,18,19,20}	1
	(1, 1)	{1,2,3,4,5,6,15,16}	4
	(1, 2)	{1,2,3,4,5,6,15,16}	0
	(7, 1)	{1,2,3,4,5,6,15,16}	0

3.2 Revenimento paralelo

O PT, também referido na literatura como *replica exchange*, reproduz o processo de revenimento, oriundo da metalurgia. Este é um método amplamente utilizado nas áreas de simulação em físico-química (EARL; DEEM, 2005) e biologia (HANSMANN, 1997). Entretanto, poucos trabalhos utilizaram o PT com aplicação em otimização.

Na ciência de materiais e metalurgia, revenimento é o processo de aquecer um metal até um certa temperatura por certo tempo e deixá-lo resfriar naturalmente no ambiente. Reaquecer o material em baixas temperaturas alivia tensões internas no material e reduz a fragilidade. Reaquecer em temperaturas mais altas implica em reduzir a dureza em troca de mais elasticidade e plasticidade (ALMEIDA, 2024). O método computacional revenimento paralelo estabelece uma analogia entre o processo metalúrgico e o processo de solução de um problema de otimização. No processo metalúrgico, há uma relação entre dureza e tenacidade, já no processo de solução de problemas de otimização, há uma relação entre a diversificação e intensificação da busca (ALMEIDA, 2024).

A fim de compreender o PT, primeiro devem ser entendidos os processos de Monte Carlo e amostragem iterativa com cadeias de Markov. O termo Monte Carlo surgiu no laboratório de desenvolvimento e pesquisa de Los Alamos, durante a criação da primeira bomba atômica (METROPOLIS et al., 1953). Dada a natureza secreta do projeto, os cientistas Nicholas Metropolis e Stanislaw Ulam sugeriram o nome Monte Carlo em referência a um grande cassino onde um tio de Stanislaw costumava apostar. Antes de se cunhar o nome Monte Carlo, o método era conhecido apenas como amostragem estatística. O termo Monte Carlo se refere a algoritmos que fazem uso de amostragem aleatória para resolver problemas.

Para problemas de otimização, o processo de Monte Carlo se dá pela simulação de diversas soluções e, no caso de problemas de maximização, escolher a de valor máximo entre todas as soluções avaliadas. Esse processo tende a convergir para o ótimo quando a quantidade de soluções exploradas é suficientemente grande, ou seja, o espaço de busca foi suficientemente explorado. A fim de melhorar a velocidade e a convergência do método, foi desenvolvido o método de Monte Carlo com cadeias de Markov (ou Monte *Carlo Markov Chain*, MCMC). Nesse caso, as soluções não são mais geradas de forma independente, mas sim de forma que uma solução s seja transformada em outra solução s' por meio da aplicação de um movimento.

Uma cadeia de Markov é uma forma de modelar um sistema que se move entre diferentes estados com certa probabilidade. A ideia principal é que o próximo estado dependa apenas do estado atual. Isso torna as cadeias de Markov úteis para estudar processos aleatórios que se alteram ao longo do tempo. Uma cadeia de Markov pode ser visualizada como um grafo direcionado como ilustrado na Figura 3.4, em que os vértices representam os estados, os arcos representam as transições entre os estados e o peso dos arcos dizem respeito a probabilidade de transição entre os estados.

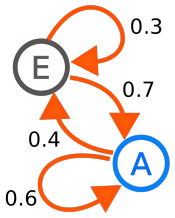

Figura 3.4 – Exemplo de uma cadeia de Markov. Fonte: Extraído de <https://en.wikipedia.org/wiki/Markov_chain>. Acessado em 20/05/2024.

A probabilidade do sistema se encontrar em um estado X_{n+1}, dado X_n como o estado anterior, é representada por $P\left(X_{n+1} = x \mid X_n = x_n\right)$. Tomando o exemplo da Figura 3.4, para determinar a probabilidade de o sistema se encontrar no estado E, basta saber qual o estado anterior. Considerando o estado anterior como A, temos $P\left(X_{n+1} = E \mid X_n = A\right) = 0,4$. Caso o estado anterior seja o E, temos $P\left(X_{n+1} = E \mid X_n = E\right) = 0,3$. O MCMC utiliza as cadeias de Markov para gerar soluções dependentes entre si, a partir de pequenas modificações entre uma solução e outra.

O Algoritmo de Metropolis é baseado no MCMC e é diretamente utilizado pelo PT. O algoritmo de Metropolis se baseia em construir uma cadeia de Markov com determinado comprimento, em que os estados são soluções e as transições representam mudanças de uma solução para outra. O resultado do algoritmo é a melhor solução encontrada. É utilizado o conceito de temperatura para determinar a aceitação ou não de uma solução. A temperatura é representada por uma constante T e é utilizada na Equação 3.1, conhecida como função de distribuição de Boltzmann

$$P\left(\Delta E, T\right) = e^{-\frac{\Delta E}{T}} \tag{3.1}$$

em que e corresponde à constante conhecida como número neperiano, T é a constante de temperatura definida previamente e ΔE diz respeito a degradação da solução e pode ser expandido para $f\left(s'\right) - f\left(s\right)$, em que f corresponde a uma função de avaliação, s é a solução atual e s' é a nova solução.

Quando a temperatura T é alta em relação ao ΔE a aceitação de uma solução com degradação é provável já que a constante T domina as outras variáveis. Já em temperaturas mais baixas, a aceitação de soluções piores tem menos chance de acontecer. A variação da temperatura é o que garante ao PT o balanceamento entre diversificação e intensificação da exploração do espaço de busca.

O funcionamento geral do algoritmo de Metropolis pode ser observado no Algoritmo 1. O algoritmo recebe como entrada a temperatura T a ser utilizada para calcular a aceitação de soluções e um inteiro N_{kmax} que determina o comprimento máximo da cadeia de Markov a ser gerada. Na linha 1 a variável s_0 é inicializada com a solução inicial, a qual pode gerada aleatoriamente ou seguindo algum algoritmo. O laço de repetição entre as linhas 1 e 10 gera a cadeia de Markov, de comprimento N_{kmax}. Dentro do laço, na linha 4, é construída uma nova solução s' a partir da solução atual s. Na linha 5, é calculada o ΔE, ou degradação da solução. Na linha 6 é verificada se a nova solução s' é aceita a partir da função de Boltzmann, a qual considera a degradação da solução e a temperatura recebida como entrada do algoritmo. Caso a solução s' seja aceita, ela é considerada a solução a atual para a próxima iteração (linhas 6 e 7). Caso contrário, a solução atual será considerada novamente (linhas 8 e 9). Ao fim do algoritmo, a última solução aceita é retornada.

Algoritmo 3.1: Algoritmo de Metropolis

Entrada: Temperatura T e o comprimento da cadeia de Markov N_{kmax}.
Saída: Solução atual.

```
1  s_0 ← solução inicial;
2  k ← 0;
3  enquanto k ≤ N_kmax faça
4      s' ← q(s_k, s');
5      ΔE ← f(s') − f(s_k);
6      se aceita s' então
7          s_{k+1} ← s';
8      senão
9          s_{k+1} ← s_k;
10     k ← k + 1;
```

Uma visualização do algoritmo de Metropolis pode ser observada no fluxograma da Figura 3.5, que ilustra o processo do algoritmo de forma simplificada. A partir de uma solução inicial S_0, são geradas N_{kmax} soluções e a aceitação é realizada por uma probabilidade a depender da temperatura T.

