Revisitando o algoritmo *Keep Tools Needed Soonest*: implementações seriais e paralelas

André Luís Barroso Almeida
Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, s/n, CEP 35400-000 - Ouro Preto/MG
andre.almeida@ifmg.edu.br

Joubert de Castro Lima
Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, s/n, CEP 35400-000 - Ouro Preto/MG
joubert@ufop.edu.br

Marco Antonio Moreira de Carvalho
Universidade Federal de Ouro Preto
Campus Morro do Cruzeiro, s/n, CEP 35400-000 - Ouro Preto/MG
mamc@ufop.edu.br

RESUMO

A utilização de máquinas flexíveis de comando numérico permite uma produção mais eficiente e versátil. No entanto, a introdução dessas máquinas gerou dois problemas combinatórios: a definição da sequência de execução das tarefas e a definição das ferramentas carregadas nos *magazines* das máquinas. O presente trabalho revisita métodos e implementações desenvolvidas pela comunidade científica para solucionar o segundo problema e as compara com três novas implementações, uma delas sequencial e outras duas paralelas. Os resultados mostram que as novas implementações geram uma redução significativa de até 96,3% no tempo computacional.

PALAVRAS CHAVE. FMS, SSP, TP, KTNS, Computação paralela.

Tópicos: AD&GP – PO na Administração e Gestão da Produção, POI – PO na Indústria.

ABSTRACT

The use of numerical control flexible machines enable a more efficient and versatile production. However, the introduction of these machines has generated two combinatorial problems: the definition of the task execution sequence and the definition of the tools loaded into the machines' magazines. This paper revisits methods and implementations developed by the scientific community to solve the second problem and compares them with three new implementations, one sequential and two parallel. The results show that the new implementations achieve a significant reduction of up to 96.3% in computational time.

KEYWORDS. FMS, SSP, TP, KTNS, Parallel computing.

Tópicos: AD&GP – OR in Production Administration and Management in Industry, POI – OR in Industry.

1. Introdução

Um sistema de manufatura flexível (*flexible manufacturing system*, FMS) é caracterizado pela utilização de máquinas de comando numérico e um sistema automatizado de fluxo de matéria prima. Cada máquina é equipada com um compartimento, conhecido como *magazine*, de capacidade limitada, responsável por armazenar as ferramentas necessárias para a produção de pelo menos um determinado produto. Em geral, em um sistema produtivo moderno, o número de ferramentas necessárias para a produção de todos os produtos excede consideravelmente a capacidade do *magazine*, sendo inevitável a constante troca de ferramentas. Neste contexto, o processo de trocas de ferramentas consome mais tempo do que qualquer outra atividade de configuração operacional [Van Hop e Nagarur, 2004]. Consequentemente reduzir a quantidade de trocas de ferramentas é crucial para o aumento da produtividade de um FMS.

Devido a sua relevância no cenário produtivo mundial, diversos problemas de sequenciamento de tarefas e troca de ferramentas (ou *job sequencing and tool switching problem*, SSP) emergiram, sendo amplamente estudados pela comunidade científica. O SSP tem como objetivo minimizar o número de trocas de ferramentas e consiste em geral de dois subproblemas: i) determinar a sequência ótima de execução das tarefas (*job sequencing problem*); e ii) determinar o *plano de trocas de ferramentas*, i.e., determinar quais ferramentas devem ser trocadas antes de uma nova tarefa ser processada (*tooling problem*, TP). O primeiro subproblema pertence à classe *NP-difícil* [Crama et al., 1994], não se conhecendo algoritmo em tempo determinístico polinomial capaz de resolvê-lo, enquanto o segundo subproblema pertence a classe *P*, devido a existência de um algoritmo correto em tempo determinístico polinomial conhecido como *Keep Tools Needed Soonest* (KTNS), desenvolvido por Tang e Denardo [1988].

Formalmente, o SSP básico ou uniforme é um problema de escalonamento de tarefas constituído de um conjunto de tarefas $J = \{1, 2, 3, ..., n\}$, um conjunto de ferramentas $T = \{1, 2, 3, ..., m\}$ e de somente uma máquina de comando numérico capaz de armazenar, no máximo, C ferramentas de forma simultânea em seu *magazine*. Para cada tarefa $j \in J$ existe um subconjunto $T_j \in T$ contendo todas as ferramentas necessárias para a execução da tarefa, com $|T_j| \leq C$. O tempo de troca entre as ferramentas é uniforme, além de cada *slot* do *magazine* comportar cada uma das ferramentas do conjunto T individualmente. O objetivo final do SSP básico ou uniforme é determinar qual tarefa será processada em cada estágio de produção e o plano de trocas de ferramentas que minimize o número total final de tais trocas.

Nas últimas três décadas, as pesquisas se concentraram no desenvolvimento de métodos baseados em metaheurísticas para solucionar o SSP [Calmels, 2019]. Em comum, este métodos possuem a necessidade de calcular repetidamente a função de avaliação, representada pelo TP. Felizmente, o algoritmo KTNS resolve esse problema em tempo determinístico polinomial, limitado por $O(nm)$, garantindo a otimalidade ao se inserir uma ferramenta somente quando ela é requerida pela próxima tarefa e retirando, quando necessário, as ferramentas que serão utilizadas mais tardiamente.

