LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

REVENIMENTO PARALELO APLICADO AO SEQUENCIAMENTO EM MÁQUINAS FLEXÍVEIS PARALELAS COM RECURSOS COMPARTILHADOS

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

Este artigo relata uma aplicação da metaheurística revenimento paralelo (ou *parallel tempering*, PT). Aborda-se o problema de sequenciamento de tarefas em máquinas paralelas idênticas com restrições de ferramentas e recursos compartilhados, um problema prático encontrado na indústria microeletrônica. É proposta a aplicação de uma implementação paralela do PT. O estudo comparativo entre o PT e o algoritmo estado da arte revela que o PT determinou novas melhores soluções para aproximadamente 82% das instâncias avaliadas, com margem de até 48% de melhoria.

PALAVRAS CHAVE. Sequenciamento. Manufatura Flexível. Revenimento Paralelo.

Tópicos (POI, MH, ADG&GP)

ABSTRACT

This paper reports an application of the parallel tempering metaheuristic (or *parallel tempering*, PT). We address the job sequencing on identical parallel machines with tooling constraints and shared resources problem, a practical problem encountered in the microelectronics industry. It is proposed to apply a parallel implementation of the PT. The comparative study between PT and the state-of-the-art algorithm reveals that PT determined new best solutions for approximately 82% of the evaluated instances, with a margin of up to 48% improvement.

KEYWORDS. Sequencing. Flexible Manufacturing. Parallel Tempering.

Paper topics (POI, MH, ADG&GP)

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

1. Introdução

Este estudo é um relato de aplicação da metaheurística revenimento paralelo a problemas de sequenciamento da produção em sistemas de manufatura flexível (SMF). Este tipo de sistema produtivo é caracterizado pela utilização de máquinas de comando numérico configuráveis e um sistema automatizado de fluxo de matéria prima, sendo comumente encontrado na indústria microeletrônica, especificamente no processo de fotolitografia, e na produção de objetos plásticos moldados. O problema de sequenciamento de máquinas paralelas com limitações de recursos (*resource constrained parallel machine scheduling*, RCPMS) é um problema prático recorrente neste tipo de sistema produtivo.

O RCPMS surge em um SMF que contém um conjunto de tarefas a serem processadas $J = \{1, 2, ..., n\}$ em que cada tarefa j $(j \in J)$ necessita de uma ferramenta f_j pertencente ao conjunto de ferramentas únicas e compartilhadas $F = \{1, 2, ..., l\}$. Essas tarefas, devem ser processadas em uma das máquinas do conjunto de máquinas idênticas paralelas $M = \{1, 2, ..., m\}$, que possuem um *magazine* que comporta somente uma ferramenta. Durante o processo produtivo, diversas tarefas são produzidas em sequência, sendo necessário substituir as ferramentas que se encontram no *magazine*. Além disso, no RCPMS, cada uma das tarefas possui um tempo de processamento τ_j $(j \in J)$ e um valor constante $\bar{p}$ referente a troca de cada uma das ferramentas, considerada uma operação de *setup*. Por fim, não há interdependência entre as tarefas, podendo ser processada em qualquer máquina em qualquer ordem.

O objetivo consiste em determinar a alocação e o sequenciamento das tarefas nas m máquinas de forma a minimizar o *makespan* resultante [Soares e Carvalho, 2022], i.e., o tempo de término de processamento da última máquina no sistema. Este é um problema pertencente à classe *NP-difícil*, não se conhecendo algoritmo em tempo determinístico polinomial capaz de resolvê-lo. A Figura 1 apresenta um exemplo de instância do RCPMS e duas diferentes soluções.