O PT coordena diferentes constantes T_i do algoritmo de Metropolis. São construídas k réplicas de um sistema – no nosso caso, um problema de otimização combinatória – em que cada uma considera uma temperatura T_i e uma solução inicial associada. Cada réplica consiste em um MCMC independente das demais. Periodicamente, são propostas trocas de temperaturas entre réplicas de temperaturas adjacentes. A troca de temperatura entre duas réplicas ocorre de acordo com uma probabilidade definida pela equação

$$P\left(r_i \ \rightarrow \ r_j\right) = min\left[1, \ exp\left(\Delta\beta\Delta E\right)\right] \tag{3.2}$$

em que $\Delta\beta$ pode ser expandido para $\frac{1}{T_j} - \frac{1}{T_i}$, com T_i e T_j correspondentes às temperaturas de cada réplica i e j envolvidas na proposta de troca, ao passo que ΔE indica a diferença entre o valor de cada solução em cada réplica, i.e., $f\left(r_j\right) - f\left(r_i\right)$.

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 continuam as mesmas para r_i e r_j. Essa característica faz com que o sistema explore o espaço de busca de forma eficiente, gerando o equilíbrio entre diversificação, associado a temperaturas altas, em que soluções com degradação são mais prováveis de serem aceitas, e intensificação, associadas a temperaturas mais baixas, em que soluções piores são aceitas mais raramente.

O funcionamento geral do PT é ilustrado no Algoritmo 2. Como entrada, são recebidos um vetor de temperaturas T, um vetor de soluções iniciais S, um inteiro qT_{emp} que indica o número de tentativas de trocas de temperatura e um inteiro N_{kmax} que determina o tamanho da cadeia de Markov a ser gerada pelo algoritmo de Metropolis. O laço de repetição principal, entre

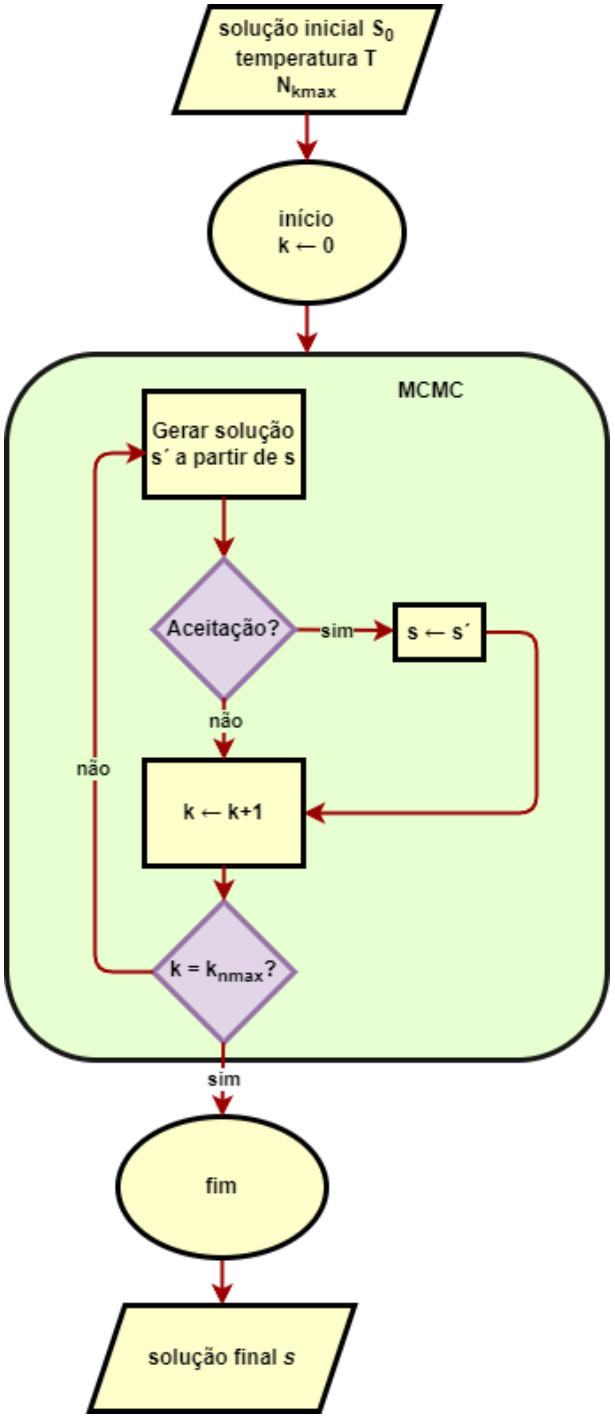

Figura 3.5 – Diagrama de execução do algoritmo de Metropolis.

as linhas 2 e 24, é executado até que todas as tentativas de trocas tenham sido realizadas. O laço de repetição interno, entre as linhas 4 e 23, itera sobre todas as réplicas executando o algoritmo de Metropolis por meio do laço de repetição entre as linhas 8 e 15. A aceitação de novas soluções é realizada caso a esta seja melhor ou de acordo com a Equação 3.1. Após executar o algoritmo de Metropolis, é realizada uma tentativa de troca de temperaturas entre duas réplicas de temperatura adjacentes seguindo a probabilidade definida pela Equação 3.2 (linhas 17 a 22). A saída do algoritmo é a última solução aceita para cada réplica, i.e., em cada uma das temperaturas.

Algoritmo 3.2: Algoritmo genérico do 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: Vetor de soluções S.

```
1  trocas_propostas ← 0;
2  enquanto trocas_propostas ≤ qT_emp faça
3      indice_replica ← 0;
4      enquanto indice_replica < tamanho(T) faça
5          s ← S[indice_replica];
6          t ← T[indice_replica];
7          k ← 0;
8          enquanto k ≤ N_kmax faça
9              s' ← vizinho(s);
10             ΔE ← f(s') − f(s);
11             se ΔE ≤ 0 então
12                 s ← s';
13             senão se aceita s' então
14                 s ← s';
15             k ← k + 1
16         S[indice_replica] ← s;
17         se indice_replica > 0 então
18             Δβ Δ E ← (1/T[indice_replica−1] − 1/T[indice_replica]) ×
                (f(S[indice_replica − 1]) − f(S[indice_replica]))
19             se Δβ Δ E ≥ 0 então
20                 troca(S[indice_replica], S[indice_replica − 1]);
21             senão se aceita a troca então
22                 troca(S[indice_replica], S[indice_replica − 1]);
23         indice_replica ← indice_replica + 1;
24     trocas_propostas ← trocas_propostas + 1;
```

A Figura 3.6 ilustra a execução do PT, em que cada réplica é disposta em um nível na horizontal e cada quadrado representa a execução do algoritmo de Metropolis a uma determinada temperatura fixa, ilustrada anteriormente na Figura 3.5. As temperaturas são fixas, e as réplicas se movimentam por elas. As setas na horizontal indicam que uma réplica manteve sua temperatura anterior e as setas na diagonal indicam que duas réplicas adjacentes trocaram de temperatura. Cada coluna na referida figura indica uma proposta de trocas de temperaturas entre réplicas adjacentes, totalizando qT_{emp} tentativas.