O algoritmo KTNS já é conhecido há mais de 35 anos, contudo, pouco se discute sobre implementações que possam reduzir seu consumo computacional. No entanto, é sabido que o cálculo da função de avaliação é o componente de uma metaheurística que mais consome poder computacional, podendo representar até 90% do tempo total de uma metaheurística conforme relatado em Janiak et al. [2008]. Embora a maioria dos trabalhos relacionados ao SSP negligenciem esse aspecto, reduzir o tempo computacional de avaliação de soluções pode propiciar um maior investimento na exploração do espaço de busca, podendo levar a uma melhor solução. Neste contexto, duas perguntas emergem: i) Qual das implementações atuais possui melhor desempenho? ii) É possível desenvolver uma implementação robusta e ao mesmo tempo com um consumo computacional menor?

Para investigar as questões levantadas, foi conduzida uma revisão de literatura a fim de identificar as implementações existentes. A partir disso, experimentos computacionais reproduziram duas implementações dos trabalhos identificados na literatura e as compararam a três novas implementações propostas neste estudo, uma sequencial e duas paralelas. Como resultado, observou-se uma redução no tempo computacional superior a 70% para a maioria das instâncias da literatura consideradas grandes.

O restante desse trabalho é estruturado em quatro seções. Na Seção 2, há o embasamento teórico do TP. Nas Seções 3 e 4, as implementações encontradas na literatura e as desenvolvidas neste estudo são detalhadas e descritas, respectivamente. Em seguida, na Seção 5, os experimentos, comparando as implementações são apresentados. Por fim, na Seção 6, o trabalho é concluído e direcionamentos para pesquisas futuras são apresentados.

2. Revisão da literatura

O objetivo dessa revisão é identificar publicações que apresentaram soluções alternativas ao algoritmo KTNS, além de implementações com melhor desempenho. Foi definido o processo de busca que abrangeu artigos de revistas e conferências, livros e capítulos de livros, indexados no período de 1945 a 2023. A busca foi realizada utilizando as bases de dados científicas WOS[1] e *Google Scholar*[2]. As publicações foram selecionadas seguindo critérios de inclusão e exclusão, ambos baseados no formato, origem e aderência ao assunto da pesquisa. Por fim, 21 publicações elegíveis foram selecionadas para a construção da presente revisão de literatura.

Verificou-se que a maioria dos estudos foram desenvolvidos à partir da década de 90 e somente três métodos distintos foram desenvolvidos para solucionar o TP uniforme, sendo o mais famoso dentre eles, o próprio KTNS. Este método foi desenvolvido por Tang e Denardo [1988] que também provaram sua corretude. Posteriormente, Crama et al. [1994] apresentaram uma prova mais compacta da corretude do algoritmo, baseada em uma formulação de programação inteira, que reduz o TP ao problema de fluxo máximo em redes. Seguindo o mesmo princípio, Cohen e Burkhard [1995] também provaram a otimalidade do KTNS através de uma nova formulação.

Embora o KTNS apresente regras simples e de fácil compreensão, provar sua otimalidade não é uma tarefa trivial [Privault e Finke, 1995]. Se apoiando nesta justificativa, Privault e Finke [1995] desenvolveram um algoritmo alternativo para resolver o TP. Neste algoritmo, o TP é formulado como sendo um problema de fluxo de custo mínimo. Entretanto, devido à topologia da rede associada ao TP, essa formulação possui complexidade limitada por $O(C^3n^2)$, considerada excessivamente alta quando comparada à complexidade do KTNS, limitada por $O(mn)$.

Cherniavskii e Goldengorin [2022] propuseram recentemente um novo algoritmo para solução do TP, o qual promete melhorar o desempenho em relação ao KTNS para as maiores instâncias da literatura. Esse novo método, denominado de *Greedy Pipe Construction Algorithm* (GPCA) possui uma complexidade amortizada limitada por $O(Cn)$. A operação deste algoritmo pode ser descrita como a de preencher, a cada estágio de produção, os *slots* vazios do *magazine* de forma a minimizar o número total de trocas. No entanto, ao contrário do método proposto por Privault e Finke [1995], o GPCA é capaz de solucionar apenas o TP uniforme, embora possa ser adaptado para outras variações do problema. A Seção 3.2 apresenta uma descrição deste algoritmo.

Ao analisar variações do SSP, novos métodos foram encontrados para solução do TP, em sua maioria derivados do KTNS. Rupe e Kuo [1997] utilizaram uma adaptação heurística do KTNS para resolver o caso em que a quantidade de ferramentas necessárias para processar cada tarefa pode exceder a capacidade do *magazine*. Outra variação do TP foi abordada por Tzur e Altman [2004], em

[1] https://clarivate.com/webofsciencegroup/solutions/web-of-science/
[2] https://scholar.google.com

que uma ferramenta pode ocupar mais de um *slot* do *magazine*. Novamente, uma heurística baseada no KTNS foi desenvolvida e chamada de *Keep Smaller Tools Needed Soonest*. Adicionalmente, Raduly-Baka et al. [2005] e Crama et al. [2007] abordaram a mesma variação do SSP. Raduly-Baka et al. [2005] propuseram uma nova heurística com elementos similares aos do KTNS para solucionar o problema. Crama et al. [2007] provaram que este é um problema *NP-Difícil*, porém, quando é conhecida *a priori* a capacidade do *magazine*, esta variação do TP pode ser solucionada em tempo determinístico polinomial ao ser modelada como sendo um problema de caminho mais curto em um grafo direcionado. Van Hop [2005], por sua vez, reuniu ambas as características anteriores do TP em uma nova versão do SSP, a qual foi abordada por um algoritmo guloso.