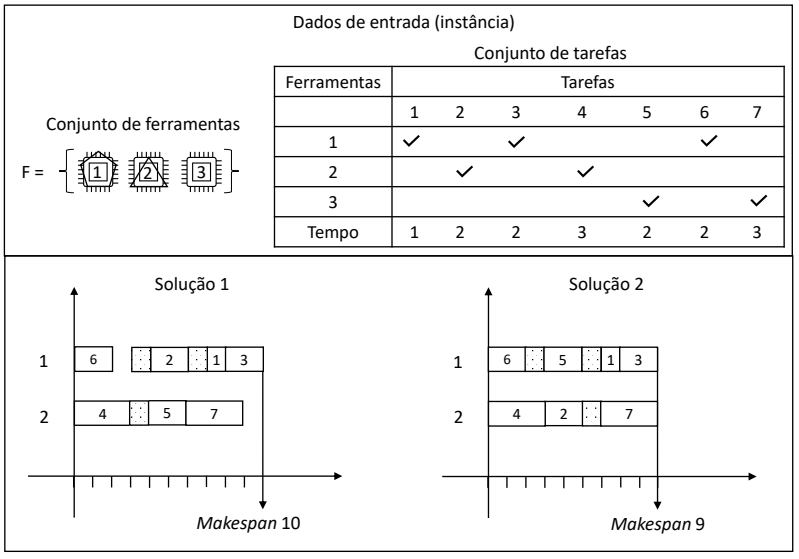


Figura 1: Exemplo de uma instância do RCPMS.

Na Figura 1, no primeiro retângulo podem ser observados os dados de entrada de uma instância do RCPMS com sete tarefas, três ferramentas e duas máquinas. No segundo retângulo são apresentadas duas soluções, em que, na primeira a máquina 1 processa a tarefa 6 e a máquina 2 processa a tarefa 4, ambas representadas por retângulos brancos. Posteriormente, para processar

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

a tarefa 2, a máquina 1 ficou ociosa por uma unidade de tempo pois a ferramenta 2, necessária, ainda estava sendo utilizada para processar tarefa 4 na máquina 2. A seguir, foi gasta uma unidade de tempo relacionado à troca de ferramentas em ambas as máquinas, representado por retângulos hachurados. Na máquina 1, após a troca da ferramenta, foi adicionada a tarefa 2, uma unidade de tempo relacionado a troca da ferramenta 2 pela ferramenta 1 e as tarefas 1 e 3, respectivamente. Na máquina 2, após a troca da ferramenta 2 pela ferramenta 3, foram adicionadas as tarefas 5 e 7. Ao final, o *makespan* foi de 10 unidades de tempo.

Na segunda solução, apresentada na Figura 1, foram atribuídas as tarefas 6, 5, 1 e 3 à máquina 1 e à máquina 2 foram atribuídas as tarefas 4, 2 e 7. A interpretação se dá como na solução anterior. Nesta solução é possível observar que não houve conflito entre as máquinas em relação a utilização de uma mesma ferramenta compartilhada, assim nenhuma máquina ficou ociosa. Por fim, a segunda solução obteve *makespan* igual a 9 unidades de tempo.

Embora o RCPMS seja um importante problema prático, poucos estudos a seu respeito são encontrados na literatura. Sua origem remonta aos anos 70, com o estudo de Garey e Graham [1975]. Entretanto, somente no início do século 21 o problema foi abordado como apresentado no estudo de caso de Hong et al. [2009]. Nesta publicação, os autores avaliaram duas versões de um algoritmo genético, uma que descarta soluções inviáveis e outra que transforma soluções inviáveis em viáveis. Nove anos após essa publicação, Zhao et al. [2018] propuseram um modelo matemático e uma heurística para solucionar o RCPMS em uma indústria de microeletrônica. No ano seguinte, esse estudo foi estendido com nova aplicação a fabricação de circuitos eletrônicos. Duas novas heurísticas foram propostas, além de duas metaheurísticas [Chung et al., 2019].

Recentemente, Soares e Carvalho [2022] propuseram uma solução baseada no BRKGA hibridizado com quatro buscas locais organizadas em uma descida em vizinhança variável, executada em paralelo. Por não existirem instâncias desafiadoras na literatura, foi proposto um novo conjunto de instâncias, juntamente com valores de limitantes inferiores e superiores para o valor das soluções. Este estudo é considerado atualmente o estado da arte para solucionar o RCPMS.

Apesar de o RCPMS ser um importante problema a ser resolvido, principalmente na indústria de microeletrônica, poucos estudos são encontrados na literatura. Assim, este estudo desenvolve uma implementação paralela para a metaheurística *revenimento paralelo*. Os resultados obtidos demonstram que o PT gera bons resultados, com uma redução de até 48,67% em relação a qualidade da solução.

O restante deste estudo é estruturado em quatro seções. Na Seção 2, o método utilizado para solucionar o SSP é apresentado. A Seção 3 apresenta a implementação paralela do método. Logo depois, na Seção 4, os experimentos comparando a implementação proposta com o estado da arte para solucionar o RCPMS são apresentados. Por fim, na Seção 5, são apresentadas as conclusões.