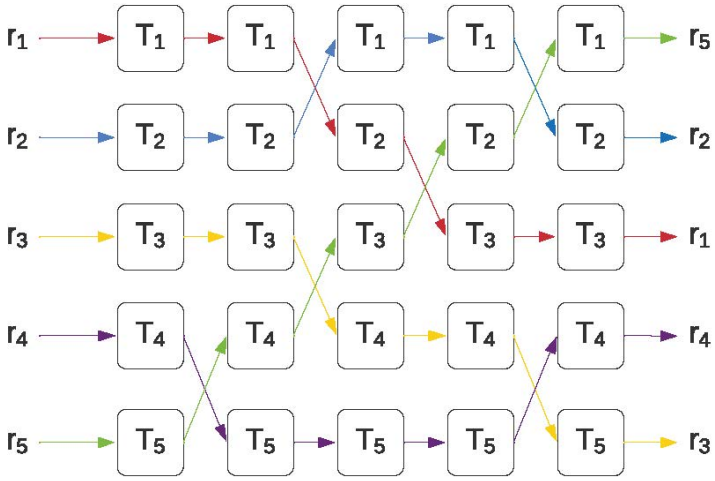

Figura 3.6 – Diagrama das trocas de temperaturas do algoritmo PT. Extraído de Almeida (2024).

A implementação do PT a ser utilizada na presente monografia foi disponibilizada via API[1] por Almeida (2024), o qual possui uma abordagem paralela para o PT. O paralelismo em CPU não é muito usual para implementações do PT, mas se mostrou promissor para resoluções de problema de otimização recentemente (ALMEIDA, 2024). Mesmo utilizando-se uma API, como a disponibilizada, ainda se faz necessário definir os componentes dependentes do problema e outros componentes complementares ao funcionamento do PT. Entre estes, citam-se:

Codificação: Estrutura de dados responsável por determinar a representação de uma solução. No caso do problema a ser tradado, consiste em uma permutação do conjunto original de tarefas em uma determinada estrutura de dados;

Decodificação: Função responsável por transformar a solução codificada em uma solução completa para o problema tratado;

Soluções iniciais: Seguindo a codificação da solução, é necessário um método que retorne uma solução inicial para cada réplica do problema, seja de forma aleatória ou seguindo algum critério estruturado;

Avaliação da solução: Função responsável por receber uma solução e retornar um valor numérico que indique a qualidade da solução;

Geração de soluções vizinhas: Componente responsável por transformar uma solução s em uma s', para utilização pelas cadeias de Markov.

Além de definir componentes e estruturas de dados para o problema em específico, também é necessário definir os parâmetros do PT via experimentos preliminares. Entre estes parâmetros, cita-se:

[1] Disponível em https://github.com/ALBA-CODES/PTAPI/. Acessado em 17 de setembro de 2024

- Valor da temperatura inicial;
- Valor da temperatura final;
- Quantidade de réplicas do algoritmo;
- Comprimento das cadeias de Markov;
- Quantidade de propostas de trocas;
- Método da distribuição inicial das temperaturas;
- Método de atualização automática do valor das temperaturas;
- Taxa de atualização das temperaturas.

Os próximos capítulos aprofundam-se no desenvolvimento da adaptação do PT básico para abordar o problema objeto de estudos desta monografia. Isso inclui a definição dos componentes e parâmetros requeridos para o funcionamento adequado do PT. Após a definição da implementação do PT, experimentos preliminares e experimentos extensivos para avaliar a eficácia dessa adaptação são reportados, no intuito de avaliar como o PT se comporta em termos de eficiência e qualidade das soluções geradas.

4 Desenvolvimento

Neste capítulo, é apresentada uma análise aprofundada dos dados disponíveis na literatura, bem como a descrição da criação de novas instâncias. Adicionalmente, serão descritas as características e adaptações realizadas na API utilizada.

4.1 Análise dos dados

Os dados reais advindos de uma indústria de manufatura de precisão, disponibilizados por Dang et al. (2023), são divididos em duas categorias principais: conjuntos de ferramentas e tarefas/máquinas. A categoria de conjuntos de ferramentas inclui uma lista de 3464 elementos, cada um contendo diversas ferramentas. As informações sobre tarefas/máquinas estão divididas em três outras coleções distintas:

- 2M376: contém 250 tarefas e 2 máquinas;
- 6M1201: contém 750 tarefas e 6 máquinas;
- 6M1401: contém 1000 tarefas e 6 máquinas.

Para cada tarefa nessas coleções, são fornecidas informações sobre o conjunto de ferramentas necessário e o tempo de processamento correspondente. No entanto, não há informações específicas sobre reentrância e prioridade de cada tarefa. Outros dados relacionados ao problema, descritos pelos autores durante o artigo, são como segue:

- Penalidade por tarefa prioritária não finalizada: $30;
- Penalidade por instância de troca de ferramentas: $10;
- Penalidade por troca de ferramenta: $1;
- Bônus por tarefa finalizada: $30;
- Horizonte de planejamento: 7 dias;
- Duração do período de não supervisão: 12 horas, na segunda metade de cada dia;
- Capacidade do *magazine*: 80 ferramentas.

A fim de analisar os dados e revelar possíveis falhas, uma primeira filtragem dos conjuntos de ferramentas consistiu em remover todos os conjuntos que não foram referenciados por nenhuma

tarefa. Esse processo reduziu o número de conjuntos de ferramentas de 3464 para apenas 382. Desta forma, 3082 dos conjuntos de ferramentas utilizados na indústria parceira não foram utilizados nas instâncias geradas, o que representa 88,98% do total.

A versão do SSP em questão estabelece que todas as ferramentas para uma dada tarefa devem caber no *magazine* das máquinas, assim, conjuntos de ferramentas maiores do que a capacidade do *magazine* não devem ser permitidos. Apesar de haver a possibilidade de alguns conjuntos de ferramentas serem subconjuntos de outros, dada a natureza real dos dados, essa não é uma propriedade desejável. Instâncias com essa característica facilitam a solução do problema, pois basta agrupar as tarefas de tais conjuntos de ferramentas e sequenciá-los de maneira contígua, evitando trocas de ferramentas.

Dos conjuntos de ferramentas não referenciados, 30 possuem mais do que 80 ferramentas e 22 possuem 0 ferramentas. Dos 382 conjuntos de ferramentas referenciados, 3 possuem cardinalidade maior que 80 ferramentas. Dos 3082 conjuntos de ferramentas não referenciados, excluindo-se aqueles que são subconjuntos de outros e conjuntos contendo 0 ou mais que 80 ferramentas, sobram 1265 conjuntos de ferramentas únicos.

Ao analisar as coleções de tarefas individualmente, foram filtradas tarefas que utilizam conjuntos de ferramentas que são subconjuntos de outros na mesma coleção. Foi observado que diversas tarefas utilizavam exatamente o mesmo conjunto de ferramentas, o que também não é uma característica desejável. Essa filtragem resultou em 50 tarefas (2M376), 143 tarefas (6M1201) e 153 tarefas (6M1201), o que representa reduções de 80%, 81% e 84,7% nos conjuntos originais.