Song e Shinn [2001] analisaram o problema em que as tarefas a serem processadas não são conhecidas *a priori*, sendo essa informação fornecida em tempo de execução de forma aleatória. Os autores propuseram uma heurística que foi comparada com o KTNS, sendo até 44,7% pior em relação a quantidade de trocas. Um ano depois, Song e Hwang [2002] abordaram o TP em um cenário em que se deseja minimizar a quantidade de movimentos de um transportador de ferramentas cuja capacidade é limitada. Para esse fim, o KTNS foi adaptado com o objetivo de permitir o adiantamento do carregamento de ferramentas e assim, ao ocupar mais espaços vazios no transportador, a quantidade de movimentos é reduzida. Para aproximar ainda mais o problema teórico do problema real, Hirvikorpi et al. [2006a] consideraram o tempo de vida de cada ferramenta, sendo necessário reformular as regras do KTNS.

Tanto o SSP quanto o TP podem ser encontrados no processo de fabricação de placas de circuitos impressos (PCB). Hirvikorpi et al. [2006b] abordaram o problema de confecção de PCBs com a existência de dois locais de armazenamento dos componentes eletrônicos, o que equivale a existência de dois *magazines* na máquina flexível. Para se adequar a esse cenário, os autores revisaram as regras do KTNS. Outra aplicação prática do TP ocorre no chamado problema de gerenciamento de memória. Neste contexto, é possível encontrar adaptações do algoritmo KTNS como nos trabalhos de Salem et al. [2016a] e Salem et al. [2016b].

Mais recentemente, Rifai et al. [2022] consideraram que as trocas de ferramentas possuem tempo não uniforme, pois, em aplicações industriais reais, o tempo de troca pode ser influenciado pelos tipos das ferramenta que serão trocadas. Desta forma, a determinação do plano de trocas inclui a especificação explícita dos pares de ferramentas a serem trocadas e o tempo associado a tais trocas. Apesar da existência do algoritmo polinomial e exato desenvolvido por Privault e Finke [1995] para esse problema, Rifai et al. [2022] propuseram uma metaheurística híbrida KTNS+*Simulated Annealing* para a solução simultânea do TP e do problema de atribuição linear dos pares de ferramentas.

As informações extraídas nesta revisão indicam que após o trabalho de Tang e Denardo [1988], houve pouca inovação algorítmica relacionada ao KTNS e ao TP. Ademais, as características da implementação do KTNS não são reveladas na maioria das publicações, pois o TP é considerado como sendo um problema secundário. Somente a publicação de Cherniavskii e Goldengorin [2022] analisou esse quesito em detalhes. Resta evidenciada a carência relacionada ao estudos específicos sobre o TP, mesmo este representando boa parte do tempo computacional para resolver o SSP.

3. Implementações originais

Esta seção é dedicada a descrever brevemente duas implementações originais de soluções para o TP encontradas na literatura. Na Seção 3.1, a implementação do KTNS clássico é descrita, sendo baseada no recente código fonte fornecido por Mecler et al. [2021][3]. Em seguida, na Seção 3.2, a implementação do GPCA apresentada em Cherniavskii e Goldengorin [2022] é descrita.

[3] https://github.com/jordanamecler/HGS-SSP

3.1. KTNS padrão

O KTNS introduzido por Tang e Denardo [1988] possui duas regras simples. Na primeira delas, em qualquer estágio de produção, não se permite a inserção de nenhuma ferramenta no *magazine*, a menos que ela seja exigida pela próxima tarefa a ser processada. Na segunda regra, quando é necessário inserir uma ferramenta, retira-se a ferramenta que será necessária mais tardiamente no sequenciamento das tarefas, a fim de liberar um *slot* para a nova ferramenta. Para realizar tal operação, Tang e Denardo [1988] definiram o conjunto $T(i, n)$, contendo todos os estágios de produção em que a ferramenta i é necessária após ou durante estágio de produção n. Além desse conjunto, foi definida a função $L(i, n)$ que retorna o primeiro estágio de produção após ou durante estágio de produção n em que a ferramenta i é necessária.

Em uma implementação considerada o estado da arte para o SSP, Mecler et al. [2021] codificaram o conjunto $T(i, n)$ como uma matriz de ocorrências $m \times n$ em que cada linha representa uma ferramenta (m) e cada coluna representa um estágio da produção (n). Cada célula da matriz representa a função $L(i, n)$. Em seguida, o *magazine* do primeiro estágio é preenchido com todas as ferramentas necessárias para processar a tarefa correspondente. Como nenhuma tarefa possui mais ferramentas que a capacidade do *magazine*, uma estrutura de repetição é utilizada para alocar as próximas ferramentas que serão utilizadas nos *slots* vazios do *magazine*.

Após completar o *magazine* do primeiro estágio de produção, esta implementação percorre todos os outros n estágios de produção em uma estrutura de repetição. Para cada estágio de produção, são alocadas no *magazine* as ferramentas necessárias para processar a tarefa correspondente. Assim, a quantidade de ferramentas no *magazine* pode exceder a capacidade máxima, sendo necessária uma correção, que consiste em retirar do *magazine* as ferramentas mais tardiamente necessárias até que o *magazine* atinja sua capacidade máxima. Para cada ferramenta retirada, uma troca é contabilizada. Ao fim do algoritmo, a quantidade total de trocas é retornada.

3.2. *Greedy Pipe Construction Algorithm*

O GPCA está intimamente relacionado com uma nova interpretação do TP. Nesta interpretação, o TP busca preencher os espaços vazios do *magazine* ao longo dos estágios de produção de forma a minimizar a quantidade de trocas de ferramentas entre dois estágios consecutivos. Entretanto, o GPCA, por si só, somente é capaz de obter o valor ótimo da solução. Para se obter o plano de trocas em detalhes, o GPCA deve ser executado em conjunto com o *ToFullMag*, um segundo algoritmo criado pelos autores cuja complexidade é limitada por $O(Cn)$.