2. Revenimento paralelo

O revenimento paralelo (ou *parallel tempering*, PT) é um método de amostragem de espaços de busca proposto por Geyer [1991] e formalizado por Hukushima e Nemoto [1996], obtendo a denominação de *replica exchange*. Esse método, como o *simulated annealing*, foi proposto no intuito de mitigar o problema de estagnação em ótimos locais do algoritmo de Metropolis, sendo atualmente um método vastamente utilizado nas áreas de física, química e biologia para resolver problemas de simulação. A sua abordagem consiste na coordenação da constante T do algoritmo de Metropolis, denominada temperatura, e na utilização da distribuição de *Boltzmann* para coordenar a diversificação e a intensificação da busca. Assim, em temperaturas altas, há mais diversificação que intensificação, e em temperaturas baixas, há mais intensificação do que diversificação.

No algoritmo PT, κ réplicas ou cópias do sistema ($R = \{r_1, r_2, r_3, ..., r_\kappa\}$) são simuladas em diferentes temperaturas, ou seja, diferentes valores de T. Cada réplica, a uma temperatura fixa e

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

utilizando o algoritmo de Metropolis, simula o sistema em uma quantidade predefinida de passos. Após esse processo, uma troca de temperatura entre duas réplicas r_i e r_j é proposta. A probabilidade de efetivação da troca é dada pela equação

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

em que $P(r_i \rightarrow r_j)$ corresponde a probabilidade de troca de temperaturas entre as réplicas r_i e r_j; $\Delta\beta = \frac{1}{T_j} - \frac{1}{T_i}$ corresponde à diferença entre o inverso das temperaturas e $\Delta E = E(r_j) - E(r_i)$ corresponde a diferença de energia entre as réplicas, i.e., a diferença entre os valores de avaliação das soluções correspondentes. Esse processo é ilustrado pela Figura 2, em que cada cor corresponde a uma réplica do sistema e cada quadrado corresponde a uma cadeia de Markov homogênea a uma temperatura T_i fixa. A homogeneidade das cadeias implica na não alteração da probabilidade de se atingir uma nova solução dada uma solução anterior. O processo de troca de temperatura entre as réplicas é representado pelas setas na diagonal na Figura 2 e o processo de manutenção da temperatura é representado pelas setas na horizontal.

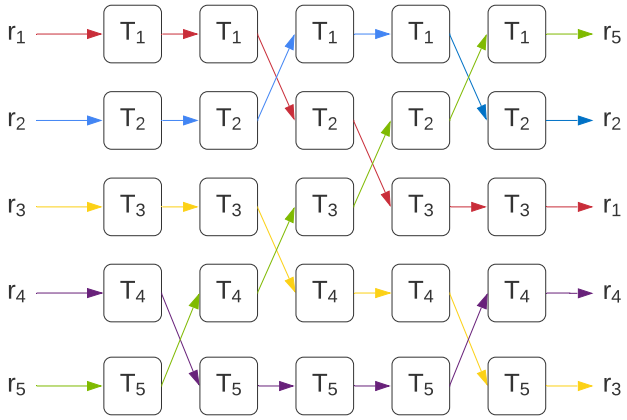


Figura 2: Representação de réplicas e cadeias de Markov no método PT.

Segundo Atchadé et al. [2011]; Rozada et al. [2019], a correta definição das temperaturas de cada réplica desempenha um papel crucial no desempenho do PT. Atualmente, os métodos existentes para definição da distribuição de temperaturas podem ser classificados como estáticos (temperaturas definidas antes do início do algoritmo) ou dinâmicos (temperaturas atualizadas durante a execução do algoritmo, a uma taxa pré-determinada). Os métodos estáticos mais populares incluem as distribuições i) linear; ii) linear-inverso; iii) exponencial; e iv) progressão geométrica. Entre os métodos dinâmicos, destacam-se i) o *feedback-optimized*; ii) a determinação de temperaturas que iguale a probabilidade de taxas de aceitação de trocas; e iii) determinação de temperaturas que gere uma taxa de aceitação de 23%, de acordo com valor recomendado na literatura [Syed et al., 2019].

2.1. Codificação

A codificação utilizada para representar o RCPMS é baseada em um vetor de n itens, em que cada item do vetor principal representa uma tarefa. Entretanto, para definir em qual máquina as tarefas serão processadas, dois vetores auxiliares são utilizados. Esses dois vetores armazenam a posição de inicio e de fim, em relação ao vetor principal, da sequência de tarefas atribuídas a cada máquina. Um exemplo dessa codificação pode ser observada na Figura 3.