Conforme mencionado anteriormente no Capítulo 2, pela Tabela 5 de Dang et al. (2023), observa-se que, para diversas instâncias, é encontrado o valor ótimo para a função objetivo sem penalidade de troca de ferramentas. O valor ótimo para uma instância pode ser encontrado multiplicando o bônus de tarefas finalizadas pela quantidade de tarefas na instância. Esse valor indica que não houve nenhuma penalidade por troca de ferramentas nem por tarefa prioritária não finalizada. O mesmo fato ocorre para os diferentes métodos comparados nos experimentos computacionais do referido artigo, em diferentes tamanhos de instâncias. Esse fato indica que todas as tarefas puderam ser processadas dentro do horizonte de planejamento e não houve a necessidade de nenhuma troca de ferramenta, o que sugere a facilidade de resolver tais instâncias.

Essas inconsistências levaram o autor desta monografia a entrar em contato com os autores de Dang et al. (2023). O contato foi estabelecido e os autores mostraram boa vontade em esclarecer dúvidas e disponibilizar as instâncias utilizadas. No entanto, as respostas foram insuficientes para reproduzir as instâncias, uma vez que se fez uso de escolhas aleatórias para decidir quais tarefas seriam prioritárias e quais seriam reentrantes. As instâncias originais não foram enviadas, apesar da disposição inicial.

Dada a considerável redução na quantidade de dados utilizáveis e a falta de informação

sobre prioridade e reentrância, tornou-se necessária a criação de novas instâncias. A próxima seção descreve este processo.

4.2 Novas instâncias

Novas instâncias foram criadas com base nos dados reais disponibilizados por Dang et al. (2023). A organização dos dados permanece a mesma para as novas instâncias, com a separação entre as tarefas e os conjuntos de ferramentas. Os dados referentes a penalidades, bônus, horizonte de planejamento, duração do período não supervisionado e capacidade do *magazine* permaneceram inalterados.

Foram criados dois conjuntos de 33 novas instâncias cada, baseadas nos 1265 conjuntos de ferramentas filtrados, disponibilizados para uso da comunidade científica[1]. Desta forma, é garantido que as novas instâncias não terão as deficiências de conjuntos de ferramentas contidos em outros, conjuntos de ferramentas redundantes e nem conjuntos de ferramentas de cardinalidade igual a 0 ou maior que 80, citados anteriormente.

O primeiro conjunto é mais fiel ao proposto por (DANG et al., 2023) em relação a tarefas reentrantes, já que considera que, para uma mesma tarefa, as operações que a compõem possuem o mesmo conjunto de ferramentas. Já no segundo conjunto de instâncias, operações de uma mesma tarefa possuem conjuntos de ferramentas diferentes. Esta versão alternativa foi introduzida a fim de aumentar o desafio na resolução do problema, uma vez que impede o agrupamento de operações de uma mesma tarefa.

As instâncias de ambos os conjuntos de instâncias gerados foram divididas em 11 tamanhos, começando em 75 tarefas e incrementando até 1236. Para cada tamanho, variou-se a porcentagem de tarefas prioritárias em 25%, 50% e 75%. Os nomes dos arquivos descrevem os detalhes da instância, no seguinte formato:

$$n = 75, p = 0.24, r = 0.5.csv \tag{4.1}$$

em que n representa o número de tarefas na instância, p a porcentagem de tarefas prioritárias, e r representa a porcentagem de tarefas reentrantes.

O número de tarefas geradas com base nos dados disponibilizados é maior do que o original. Desta forma, tornou-se necessário gerar novos tempos de processamento para as novas tarefas. Estes dados foram gerados de forma pseudoaleatória da seguinte maneira: foram recuperados e armazenados todos os tempos de processamento das coleções originais de tarefas. Em seguida, foi construída uma distribuição com esses valores, e os novos tempos de processamento foram gerados a partir desta distribuição. Essa abordagem garantiu que os novos valores não destoassem dos dados originais coletados por Dang et al. (2023).

[1] https://github.com/mateus2k2/SSP

Nestas novas instâncias, faz-se uma distinção clara entre tarefa e operação para indicar a reentrância. Uma tarefa pode conter mais de uma operação, indicando que essa tarefa possui reentrância. Como já estabelecido por Dang et al. (2023), tarefas podem conter no máximo uma reentrada, ou seja, podem conter no máximo duas operações. Desta forma, o número de operações por tarefa é explicitamente mencionado na instância.

A porcentagem de tarefas reentrantes para cada instância foi determinada com base na quantidade total de tarefas. Em comunicação privada, os autores de Dang et al. (2023) informaram que as porcentagens de tarefas reentrantes variavam conforme o tamanho das instâncias: 50% para 376 tarefas, 60% para 1201 tarefas e 40% para 1401 tarefas. Portanto, para as novas instâncias, as mesmas porcentagens mantidas, atribuídas de acordo com a proximidade do número de tarefas das instâncias em relação aos dados da recomendação dos autores de Dang et al. (2023).

4.3 Função de avaliação

Dada uma sequência de processamento de tarefas, a avaliação padrão consiste em determinar o plano ótimo para as trocas de ferramentas, o que pode ser feito pelo o algoritmo KTNS (TANG; DENARDO, 1988). Na prática, dada uma sequência de tarefas, o KTNS as avalia e determina a sequência em que as ferramentas serão necessárias. Mantendo uma lista de prioridades dinâmica, o algoritmo garante que as ferramentas requeridas sejam mantidas no *magazine*, enquanto aquelas não requeridas imediatamente são desconsideradas. Havendo a necessidade de trocas de trocas de ferramentas, são mantidas no *magazine* prioritariamente aquelas que serão utilizadas mais em breve. A implementação do KTNS utilizada na presente monografia foi adaptada da implementação de melhor desempenho encontrada na literatura, disponibilizada por Almeida, Lima e Carvalho (2023).

O problema tratado nesta monografia considera um período de produção não supervisionado e um horizonte de planejamento para calcular o valor da função objetivo de uma dada sequência de tarefas. Caso o processamento de tarefas ultrapasse o horizonte de planejamento, apenas as tarefas finalizadas até o término deste horizonte devem ser contabilizadas no bônus, enquanto as outras resultarão em penalidades. Além disso, se uma tarefa iniciar seu processamento durante o período não supervisionado e requerer que alguma ferramenta seja adicionada ao *magazine*, o início do processamento deverá ser postergado para o início do próximo período supervisionado.

As peculiaridades do problema em questão fazem com que o KTNS precise ser adaptado para considerar as características adicionais. A princípio, todas as tarefas a serem processadas em uma dada máquina são enviadas para o algoritmo. À medida que o plano de ferramentas é construído, o número de trocas de ferramentas e a quantidade de instantes de troca são acumulados. Além disso, o tempo de processamento é acumulado em um contador de tempo. Isso permite realizar verificações de período não supervisionado e de horizonte de planejamento. O algoritmo

termina quando todas as tarefas da solução são processadas ou quando o contador de tempo alcança o horizonte de planejamento. O valor da solução é então retornado: bônus das tarefas finalizadas, menos penalidade por tarefas prioritárias não finalizadas e trocas de ferramentas.