Diferente do KTNS, que monta o *magazine* a cada estágio de produção, o GPCA identifica o que se chamou de *pipe*. Um *pipe* $\pi^t_{s,e}$, corresponde a uma sequência em que a ferramenta t é utilizada nos estágios de produção s e e, porém, não é utilizada nos estágios de produção intermediários $(s + 1, ..., e - 1)$. A conexão entre os *magazines* dos estágios de produção s e e caracteriza um *pipe*. Entretanto, é necessário observar se há *slots* livres nos *magazines* correspondentes aos estágios de produção intermediários para que a ferramenta t seja inserida. Quanto maior a quantidade de *pipes*, menor a quantidade de trocas de ferramentas. Um exemplo de construção de *pipes* pode ser observado na Figura 1, em que cada quadrado com fundo branco representa um *slot* no *magazine* ocupado por uma ferramenta obrigatória, os quadrados pontilhados representam *slots* vazios no *magazine* e as linhas entre os estágios de produção representam os *pipes*.

Para identificar todos os *pipes*, o GPCA possui uma estrutura de repetição principal que percorre todos os estágios de produção. Para cada um deles, uma nova estrutura de repetição é executada, a qual analisa todas as ferramentas necessárias para processar a tarefa atual. Se a ferramenta em análise foi utilizada em um estágio de produção anterior ao atual e ainda há *slots* livres no *magazine* nos estágios de produção intermediários, é contabilizado um *pipe* e a ferramenta

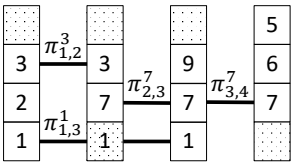

Figura 1: Exemplo de *pipes*.

é adicionada aos *slots* vazios nos estágios de produção intermediários. Caso se complete o *magazine* em algum dos estágios de produção, ele é registrado para futuras análises. Ao final, é contabilizada a quantidade de trocas como $\sum_{i=1}^{n} |T_i| - C - |pipes|$, em que n representa a quantidade de tarefas, T_i o conjunto de ferramentas exigidas para processar a tarefa i e C a capacidade do *magazine*.

O GPCA possui como grande benefício um tempo de execução atrativo. Entretanto, como se pode observar na Figura 1, esse algoritmo não determina o plano de troca de ferramentas completo, ou seja, a solução do TP. Para contornar esse problema, os autores propuseram o algoritmo *ToFullMag*. Esse algoritmo possui uma estrutura de repetição que percorre todos os pares adjacentes, tanto da esquerda para direita quanto da direita para esquerda, de todos os estágios de produção $((i, j) = (1, 2), ..., (n-1, n), (n, n-1), ...(2, 1))$. Para cada par, uma nova estrutura de repetição percorre todas as ferramentas necessárias para processar a tarefa do estágio i. Se a ferramenta atual não for utilizada no estágio j e o *magazine* ainda possuir *slots* vazios, a ferramenta é adicionada ao *magazine* deste estágio. Esse procedimento é capaz de completar os *slots* vazios do *magazine* em todos os estágios de produção, retornando a solução completa para o TP.

4. Novas implementações

Esta seção é dedicada a descrever brevemente as novas implementações do KTNS e do GPCA. Na Seção 4.1 é descrita uma implementação aprimorada do KTNS sequencial. Nas próximas duas seções, relatam-se implementações paralelas do algoritmo KTNS e do GPCA. O código fonte das implementações descritas nesta seção e na Seção 3 estão disponíveis para *download* [4].

4.1. KTNS sequencial

Após uma análise cuidadosa do KTNS, foi observado que o gargalo do custo computacional se encontra na construção do conjunto $T(i, n)$ e na etapa de ordenação responsável por determinar qual ferramenta é mais tardiamente necessária. Assim, com o objetivo de reduzir tais custos computacionais, o conjunto $T(i, n)$ e a ordenação passaram a serem realizadas simultaneamente com a construção do *magazine*, resultando em uma redução significativa no tempo computacional exigido. Essa modificação da implementação do KTNS o tornou novamente competitivo quando comparado com o GPCA.

A implementação proposta possui uma estrutura de repetição principal que percorre cada tarefa na sequência previamente definida. Uma segunda estrutura de repetição percorre todas as ferramentas necessárias para processar a tarefa atual, adicionando-as ao *magazine* do estágio de produção correspondente. Se a ferramenta já está no *magazine* anterior, nenhuma troca é contabilizada. Caso contrário, uma troca é registrada. Após esse processo, caso haja *slots* vazios no *magazine*, o algoritmo analisa e insere as ferramentas necessárias para as tarefas subsequentes na ordem em que elas aparecem. As ferramentas adicionadas obrigatoriamente devem estar presentes no *magazine* do estágio de produção imediatamente anterior ao atual. Dessa forma, as ferramentas necessárias mais próximas são sempre inseridas primeiro, sem a necessidade de uma ordenação.

[4] `https://github.com/ALBA-CODES/KTNS/`

Após o término desta iteração, o *magazine* construído se torna o novo *magazine* anterior e a estrutura de repetição principal segue para o próximo estágio. O algoritmo termina após percorrer todas os estágios da produção. A Figura 2 ilustra a execução da implementação proposta para um problema com quatro tarefas, nove ferramentas e a capacidade do *magazine* igual a quatro. Na referida figura, em (a) tem-se as ferramentas exigidas por cada tarefa sequenciada e em (b)-(e), tem-se os magazines referentes a cada estágio da produção em que cada célula branca representa uma ferramenta obrigatória no estágio e células hachuradas indicam *slots* vazios inicialmente.