Na Figura 3 os retângulos brancos representam os elementos do vetor com seus respectivos valores. A máquina um executa as tarefas $[3, 5, 2]$, a máquina dois as tarefas $[1, 7]$ e a máquina três as tarefas $[6, 4]$, nesta ordem.

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

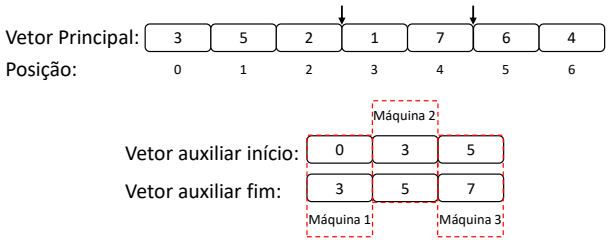


Figura 3: Codificação da solução $\pi = [3, 5, 2, 1, 7, 6, 4]$.

2.2. Decodificação

A decodificação é realizada por um algoritmo em dois passos. No passo 1, a última tarefa da máquina com o maior tempo de término é removida e posteriormente inserida no início da próxima máquina, repetindo esse processo até que a última máquina se torne aquela com o maior tempo de término. Nesse ínterim, é registrada a melhor distribuição das tarefas.

A Figura 4 ilustra o primeiro passo, em que os retângulos brancos representam as tarefas, os retângulos hachurados representam o tempo de troca das ferramentas, os espaços em branco representam o tempo que a máquina ficou ociosa e as máquinas indicadas em vermelho possuem o maior tempo de processamento. Neste exemplo, a ferramenta 1 é compartilhada pelas tarefas 2 e 3, a ferramenta 2 é compartilhada pelas tarefas 1, 4 e 5 e a ferramenta 3 é compartilhadas pelas tarefas 6 e 7.

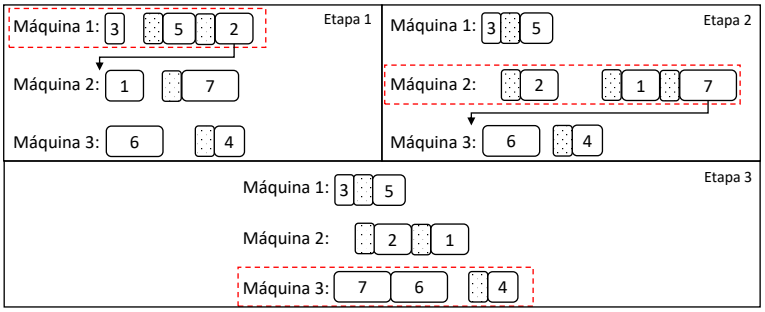


Figura 4: Exemplo do passo um da decodificação.

No passo 2, a tarefa inicial da máquina com o maior tempo de término é removida, posicionando-a no final da máquina anterior. Esse ciclo prossegue até que a primeira máquina se torne a máquina com o maior tempo de término. A Figura 5 ilustra o segundo passo, usando o mesmo padrão da figura anterior.

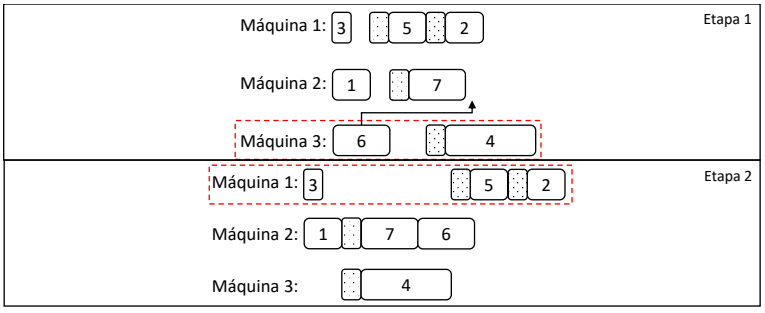


Figura 5: Exemplo do passo dois da decodificação.

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

Ao concluir a execução do algoritmo de decodificação, as melhores distribuições de tarefas obtidas são comparadas, prevalecendo a melhor solução.