4.4 Solução inicial

A fase de solução inicial desempenha um papel crucial nas metaheurísticas, pois oferece uma base sobre a qual as melhorias subsequentes serão realizadas. A abordagem adotada para gerar soluções iniciais é criar uma permutação de todas as tarefas e suas respectivas operações, conforme o tamanho da instância. Por exemplo, considere uma instância de 7 tarefas e 9 operações, como o exemplo da Tabela 3.1. Cada tarefa é representada por um par ordenado (*id*, *op*) em que *id* identifica a tarefa e *op* identifica a operação. Uma possível permutação é $[(1,1),(6,1),(1,2),(7,1),(2,1),(4,1),(4,2),(3,1),(3,2)]$.

Cada permutação representa uma solução inicial distinta, que será posteriormente codificada e decodificada pelo PT. Esse processo permite que diferentes pontos de partida sejam explorados pela metaheurística, aumentando a diversidade das soluções e melhorando potencialmente a qualidade das soluções finais encontradas.

4.5 Codificação e decodificação

A codificação da solução é realizada por meio de um único arranjo com n itens, em que cada um deles corresponde ao índice de uma tarefa. Desta forma, trata-se o problema como se houvesse apenas uma máquina para processar todas as tarefas. A decodificação é realizada considerando um horizonte de planejamento m vezes maior que o original, em que m é o número de máquinas disponíveis para processar as tarefas. Dessa forma, o algoritmo KTNS é executado apenas uma vez, mesmo que haja mais de uma máquina. A divisão de tarefas entre as diferentes máquinas é realizada de maneira lógica, de acordo com os intervalos definidos pelo tamanho do horizonte de planejamento original.

A Figura 4.1 ilustra o processo de codificação e decodificação. Na parte superior, tem-se a solução codificada em um único arranjo. Na parte inferior, tem-se a solução decodificada considerando um horizonte de planejamento de 4 dias e considerando duas máquinas para o processamento das tarefas. Cada retângulo sólido representa uma tarefa sendo processada e é rotulado novamente como (*id*, *op*), em que *id* identifica a tarefa e *op* identifica a operação. Os retângulos opacos indicam os períodos de produção não supervisionada. O eixo x denota o tempo, em minutos, de 0 até 5760 (48 horas). A divisão das tarefas entre as máquinas é indicada pelas linhas verdes.

De acordo com a abordagem de decodificação, torna-se necessário criar um mecanismo que considere situações de inviabilidade. Por exemplo, uma tarefa que exija trocas de ferramen-

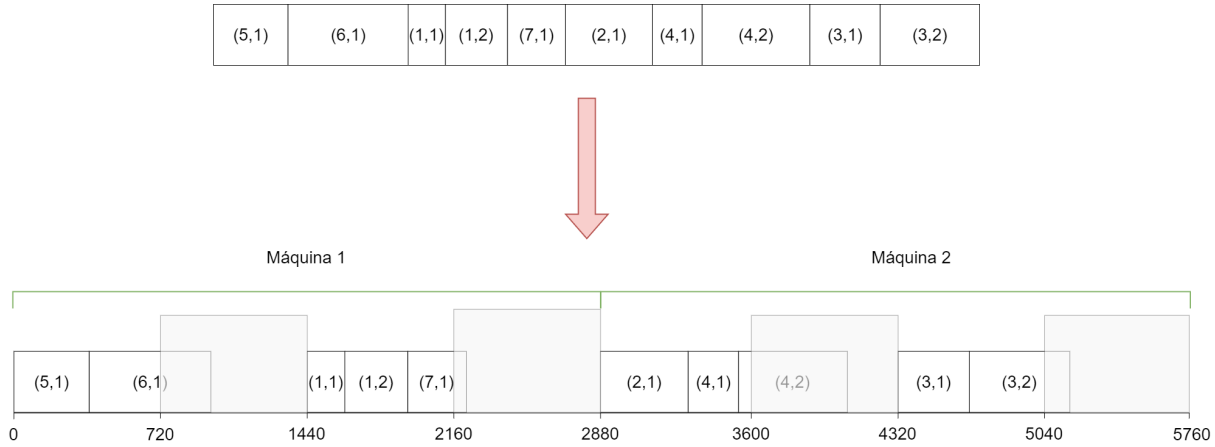

Figura 4.1 – Exemplo codificação e decodificação de uma solução.

tas durante o período não supervisionado, terá seu processamento postergado para o próximo período supervisionado, vide alocação da tarefa $(1, 1)$ no instante 1440. O mesmo ocorre caso o processamento de uma tarefa exceda o horizonte de planejamento para uma máquina, pois isso implicaria no processamento da tarefa começar em uma máquina e terminar em outra. Esses problemas são mitigados através de contadores de tempo e verificações periódicas durante a execução do KTNS adaptado.

4.6 Estruturas de vizinhança

A determinação de estruturas de vizinhança é um dos elementos fundamentais na busca local para problemas de sequenciamento de tarefas, permitindo explorar o espaço de soluções a partir de pequenas modificações em uma solução atual. Esta exploração visa gerar soluções similares, que podem melhorar a solução atual ou introduzir diversificação, possibilitando a fuga de ótimos locais e contribuindo para a eficácia de metaheurísticas e métodos semelhantes. As seguintes estruturas de vizinhança são consideradas:

2-opt: Dada duas posições definidas dentro da sequência de tarefas, inverte-se a ordem de todos os elementos no intervalo definido pelas posições. Especificamente na implementação utilizada, as posições são definidas aleatoriamente. Como pode ser observado na Figura 4.2, a solução $[(1, 1), (1, 2), (2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (5, 1), (6, 1), (7, 1)]$ se transforma em $[(1, 1), (1, 2), (2, 1), (4, 2), (4, 1), (3, 2), (3, 1), (5, 1), (6, 1), (7, 1)]$ após a definição das posições 3 e 6.

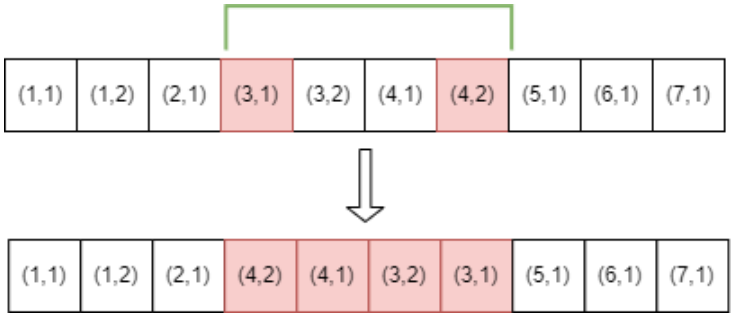

Figura 4.2 – Exemplo movimento 2-opt.