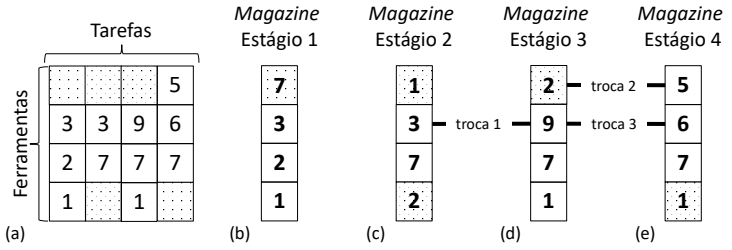

Figura 2: Exemplo de execução do KTNS proposto.

Inicialmente, são fornecidas a ordem das tarefas e as ferramentas necessárias para processá-las. No primeiro estágio, as ferramentas 1, 2 e 3 são adicionadas ao *magazine* obrigatoriamente para execução da primeira tarefa. Em seguida, para completar os *slots* vazios deste *magazine*, as ferramentas da próxima tarefa são analisadas. Como a ferramenta 3 já se encontra no *magazine*, somente a ferramenta 7 é adicionada.

No segundo estágio, são adicionadas as ferramentas 3 e 7 ao *magazine*. Para completar os *slots* vazios, foi adicionada a ferramenta 1 que é necessária para processar a tarefa do estágio subsequente e já se encontra no magazine do estágio imediatamente anterior (estágio 1). Como ainda há mais um *slot* disponível no *magazine*, mesmo sem ser necessária às tarefas subsequentes, a ferramenta 2 é adicionada, pois não há ferramentas nas tarefas dos estágios subsequentes que já se encontrem no *magazine* do estágio imediatamente anterior. No terceiro estágio, são adicionadas ao *magazine* as ferramentas 1, 7 e 9. Como as ferramentas 2 e 3 não são necessárias nas próximas tarefas e com um *slot* vazio no *magazine*, a ferramenta 2 é adicionada. Como resultado da diferença entre o *magazine* do estágio atual e do estágio anterior, uma troca é contabilizada. No último estágio, as ferramentas 5, 6 e 7 são inseridas no *magazine* e a ferramenta 1 é utilizada para completá-lo, resultando em mais duas trocas.

4.2. KTNS paralelo

A partir da nova implementação do KTNS sequencial, propõe-se sua execução em paralelo. Trata-se de um algoritmo com alto grau de dependabilidade entre as tarefas, portanto, o nível de paralelização é limitado. Propõe-se dividir a execução do KTNS em duas partes, preenchendo simultaneamente os *magazines* de cada estágio de produção da esquerda para direita e da direita para a esquerda, com um ponto de união central. A Figura 3 ilustra este processo para um problema com cinco tarefas, nove ferramentas e capacidade do *magazine* igual a quatro.

A implementação paralela consiste de quatro etapas. Na primeira, o algoritmo determina qual estágio central, que corresponde à metade da solução do problema ($\frac{n}{2}$). Na segunda etapa, duas instâncias do algoritmo sequencial são executadas em paralelo, uma partindo do primeiro estágio até o estágio central e outra partindo do último estágio até o estágio central. Na terceira etapa, os *slots* vazios do *magazine* referente ao estágio central são preenchidos com as ferramentas que se encontram tanto no *magazine* à esquerda quanto no *magazine* à direita do estágio central. Caso não se

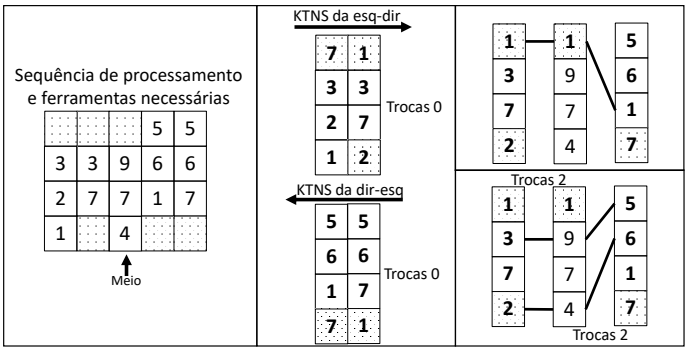

Figura 3: Exemplo de execução do KTNS proposto paralelo.

tenha completado este *magazine*, ferramentas são adiantadas do *magazine* à direita. Na última etapa, é contabilizada a quantidade de trocas entre os *magazines* dos estágios de produção, $\frac{n}{2}-1$, $\frac{n}{2}$ e $\frac{n}{2}+1$. A quantidade de trocas total corresponde a soma de todas as trocas. Ao dividir o algoritmo em duas etapas paralelas, a propriedade de otimalidade é perdida devido a etapa de união dos subproblemas, podendo gerar trocas desnecessárias. Entretanto, sua taxa de acerto supera os 90% na maioria das instâncias, conforme apresentado na Seção 5.

4.3. GPCA paralelo

No intuito de melhorar o desempenho do algoritmo GPCA, propõe-se uma implementação paralela baseada na GPCA sequencial. De forma similar ao algoritmo paralelo anterior, a proposta é executar o GPCA em duas etapas simultâneas, contabilizando os *pipes* em paralelo. Após a execução paralela, o GPCA é novamente executado, partindo da posição mais à esquerda em que o *magazine* foi completado até a posição mais à direita em que o *magazine* foi completado, de forma a unir os subproblemas. Na Figura 4 é apresentado um exemplo com cinco tarefas, nove ferramentas e capacidade do *magazine* igual a quatro.