2.3. Soluções iniciais

A construção das soluções iniciais leva em consideração a característica de ferramentas compartilhadas do RCPMS. Inicialmente, uma permutação das ferramentas é gerada de forma aleatória a cada temperatura. Em seguida, as tarefas que necessitam de cada uma das ferramentas são inseridas na ordem das ferramentas previamente definidas, gerando um único sequenciamento de tarefas.

Posteriormente, o ponto de corte deste sequenciamento de tarefas é calculado, equilibrando a quantidade de tarefas por máquina de acordo com o valor $|J|/|M|$, em que $|J|$ corresponde a quantidade de tarefas e $|M|$ a quantidade de máquinas. Por fim, o melhor ponto de corte é calculado pela função de avaliação. Um exemplo de duas soluções iniciais para o problema ilustrado na Figura 1 pode ser observado na Figura 6.

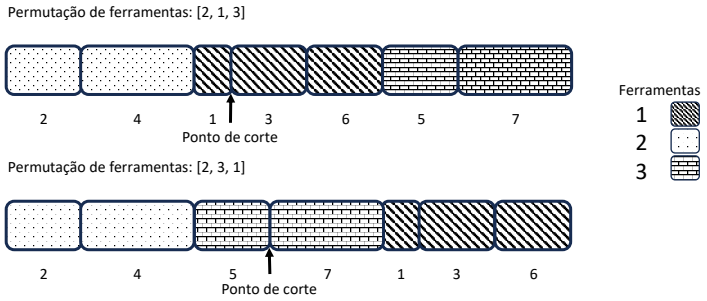


Figura 6: Exemplo de soluções iniciais.

Na Figura 6, inicialmente a permutação de ferramentas $[2, 1, 3]$ é definida de forma aleatória. Em seguida, as tarefas são sequenciadas de acordo com as ferramentas que são necessárias para cada uma delas, gerando a permutação de tarefas $[2, 4, 1, 3, 6, 5, 7]$. Ao final, o ponto de corte inicial, representado pela seta preta, é definido dividindo a quantidade de tarefas pela quantidade de máquinas. Na segunda solução inicial, uma nova permutação de ferramentas é gerada $[2, 3, 1]$, definindo uma nova permutação de tarefas $[2, 4, 5, 7, 1, 3, 6]$. Novamente, ao final da alocação das tarefas, o ponto de corte é adicionado na posição representado pela seta preta. Esse procedimento tem o intuito de começar a busca com o menor número possível de trocas.

2.4. Função de avaliação

O algoritmo proposto para o cálculo do valor da função de avaliação segue uma abordagem iterativa, em que as tarefas são adicionadas uma a uma em suas respectivas máquinas, respeitando uma ordem que é definida e reavaliada a cada nova tarefa inserida. Essa ordem é estabelecida de forma crescente em relação ao tempo de término de cada máquina. No processo de inserção de uma tarefa, o algoritmo realiza um teste para verificar se a ferramenta necessária está em uso. Em caso afirmativo, um tempo ocioso é adicionado à máquina. Posteriormente, é verificada a necessidade de troca da ferramenta, inserindo, caso seja necessário, um tempo de troca à máquina. Por fim, o tempo de processamento da tarefa é somado à máquina correspondente. Ao término do algoritmo, o maior tempo de término entre todas as máquinas determina o *makespan*, que por sua vez representa o valor obtido pela função de avaliação.

2.5. Movimentos

Nas cadeias de Markov, em cada temperatura, um movimento é aplicado de forma a gerar uma nova solução vizinha. Neste estudo, foram avaliados três tipos de movimentos tradicionais para

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

problemas permutacionais: 2-opt, 2-*swap* e a inserção aleatória. Cada um desses movimentos é descrito abaixo.

2-opt: dada uma permutação inicial de tarefas, duas posições são aleatoriamente sorteadas. O intervalo de elementos entre as duas posições sorteadas é invertido dentro da permutação, gerando uma nova solução.

2-*swap*: na primeira etapa, duas posições são aleatoriamente geradas. Na segunda etapa, os elementos das posições geradas são trocados.

Inserção aleatória: duas posições são aleatoriamente geradas. Logo em seguida, o elemento da primeira posição é inserido na segunda posição.

2.6. Critério de parada

Nesta implementação foram utilizados dois critérios de parada. O primeiro foi o número de propostas de trocas entre as temperaturas do PT. O valor utilizado foi definido através de experimentos preliminares descritos na Seção 4.1. O segundo foi o número de propostas de trocas entre as temperaturas do PT sem a melhora da solução, sendo esse valor igual a 10% do número total de propostas de troca. Dentre esses dois critérios, o primeiro que for acionado será responsável por encerrar a execução do algoritmo.