2-swap: Dada duas posições definidas dentro da sequência de tarefas, troca-se os dois elementos correspondentes às estas duas posições. Especificamente na implementação utilizada, as posições são definidas aleatoriamente. Seguindo essa estrutura, a solução $[(1,1),(1,2),(2,1),(3,1),(3,2),(4,1),(4,2),(5,1),(6,1),(7,1)]$ se transforma em $[(1,1),(1,2),(2,1),(4,2),(3,2),(4,1),(3,1),(5,1),(6,1),(7,1)]$ com observado na Figura 4.3 após a definição das posições 3 e 6.

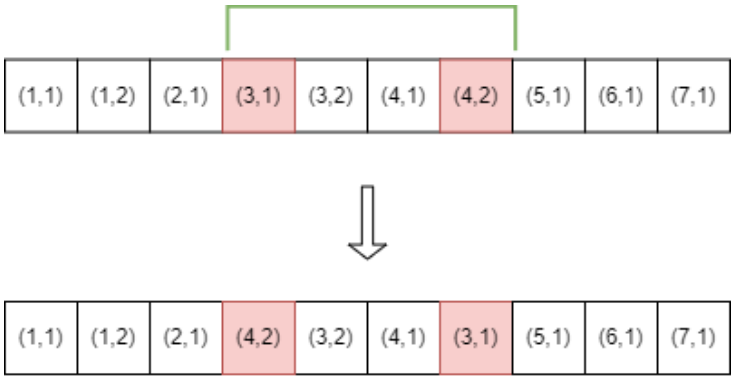

Figura 4.3 – Exemplo movimento 2-swap.

Inserção: Dada duas posições definidas dentro da sequência de tarefas, o elemento da primeira posição é inserido na segunda posição. Especificamente na implementação utilizada, as posições são definidas aleatoriamente. Como se observa no exemplo na Figura 4.4 a solução $[(1,1),(1,2),(2,1),(3,1),(3,2),(4,1),(4,2),(5,1),(6,1),(7,1)]$ se transforma em $[(1,1),(1,2),(2,1),(3,2),(4,1),(4,2),(3,1),(5,1),(6,1),(7,1)]$ após a definição das posições 3 e 6.

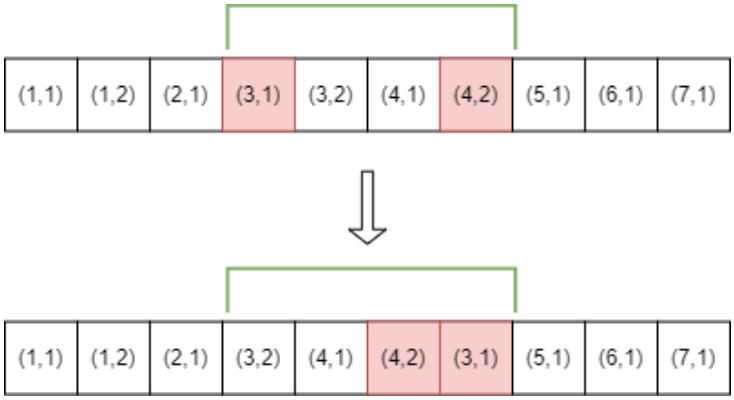

Figura 4.4 – Exemplo movimento inserção.

Com a implementação da função de avaliação, codificação e estrutura de vizinhança concluída, o próximo capítulo é dedicado à realização de experimentos preliminares.

5 Experimentos Computacionais

Neste capítulo, são apresentados os experimentos preliminares realizados com o intuito de validar a abordagem proposta para o problema em estudo. Além disso, foi realizada uma avaliação da consistência dos conjuntos de instâncias criados. Os experimentos computacionais foram realizados em um computador Intel Core i5-10400 CPU @ 2.90GHz com 8 cores e 16 GB de RAM sob o sistema operacional Ubuntu 21.10. O código foi implementado em C++, compilado com g++ 11.2.0 e opção de otimização -O3.

A Tabela 5.1 apresenta as características das instâncias geradas, como a quantidade de tarefas, a taxa de reentrância, a taxa de tarefas prioritárias e o número de máquinas. Foram gerados dois conjuntos de 33 instâncias com as mesmas configurações. Contudo, no primeiro conjunto as tarefas reentrantes utilizam sempre os mesmos conjuntos de ferramentas em todas as operações, enquanto no segundo conjunto, cada operação de uma tarefa reentrante exige um conjunto diferente de ferramentas, aumentando o nível de dificuldade das instâncias.

Tabela 5.1 – Características das instâncias.

Tarefas	Taxa de Prioritárias	Taxa de Reentrantes	Máquinas
75	{0,24; 0,51; 0,75}	0,5	2
212	{0,25; 0,50; 0,75}	0,4	2
228	{0,25; 0,50; 0,75}	0,6	2
399	{0,25; 0,50; 0,75}	0,5	2
499	{0,25; 0,50; 0,75}	0,5	6
600	{0,25; 0,50; 0,75}	0,5	6
699	{0,25; 0,50; 0,75}	0,5	6
800	{0,25; 0,50; 0,75}	0,6	6
899	{0,25; 0,50; 0,75}	0,6	6
1000	{0,25; 0,50; 0,75}	0,6	6
1236	{0,25; 0,50; 0,75}	0,6	6

5.1 Experimentos preliminares

O PT necessita que sejam definidos alguns parâmetros para a execução do algoritmo. Entretanto, os principais objetivos dos experimentos preliminares foram instrumentalizar o uso da API para abordar a versão do SSP tratada e também validar a consistência das instâncias criadas. Dessa maneira, os experimentos preliminares foram realizados com parâmetros não definidos metodologicamente, com os valores definidos como a seguir.

Temperatura inicial: 0,1;

Temperatura final: 0,5;

Numero de réplicas: 11;

Tamanho da cadeia de Markov: 400;

Numero de trocas de temperaturas: 75000;

Distribuição inicial das temperaturas: linear;

Tipo de ajuste da temperatura: igualar taxas de aceitação;

Taxa de ajuste da temperatura: 20000;

Vizinhança: 2-opt.

Os resultados obtidos para os conjuntos de instâncias I e II são apresentados respectivamente nas Tabelas 5.2 e 5.3. Nas referidas tabelas, o valor da solução objetivo foi dividido entre o componente de bônus por tarefas finalizadas e os três componentes de penalidade. Para cada instância, são indicados o número de tarefas finalizadas, o número de tarefas prioritárias não finalizadas, o número de instâncias de trocas de ferramentas, o número de trocas de ferramentas, o valor total da função de avaliação e o tempo de execução, expresso em minutos. Adicionalmente, a última coluna das tabelas apresenta um limite inferior para o tamanho do horizonte de planejamento, dado pelo somatório dos tempos de processamento de todas as tarefas dividido pelo número de máquinas. O resultado está expresso em dias.

Tabela 5.2 – Resultados Preliminares conjunto de instâncias I.