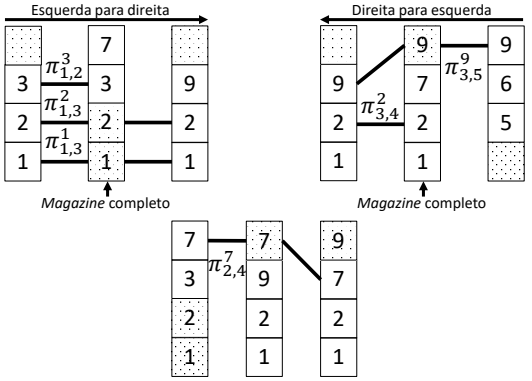

Figura 4: Exemplo do GPCA paralelo.

O algoritmo é executado em paralelo, gerando, da esquerda para a direita os *pipes* $\pi^3_{1,2}$, $\pi^2_{1,3}$ e $\pi^1_{1,3}$, e, da direita para a esquerda, os *pipes* $\pi^2_{3,4}$ e $\pi^9_{3,5}$. Além dos *pipes*, o algoritmo identifica o estágio mais à esquerda (estágio dois) e o mais à direita (estágio quatro) em que o *magazine* se encontra completo. Em seguida, o GPCA é executado novamente, partindo do estágio dois até o estágio quatro, identificando mais um *pipe* ($\pi^7_{2,4}$). Ao final da execução do GPCA paralelo, a quantidade total de trocas é dada por $\sum_{i=1}^{n} |T_i| - C - |pipes|$.

5. Experimentos

Um conjunto de experimentos computacionais foi realizado com o objetivo de avaliar o tempo computacional exigido pelas implementações descritas para avaliação de soluções para instâncias *benchmark*, além de comparar as implementações entre si de forma justa. O ambiente computacional consistiu em um computador com um processador M1 8-core e 8GB de memória RAM, sob o sistema operacional macOS ventura 13.3.1. Os algoritmos foram implementados em C++ e compilados pelo compilador Apple clang 14.0.3 utilizando as opções -O3 e -lpthread.

Foram consideradas as instâncias propostas por Catanzaro et al. [2015] e Mecler et al. [2021], respectivamente, um *benchmark* clássico e outro considerado como o *benchmark* de maiores dimensões na literatura sobre o SSP uniforme. Para cada instância, 2×10^6 soluções únicas foram geradas por meio de permutações aleatórias, em um processo repetido dez vezes de forma independente.

Os tempos gerados pelas execuções das implementações sequenciais foram sumarizados na Tabela 1 e das implementações paralelas na Tabela 2. Ambas tabelas apresentam o número de tarefas (n), o número de ferramentas (m), a capacidade do *magazine* (c) e a quantidade de instâncias por grupo (#). Para cada implementação, são apresentados a média do tempo computacional de dez execuções em milissegundos (μ) e a distância percentual (ou *gap*) entre a média de tempo do KTNS sequencial disponibilizado por Mecler et al. [2021] e as médias dos tempos de todas as outras implementações. O *gap* foi calculado como $gap = \frac{(ALT-KTNS)}{KTNS} \times 100$, em que KTNS representa os tempos do *KTNS* de referência e *ALT* representa cada uma das implementações alternativas. Os melhores resultados, dentre as implementações alternativas, são destacados em negrito.

Tabela 1: Resultados dos experimentos dos algoritmos sequenciais.

					KTNS [Mecler et al., 2021]	KTNS aprimorado		GPCA		ToFullMag	
	n	m	c	#	μ (ms)	μ (ms)	gap (%)	μ (ms)	gap (%)	μ (ms)	gap (%)
Catanzaro et al. [2015]	10	10	4	10	962,11	840,35	**-12,66**	1189,53	23,64	2162,36	124,75
	10	10	5	10	895,54	1044,47	**16,63**	1419,1	58,46	2571,27	187,12
	10	10	6	10	832,34	1205,55	**44,84**	1502,54	80,52	2740,56	229,26
	10	10	7	10	776,26	1311,29	**68,92**	1599,57	106,06	3017,1	288,67
	15	20	6	10	2910,18	1394,08	**-52,10**	1733,08	-40,45	3665,04	25,94
	15	20	8	10	2834,62	1886,76	**-33,44**	2135,08	-24,68	4585,43	61,77
	15	20	10	10	2617,19	2298,27	**-12,19**	2380,97	-9,03	5482,25	109,47
	15	20	12	10	2361,85	2607,21	10,39	2566,71	**8,67**	6047,89	156,07
	30	40	15	10	16598,22	4045,38	**-75,63**	4439,92	-73,25	14749,08	-11,14
	30	40	17	10	15353,37	4793,61	**-68,78**	4938,16	-67,84	16276,26	6,01
	30	40	20	10	13530,57	5771,12	-57,35	5349,75	**-60,46**	18041,04	33,34
	30	40	25	10	10917,17	7101,42	-34,95	6105,21	**-44,08**	21312,85	95,22
	40	60	20	10	42049,16	6401,7	-84,78	6036,09	**-85,65**	24655,05	-41,37
	40	60	22	10	39558,94	7321,98	-81,49	6591,3	**-83,34**	26520,95	-32,96
	40	60	25	10	35990,96	8638,63	-76,00	7169,44	**-80,08**	29186,33	-18,91
	40	60	30	10	30654,9	10645,68	-65,27	7950,61	**-74,06**	33265,57	8,52
Mecler et al. [2021]	50	75	25	5	85943,84	9626,46	-88,80	7977,52	**-90,72**	36441,5	-57,60
	50	75	30	5	68825,5	12544,1	-81,77	9114,66	**-86,76**	41765,7	-39,32
	50	75	35	5	58216,66	14855,82	-74,48	10269,02	**-82,36**	47752,58	-17,97
	50	75	40	5	49764,42	17070,14	-65,70	11469,12	**-76,95**	53460,36	7,43
	60	90	35	5	185925,84	14663,5	-92,11	11043,6	**-94,06**	58654,86	-68,45
	60	90	40	5	153737,5	18264,62	-88,12	12699,16	**-91,74**	65518,78	-57,38
	60	90	45	5	122621,44	21563,42	-82,41	14329,54	**-88,31**	72193,54	-41,12
	60	90	50	5	96378,76	24797,32	-74,27	15786,98	**-83,62**	79279,28	-17,74
	70	105	40	5	310697,12	19582,7	-93,70	14465,96	**-95,34**	77960,74	-74,91
	70	105	45	5	274235,16	23380,68	-91,47	16247,34	**-94,08**	85592,34	-68,79
	70	105	50	5	239108,42	27324,44	-88,57	18101,3	**-92,43**	93172,66	-61,03
	70	105	55	5	200090,54	31769,02	-84,12	19674,64	**-90,17**	101103,64	-49,47