3. Implementação paralela

É proposta uma implementação que utiliza o modelo de programação paralela *dataflow*. O algoritmo é representado como um grafo em que os nós representam segmentos de código e os arcos representam dependências de dados entre eles. Após a criação do grafo, o paralelismo é exposto naturalmente, com cada nó sendo executado conforme suas dependências são satisfeitas, permitindo que instruções independentes sejam executadas paralelamente. Além disso, o padrão *dataflow* proporciona um gerenciamento mais eficiente de seções críticas, contribuindo para melhorias de desempenho.

Internamente, cada nó cujas dependências foram satisfeitas é adicionado a uma fila que controla o fluxo de execução. À medida que os nós são adicionados, cada um deles é vinculado a uma *thread* consumidora, que é responsável por executar o código associado ao nó. No final da execução, a *thread* consumidora notifica todos os nós adjacentes sobre sua conclusão, portanto, se cada nó adjacente não tiver mais dependências, ele será adicionado à fila de execução. Este processo se repete até que não haja mais nós a serem processados. Focando no desempenho do algoritmo, o número total de *threads* consumidoras é igual ao número de núcleos da CPU, evitando assim o custo extra relacionado à troca de contexto entre os núcleos da CPU.

Primeiro, a implementação gera a distribuição de temperaturas iniciais. Simultaneamente, é construído o grafo, composto por três tipos de nós conforme ilustrado na Figura 7. O primeiro tipo de nó, denominado MC, representa o segmento de código relacionado ao cálculo da cadeia de Markov homogênea. Vários nós do tipo MC são executados em paralelo. Neste tipo de nó, uma estrutura de *loop* é usada para executar um procedimento composto por quatro componentes: *move*, *evaluation*, *acceptance* e *exchange*. No primeiro componente, uma solução vizinha é gerada de acordo com um movimento pré-definido. A seguir, o valor da função de avaliação é calculado no segundo componente. Ambos os componentes anteriores dependem do problema, variando de um problema para outro. O próximo componente aplica um critério de aceitação, e se a solução vizinha for eventualmente aceita por este critério, o quarto componente a torna a solução atual do nó.

O segundo tipo de nó, denominado SW, realiza a troca de soluções entre temperaturas que estão ligadas a nós do tipo MC. Este tipo de nó consiste em três componentes: *acquisition*, *acceptance*

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

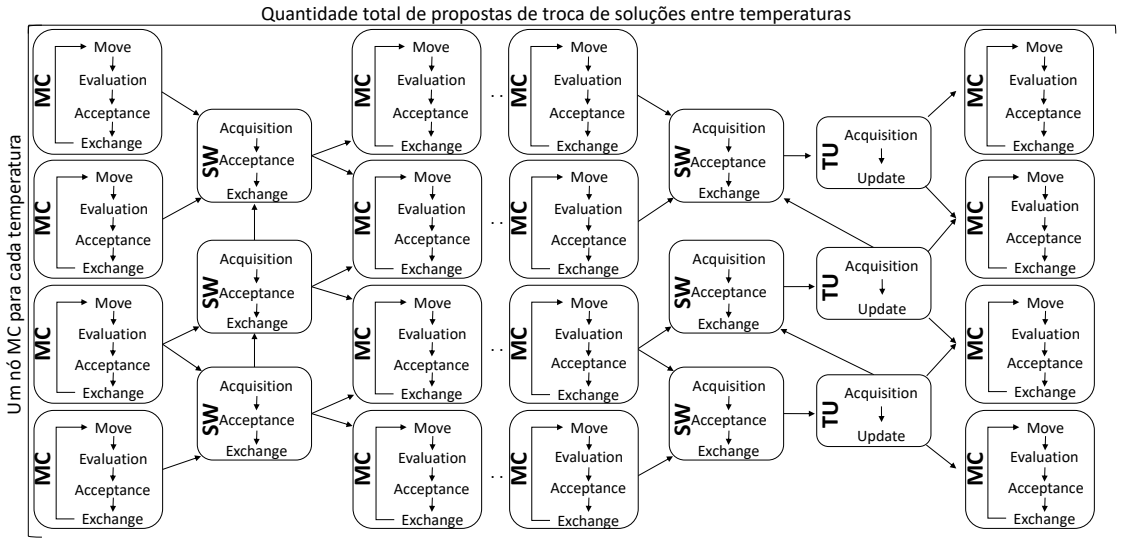


Figura 7: Modelo paralelo proposto.

e *exchange*. Na fase *acquisition*, as soluções e temperaturas vinculadas aos nós MC conectados ao nó SW são recuperadas e transferidas para o componente *acceptance*. Neste componente, as soluções e temperaturas são submetidas a um critério probabilístico de aceitação de trocas, baseado na Equação 1. Se a troca for aceita, as soluções são trocadas entre os nós MC.