Instância	Tarefas finalizadas	Tarefas prioritárias não finalizadas	Instâncias de troca	Trocas de ferramentas	Valor solução	Tempo de execução	Limite inferior para horizonte
1	75	0	34	528	1382	2,80	2,40
2	75	0	34	528	1382	3,25	2,54
3	75	0	34	531	1379	3,24	2,18
4	162	0	95	1006	2904	16,01	6,94
5	166	9	96	1074	2676	16,17	6,63
6	165	18	109	1241	2079	15,95	6,79
7	165	0	94	952	3058	15,79	6,95
8	179	11	108	1391	2569	15,97	6,86
9	170	26	109	1299	1931	15,61	7,32
10	171	17	117	1126	2324	22,89	12,4
11	191	66	129	1330	1130	23,23	12,4
12	196	131	132	1309	-679	24,91	12,2
13	410	7	314	3304	5646	57,99	5,24
14	428	6	330	3836	5524	56,82	5,33
15	431	34	328	3828	4802	57,70	5,27
16	421	16	321	3536	5404	69,80	6,40
17	427	48	335	3553	4467	70,55	6,34
18	429	99	332	3714	2866	70,48	6,37
19	427	30	334	3423	5147	79,90	7,26
20	436	77	348	3698	3592	78,90	7,39
21	454	154	365	4231	1119	79,66	7,36
22	447	37	357	3815	4915	88,06	8,23
23	430	122	348	3704	2056	86,00	8,40
24	455	235	374	4248	-1388	85,61	8,52
25	442	52	361	3729	4361	98,07	9,17
26	443	174	361	3657	803	94,74	9,27
27	451	294	381	3960	-3060	97,40	9,52
28	442	73	363	3507	3933	101,72	10,3
29	456	211	376	4023	-433	103,44	10,3
30	449	379	371	4136	-5746	101,64	10,9
31	448	139	376	3707	1803	117,54	13,0
32	460	322	391	3748	-3518	119,10	13,3
33	458	537	401	4176	-10556	115,85	13,2

Tabela 5.3 – Resultados Preliminares conjunto de instâncias II.

Instância	Tarefas finalizadas	Tarefas prioritárias não finalizadas	Instâncias de troca	Trocas de ferramentas	Valor solução	Tempo de execução	Limite inferior para horizonte
1	75	0	51	777	963	3,40	2,320
2	75	0	52	618	1112	2,90	2,225
3	75	0	56	809	881	3,44	2,355
4	150	1	121	1120	2140	14,74	6,835
5	145	8	114	1216	1754	13,80	7,229
6	148	34	123	1245	945	14,84	7,065
7	164	1	135	1159	2381	16,40	6,930
8	160	8	130	1340	1920	16,92	7,040
9	169	27	130	1320	1640	16,21	6,512
10	180	21	146	1257	2053	27,60	12,37
11	183	68	152	1419	511	27,30	12,98
12	182	144	154	1572	-1972	27,14	13,30
13	395	3	355	3399	4811	62,50	5,326
14	416	10	383	3962	4388	63,60	5,120
15	414	37	372	3985	3605	64,02	5,240
16	402	13	371	3482	4478	74,63	6,501
17	425	45	393	3912	3558	76,98	6,366
18	441	90	398	3985	2565	83,68	6,123
19	439	24	400	4087	4363	91,88	7,208
20	426	89	391	4011	2189	85,74	7,544
21	442	145	410	3978	832	86,45	7,445
22	431	40	395	3951	3829	100,84	8,268
23	454	124	419	3943	1767	98,13	8,362
24	458	213	424	4554	-1444	101,37	8,316
25	429	62	392	3727	3363	107,22	9,517
26	429	179	391	3845	-255	108,97	9,849
27	453	287	415	4146	-3316	109,76	9,345
28	442	71	409	4004	3036	114,92	10,66
29	445	206	411	4008	-948	119,63	10,51
30	463	350	429	4292	-5192	122,16	10,60
31	452	126	417	3878	1732	136,15	13,00
32	448	316	415	3732	-3922	137,12	13,29
33	466	529	433	4393	-10613	140,79	13,00

Para instâncias pequenas, o algoritmo foi capaz de finalizar todas as tarefas prioritárias rapidamente, mesmo sem os parâmetros do PT ajustados adequadamente para o problema especificado. Para instâncias grandes, pôde-se observar uma degradação do valor das soluções e um aumento do tempo de execução à medida em que a taxa de tarefas prioritárias aumenta.

A análise dos resultados revelou que o horizonte de planejamento de 7 dias é muito curto para instâncias com muitas tarefas, impactando o valor da solução, uma vez que muitas tarefas prioritárias não são finalizadas. Isto pode ser verificado pelos valores negativos para algumas instâncias, o que indica que o número de tarefas prioritárias não finalizadas é excessivamente alto. Como consequência, um ajuste da duração do horizonte de planejamento baseada no número de tarefas e máquinas é necessário para que algumas das instâncias se tornem consistentes. Esta tarefa integrará a parte final deste estudo.

A execução dos experimentos preliminares revelou um significativo gargalo no desem-

penho do algoritmo, que foi identificado, após análise do código, como decorrente do uso do algoritmo KTNS. Testes exploratórios foram conduzidos utilizando uma versão adaptada do algoritmo GPCA, visando reduzir o tempo de execução. A aplicação do GPCA na construção do plano de trocas será investigada de forma mais aprofundada no decorrer da segunda parte deste estudo.

6 Plano de Atividades Restantes

Como planejamento para conclusão do projeto de Monografia, na Tabela 5.1 são apresentadas informações sobre atividades subsequentes, a serem realizadas na disciplina Monografia II.

Tabela 6.1 – Planejamento de atividades para Monografia II.

Atividades	Mês 1	Mês 2	Mês 3	Mês 4
Ajuste das instâncias	X			
Adaptação da API do PT para a versão específica do SSP	X	X		
Implementação do modelo matemático		X		
Realização de experimentos computacionais		X	X	
Testes para calibração de parâmetros		X	X	
Descrição dos experimentos			X	
Análise dos experimentos			X	X
Conclusão da Monografia				X

As atividades basicamente se concentram em ajustar as instâncias e implementar os componentes necessários para utilizar a API do PT. Adicionalmente, pretende-se adaptar o algoritmo GPCA para substituir o KTNS, já que análises preliminares indicaram uma melhora significativa no tempo de execução. O método, então, será submetido a experimentos computacionais e análises adicionais. O método será avaliado utilizando-se as novas instâncias criadas. Nestes experimentos, serão gerados os primeiros resultados comparáveis para o problema em questão na literatura e, como *benchmark* externo, pretende-se implementar o modelo de programação matemática proposto por Dang et al. (2023).

7 Conclusão

Neste estudo, foram abordadas a definição formal, uma revisão detalhada da literatura e a descrição de uma meta-heurística para abordagem de uma variante do problema de minimização de trocas de ferramentas. Observou-se que este é um problema de relevância significativa no contexto prático industrial, uma vez que a redução das trocas de ferramentas, entre outros objetivos, pode levar a uma diminuição substancial dos custos operacionais e aumentar a eficiência produtiva. Além disso, a revisão da literatura revelou que existe apenas uma única abordagem para a resolução desta versão específica do problema, havendo espaço considerável para contribuições inovadoras, especialmente quando se trata de transparência e reprodutibilidade de experimentos. Os experimentos computacionais preliminares demonstraram que a meta-heurística proposta oferece resultados promissores. Trabalhos futuros se concentrarão na realização de novos experimentos com diferentes componentes algorítmicos para validar e fortalecer os achados deste estudo.