Na primeira parte da Tabela 1, o KTNS aprimorado superou o desempenho do KTNS de referência em 12 dos 16 grupos de instâncias. Particularmente, a nova implementação do KTNS teve pior desempenho em instâncias de menores dimensões. A implementação do GPCA e a implementação aprimorada do KTNS apresentaram desempenho equilibrado, cada uma se sobressaindo em 7 e 9 grupos de instâncias respectivamente. Nota-se que o GPCA se sobressai especificamente nas maiores instâncias. Na segunda parte da referida tabela, a implementação do GPCA obteve os melhores tempos para todas as instâncias. De maneira geral, tanto o KTNS aprimorado quanto o GPCA apresentaram uma redução significativa em relação ao tempo computacional de referência, chegando a reduzi-lo em 93,70% e 95,34% respectivamente. É importante ressaltar, entretanto, que caso seja necessária a determinação do plano de trocas de ferramentas completo, o algoritmo *ToFullMag* deve ser executado de maneira complementar ao GPCA. Neste caso específico, o tempo do algoritmo *ToFullMag* supera o tempo de execução da implementação aprimorada do KTNS.

A Tabela 2 sumariza os dados das implementações paralelas, seguindo o mesmo padrão da tabela anterior. Adicionalmente, a coluna α indica a acurácia da avaliação do KTNS paralelo, que nesta versão perde sua garantia de corretude. As implementações paralelas obtiveram desempenhos melhores que o KTNS sequencial para a maioria dos conjuntos de menores instâncias, embora em alguns grupos o *overhead* gerado pelo paralelismo tenha se tornado um aspecto negativo e levado a um pior desempenho. A redução do tempo computacional em relação aos valores de referência foi superior a 78%, chegando a até 96,3%. Comparando-se as versões paralelas com as suas versões sequenciais, somente no primeiro e menor dos grupos de instâncias o tempo computacional foi superior.

Tabela 2: Resultados dos experimentos dos algoritmos paralelos.

	n	m	c	#	KTNS [Mecler et al., 2021]	KTNS paralelo			GPCA paralelo	
					μ (ms)	μ (ms)	gap (%)	α (%)	μ (ms)	gap (%)
Catanzaro et al. [2015]	10	10	4	10	962,11	887,47	**-7,76**	99,84	1648,56	71,35
	10	10	5	10	895,54	1006,98	**12,44**	99,13	1977,51	120,82
	10	10	6	10	832,34	1076,73	**29,36**	96,70	2225,83	167,42
	10	10	7	10	776,26	1112,1	**43,26**	93,29	2415,62	211,19
	15	20	6	10	2910,18	1369,85	**-52,93**	99,66	2317,67	-20,36
	15	20	8	10	2834,62	1672,53	**-41,00**	97,59	2795,97	-1,36
	15	20	10	10	2617,19	1834,41	**-29,91**	90,97	3161,32	20,79
	15	20	12	10	2361,85	1930,99	**-18,24**	76,95	3441,55	45,71
	30	40	15	10	16598,22	3069,88	**-81,50**	99,14	4078,91	-75,43
	30	40	17	10	15353,37	3551,99	**-76,87**	97,96	4470,81	-70,88
	30	40	20	10	13530,57	4086,19	**-69,80**	94,33	4821,61	-64,37
	30	40	25	10	10917,17	4579,86	**-58,05**	84,35	5554,66	-49,12
	40	60	20	10	42049,16	4565,24	**-89,14**	99,14	5199,14	-87,64
	40	60	22	10	39558,94	5259,3	**-86,71**	98,24	5760,04	-85,44
	40	60	25	10	35990,96	6162,66	**-82,88**	95,71	6419,2	-82,16
	40	60	30	10	30654,9	7233,94	**-76,40**	87,94	7298,65	-76,19
Mecler et al. [2021]	50	75	25	5	85943,84	9626,46	-88,80	98,81	6636,88	**-92,28**
	50	75	30	5	68825,5	8180,48	-88,11	95,56	7963,94	**-88,43**
	50	75	35	5	58216,66	9568,32	-83,56	88,83	9034,36	**-84,48**
	50	75	40	5	49764,42	10558	-78,78	79,85	10188,12	**-79,53**
	60	90	35	5	185925,84	8917,7	**-95,20**	98,61	8970,86	-95,18
	60	90	40	5	153737,5	11426,76	-92,57	96,26	10571,72	**-93,12**
	60	90	45	5	122621,44	13676,48	-88,85	91,87	11969,6	**-90,24**
	60	90	50	5	96378,76	15515,14	-83,90	85,35	13269,44	**-86,23**
	70	105	40	5	310697,12	11821,48	-96,20	98,89	11501,86	**-96,30**
	70	105	45	5	274235,16	14418,82	-94,74	97,06	13184,88	**-95,19**
	70	105	50	5	239108,42	16873,84	-92,94	93,43	14575,98	**-93,90**
	70	105	55	5	200090,54	18986,5	-90,51	87,85	15701,1	**-92,15**