Por fim, o terceiro tipo de nó, denominado TU, é responsável por atualizar os valores das temperaturas. Este tipo de nó é gerado a uma taxa pré-definida, executando internamente dois componentes: *acquisition* e *update*. Inicialmente, o primeiro componente recupera informações vinculadas aos nós MC conectados ao nó TU. Esta informação é então passada para o segundo componente, que atualiza as temperaturas de acordo com o método definido.

Após a construção do grafo, o primeiro conjunto de nós MC é enviado para a fila de execução, iniciando uma reação em cadeia que culmina na execução de todos os nós. Além do modelo de programação paralela *dataflow*, é usado o conceito de *threadpool*. Neste conceito, *threads* são criadas apenas uma vez e reutilizadas para executar múltiplas tarefas. Assim, evita-se o custo computacional extra relacionado à criação de novas *threads* para cada nova tarefa. Além disso, este conceito permite um melhor controle das *threads* ativas.

A implementação do PT para problemas discretos descrita nesta seção é disponibilizada na forma de uma API em https://github.com/ALBA-CODES/PTAPI/ sob a licença *Creative Commons* BY-NC (CC BY-NC). Além do funcionamento básico do PT, a API inclui implementações da função de avaliação, dos movimentos e dos métodos estáticos e dinâmicos para determinação de temperaturas descritos na Seção 2.

4. Experimentos computacionais

Um conjunto de experimentos computacionais foi realizado com o objetivo de refinar e definir os parâmetros do PT, além de compará-lo, de forma justa, com o método recente considerado o estado da arte para solução do RCPMS. O ambiente computacional adotado consiste em um computador com dois processadores Intel(R) Xeon(R) CPU E5-2660 v2 2.20GHz 10-Core 20 *threads* e 384 GB de memória RAM, rodando o sistema operacional CentOS Linux 7.9.2009. O método PT foi implementado utilizando a linguagem de programação C++ e compilado utilizando o compilador GCC 10.1.0 e as opções -O3 e -lpthread.

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

Foram utilizadas as instâncias propostas por Soares e Carvalho [2022]. Essas instâncias foram divididas em dois conjuntos denominados RCPMS-I e RCPMS-II. O RCPMS-I é composto por 90 instâncias distribuídas em 18 grupos distintos e o RCPMS-II é composto por 180 instâncias distribuídas entre 36 grupos distintos. Uma síntese das características das instâncias tanto do RCPMS-I quanto do RCPMS-II encontra-se na Tabela 1, em que cada linha representa um conjunto composto por 15 instâncias e três combinações.

Tabela 1: Características das instâncias RCPMS-I e RCPMS-II propostas por Soares e Carvalho [2022].

	m	n	l
RCPMS-I	{2}	{8}	{3, 5, 7}
	{2, 3}	{15}	{4, 5, 7}
	{2, 3, 4}	{25}	{4, 5, 7, 11}
RCPMS-II	{3, 4, 5}	{50}	{4,5, 6, 7, 11}
	{4, 5, 6, 7}	{100}	{5, 6, 7, 8, 11, 13}
	{6, 7, 8, 9, 10}	{200}	{7, 8, 9, 10, 11, 13, 17}

4.1. Experimentos preliminares

Para definir os diversos parâmetros do PT, foi realizado um experimento preliminar baseado no método de configuração automática *irace* [López-Ibáñez et al., 2016]. Neste método, dado um conjunto de instâncias do problema e um conjunto de possíveis valores para cada parâmetro, o *irace* determina, de forma estatística, as melhores combinações de valores dos parâmetros. As instâncias utilizadas correspondem a 10% do total de instâncias, sendo estas selecionadas de forma aleatória. A Tabela 2 lista todos os parâmetros do PT e os valores selecionados pelo *irace*, destacados em negrito na referida tabela.