Referências

AGNETIS, A.; DETTI, P.; PRANZO, M.; SODHI, M. Sequencing unreliable jobs on parallel machines. *J. Scheduling*, v. 12, p. 45–54, 02 2009.

ALMEIDA, A. L. B. *Revisitando o Revenimento Paralelo: Computação de Alto Desempenho e Aplicação em Pesquisa Operacional*. Tese (Tese de doutorado) — Universidade Federal de Ouro Preto, Ouro Preto, Agosto 2024.

ALMEIDA, A. L. B.; LIMA, J. C.; CARVALHO, M. A. M. Revisitando o algoritmo keep tools needed soonest: implementações seriais e paralelas. *Anais do Simpósio Brasileiro de Pesquisa Operacional*, 2023.

BEEZãO, A. C.; CORDEAU, J.-F.; LAPORTE, G.; YANASSE, H. H. Scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research*, v. 257, n. 3, p. 834–844, 2017. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221716306233>.

CALMELS, D. The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends. *International Journal of Production Research*, Taylor & Francis, v. 57, n. 15-16, p. 5005–5025, 2019. Disponível em: <https://doi.org/10.1080/00207543.2018.1505057>.

CHCOMPANY the B. R. *Metalworking Machinery Global Market Report 2024 – By Type*. 2024. Accessed: April 15, 2024. Disponível em: <https://www.thebusinessresearchcompany.com/report/metalworking-machinery-global-market-report>.

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

CRAMA, Y.; KOLEN, A. W. J.; OERLEMANS, A. G.; SPIEKSMA, F. C. R. *Minimizing the Number of Tool Switches on a Flexible Machine*. 1994. 33–54 p. Disponível em: <https://doi.org/10.1007/BF01324874>.

DANG, Q.-V.; HERPS, K.; MARTAGAN, T.; ADAN, I.; HEINRICH, J. Unsupervised parallel machines scheduling with tool switches. *Computers and Operations Research*, v. 160, p. 106361, 2023. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054823002253>.

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

FARUGHI, H.; DOLATABADIAA, M.; MORADI, V.; KARBASI, v.; MOSTAFAYI, S. Minimizing the number of tool switches in flexible manufacturing cells subject to tools reliability using genetic algorithm. *Journal of Industrial and Systems Engineering*, Iranian Institute of Industrial Engineering, v. 10, n. special issue on Quality Control and Reliability, p. 17–33, 2017. ISSN 1735-8272. Disponível em: <https://www.jise.ir/article_33655.html>.

FOGLIATTO, F. S.; da Silveira, G. J.; BORENSTEIN, D. The mass customization decade: An updated review of the literature. *International Journal of Production*

Economics, v. 138, n. 1, p. 14–25, 2012. ISSN 0925-5273. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925527312000989>.

GAN, Z. L.; MUSA, S. N.; YAP, H. J. A review of the high-mix, low-volume manufacturing industry. *Applied Sciences*, v. 13, n. 3, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/3/1687>.

GHRAYEB, N. P. O. A.; FINCH, P. R. A mathematical model and heuristic procedure to schedule printed circuit packs on sequencers. *International Journal of Production Research*, Taylor & Francis, v. 41, n. 16, p. 3849–3860, 2003. Disponível em: <https://doi.org/10.1080/0020754031000118071>.

HANSMANN, U. H. Parallel tempering algorithm for conformational studies of biological molecules. *Chemical Physics Letters*, Elsevier BV, v. 281, n. 1–3, p. 140–150, dez. 1997. ISSN 0009-2614. Disponível em: <http://dx.doi.org/10.1016/S0009-2614(97)01198-6>.

HIRVIKORPI, M.; SALONEN, K.; KNUUTILA, T.; NEVALAINEN, O. S. The general two-level storage management problem: A reconsideration of the ktns-rule. *European Journal of Operational Research*, v. 171, n. 1, p. 189–207, 2006. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221704005934>.

INDUSTRYARC. *Printed Circuit Board Assembly Market - Forecast(2024 - 2030)*. 2024. Accessed: April 15, 2024. Disponível em: <https://www.industryarc.com/Report/18291/printed-circuit-boards-pcb-assembly-market.html>.

KEUNG, K. W.; IP, W. H.; LEE, T. C. The solution of a multi-objective tool selection model using the ga approach. *The International Journal of Advanced Manufacturing Technology*, v. 18, n. 11, p. 771–777, 2001. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s001700170001>.

MATZLIACH, B.; TZUR, M. Storage management of items in two levels of availability. *European Journal of Operational Research*, v. 121, n. 2, p. 363–379, 2000. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221799000375>.

MECLER, J.; SUBRAMANIAN, A.; VIDAL, T. A simple and effective hybrid genetic search for the job sequencing and tool switching problem. *Computers and Operations Research*, v. 127, p. 105153, 11 2020.

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*, v. 21, n. 6, p. 1087–1092, 06 1953. ISSN 0021-9606. Disponível em: <https://doi.org/10.1063/1.1699114>.

MORGAN, J.; HALTON, M.; QIAO, Y.; BRESLIN, J. G. Industry 4.0 smart reconfigurable manufacturing machines. *Journal of Manufacturing Systems*, v. 59, p. 481–506, 2021. ISSN 0278-6125. Disponível em: <https://www.sciencedirect.com/science/article/pii/S027861252100056X>.

NOëL, M.; SODHI, M. S.; LAMOND, B. F. Tool planning for a lights-out machining system. *Journal of Manufacturing Systems*, v. 26, n. 3, p. 161–166, 2007. ISSN 0278-6125. Design, Planning, and Control for Reconfigurable Manufacturing Systems. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0278612508000344>.

PRIVAULT, C.; FINKE, G. Modelling a tool switching problem on a single nc-machine. *Journal of Intelligent Manufacturing*, v. 6, n. 2, p. 87–94, abr. 1995. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/BF00123680>.

PROFESSOR, J. F. B. A. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, Taylor & Francis, v. 20, n. 4, p. 382–391, 1988. Disponível em: <https://doi.org/10.1080/07408178808966195>.

RUPE, J.; KUO, W. Solutions to a modified tool loading problem for a single fmm. *International Journal of Production Research*, Taylor & Francis, v. 35, n. 8, p. 2253–2268, 1997. Disponível em: <https://doi.org/10.1080/002075497194831>.

SPLUNK. *Cyber-Physical Systems (CPS) Explained*. 2024. Accessed: April 22, 2024. Disponível em: <https://www.splunk.com/en_us/blog/learn/cyber-physical-systems.html>.

TANG, C. S.; DENARDO, E. V. Models arising from a flexible manufacturing machine, part i: Minimization of the number of tool switches. *Operations Research*, v. 36, n. 5, p. 767–777, 1988. Disponível em: <https://doi.org/10.1287/opre.36.5.767>.

TZUR, M.; ALTMAN, A. Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions*, v. 36, p. 95–110, 02 2004.

ÖZPEYNIRCI, S.; GöKGüR, B.; HNICH, B. Parallel machine scheduling with tool loading. *Applied Mathematical Modelling*, v. 40, n. 9, p. 5660–5671, 2016. ISSN 0307-904X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0307904X16000093>.