Comparando-se as implementações paralelas entre si, é possível observar desempenhos similares. Para as menores instâncias [Catanzaro et al., 2015], o KTNS aprimorado paralelo leva ligeira vantagem, ao passo em que para as maiores instâncias [Mecler et al., 2021], o GPCA paralelo obtém uma diferença de no máximo 3,48% entre os tempos computacionais. na maior parte das instâncias, essa diferença é menor que 1%. É importante ressaltar que a etapa de união dos subproblemas na implementação do KTNS aprimorado paralelo ocasiona eventuais erros no valor de avaliação. Este método apresentou uma acurácia média entre 76% e 98,89%, ao passo que a implementação do GPCA paralelo mantém sua corretude.

6. Conclusão

Abordou-se neste estudo o *Tooling Problem* (TP), um subproblema recorrente do *job sequencing and tool switching problem* (SSP). Foram propostas uma implementação sequencial e duas implementações paralelas de algoritmos para solução do TP. As implementações propostas foram comparadas com a implementação considerada o estado da arte, obtendo reduções significativas dos tempos computacionais, de até 96,3%. Esta redução do tempo computacional possui ampla aplicação e beneficia os métodos para solução do SSP e suas diversas variantes, permitindo que o tempo computacional seja reduzido, ou que possa ser investido em uma maior exploração do espaço de busca, na expectativa de geração de melhores soluções.

7. Agradecimentos

O presente estudo foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [Código de Financiamento 408341/2018-1]; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) [Código de Financiamento 001]; Universidade Federal de Ouro Preto; e Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Ouro Preto.

Referências

Calmels, D. (2019). The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends. *International Journal of Production Research*, 57(15-16):5005–5025.

Catanzaro, D., Gouveia, L., e Labbé, M. (2015). Improved integer linear programming formulations for the job sequencing and tool switching problem. *European journal of operational research*, 244(3):766–777.

Cherniavskii, M. e Goldengorin, B. (2022). An almost linear time complexity algorithm for the tool loading problem. *arXiv preprint arXiv:2207.02004*.

Cohen, A. e Burkhard, W. A. (1995). A proof of the optimality of the min paging algorithm using linear programming duality. *Operations research letters*, 18(1):7–13.

Crama, Y., Moonen, L. S., Spieksma, F. C., e Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, 182(2):952–957.

Crama, Y., Oerlemans, A. G., e Spieksma, F. C. (1994). Minimizing the number of tool switches on a flexible machine. In *Production Planning in Automated Manufacturing*, p. 165–195. Springer.

Hirvikorpi, M., Nevalainen, O., e Knuutila, T. (2006a). Job ordering and management of wearing tools. *Engineering optimization*, 38(2):227–244.

Hirvikorpi, M., Salonen, K., Knuutila, T., e Nevalainen, O. S. (2006b). The general two-level storage management problem: A reconsideration of the ktns-rule. *European journal of operational research*, 171(1):189–207.

Janiak, A., Janiak, W. A., e Lichtenstein, M. (2008). Tabu search on gpu. *J. Univers. Comput. Sci.*, 14(14):2416–2426.

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

Privault, C. e Finke, G. (1995). Modelling a tool switching problem on a single nc-machine. *Journal of Intelligent Manufacturing*, 6:87–94.

Raduly-Baka, C., Knuutila, T., e Nevalainen, O. (2005). *Minimising the number of tool switches with tools of different sizes*. TUCS technical report. Turku Centre for Computer Science, Turku.

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

Rupe, J. e Kuo, W. (1997). Solutions to a modified tool loading problem for a single fmm. *International Journal of Production Research*, 35(8):2253–2268.

Salem, K. H., Kieffer, Y., e Mancini, S. (2016a). Efficient algorithms for memory management in embedded vision systems. In *2016 11th IEEE Symposium on Industrial Embedded Systems (SIES)*, p. 1–6. IEEE.

Salem, K. H., Kieffer, Y., e Mancini, S. (2016b). Memory management in embedded vision systems: Optimization problems and solution methods. In *2016 conference on design and architectures for signal and image processing (DASIP)*, p. 200–207. IEEE.

Song, C.-Y. e Hwang, H. (2002). Optimal tooling policy for a tool switching problem of a flexible machine with automatic tool transporter. *International Journal of Production Research*, 40(4): 873–883.

Song, C.-Y. e Shinn, S.-W. (2001). The tool switching problem on a flexible machine in a dynamic environment. In *Proceedings of the Safety Management and Science Conference*, p. 189–193. Korea Safety Management & Science.

Tang, C. S. e Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part i: minimization of the number of tool switches. *Operations research*, 36(5):767–777.

Tzur, M. e Altman, A. (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions*, 36(2):95–110.

Van Hop, N. (2005). The tool-switching problem with magazine capacity and tool size constraints. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(5): 617–628.

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