Tabela 2: Parâmetros utilizados no *irace*.

Parâmetro	Opções
Temperatura inicial	{0,001; 0,01; **0,1**; 0,2}
Temperatura final	{**0,5**; 1; 5; 10}
Número de temperaturas entre a mínima e a máxima	{8; 12; 16; **20**}
Tamanho da cadeia de Markov em cada temperatura	{200; 300; 400; **500**}
Quantidade de propostas de troca entre as temperaturas	{300; **400**; 500; 600}
Distribuição inicial das temperaturas[a]	{1; 2; **3**; 4}
Tipo de movimento[b]	{**1**; 2; 3}
Tipo do ajuste dinâmico das temperaturas[c]	{0; **1**; 2; 3}
Taxa de atualização das temperaturas	{**50**; 100; 150; 200}

[a] 1 - linear, 2 - linear-inverso, 3 - exponencial, 4 - progressão geométrica.
[b] 1 - 2-opt, 2 - 2-*swap*, 3 - inserção aleatória.
[c] 0 - desativado, 1 - 23%, 2 - iguala as taxas de aceitação, 3 - *feedback-optimized*.

4.2. Comparação com o estado da arte

Após a determinação da versão final do algoritmo, o PT foi comparado com o algoritmo proposto por Soares e Carvalho [2022], reconhecido como estado da arte para solução do RCPMS. Este algoritmo é baseado na metaheurística BRKGA e na incorporação de quatro buscas locais que consideram características do RCPMS com o objetivo de intensificar a exploração do algoritmo. Estas buscas locais são organizadas em uma busca em descida variável, executada em paralelo. Tanto

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

o PT quanto o BRKGA hibridizado foram executados no mesmo ambiente computacional. Para cada uma das instâncias, dez execuções independentes foram realizadas.

Os dados obtidos foram sumarizados nas Tabelas 3 e 4. Nas tabelas, são apresentados o número de máquinas, o número de tarefas e o número de ferramentas. Para cada método, BRKGA e PT, são apresentados a média dos melhores valores das soluções geradas durante as dez execuções, o valor médio das soluções e o tempo computacional médio em segundos. Ademais, é apresentada a distância percentual (*gap*) entre as médias das melhores soluções e as médias de todas as soluções. O *gap* foi calculado seguindo a fórmula $gap = \frac{(PT-BRKGA)}{BRKGA} \times 100$. Os melhores resultados são destacados em negrito.

Tabela 3: Resultados dos experimentos para os grupos com 8, 15 e 25 tarefas (RCPMS-I).

m	j	t	BRKGA			PT			*gap*	
			Melhor	Avg	T(s)	Melhor	Avg	T(s)	Melhor(%)	Avg(%)
2	8	3	**161,80**	**161,80**	**0,10**	**161,80**	**161,80**	3,67	0,00%	0,00%
2	8	5	**265,00**	**265,00**	**0,11**	**265,00**	**265,00**	3,59	0,00%	0,00%
2	8	7	**287,00**	**287,00**	**0,11**	**287,00**	**287,00**	3,33	0,00%	0,00%
2	15	3	**276,80**	278,56	**0,32**	**276,80**	**276,80**	3,94	0,00%	-0,63%
2	15	5	**359,80**	**359,80**	**0,29**	**359,80**	**359,80**	3,97	0,00%	0,00%
2	15	7	**330,60**	330,96	**0,34**	**330,60**	**330,60**	3,74	0,00%	-0,11%
3	15	4	201,40	203,96	**0,17**	**201,00**	**201,00**	4,73	-0,20%	-1,45%
3	15	5	217,40	219,00	**0,33**	**217,00**	**217,00**	4,68	-0,18%	-0,91%
3	15	7	**250,00**	251,08	**0,33**	**250,00**	**250,00**	4,44	0,00%	-0,43%
2	25	3	**391,20**	391,38	**0,67**	**391,20**	**391,20**	4,42	0,00%	-0,05%
2	25	5	**444,00**	**444,00**	**0,64**	**444,00**	**444,00**	4,34	0,00%	0,00%
2	25	7	**532,60**	538,50	**0,75**	**532,60**	**532,60**	4,19	0,00%	-1,10%
3	25	4	**335,20**	336,56	**0,04**	**335,20**	**335,20**	6,01	0,00%	-0,40%
3	25	5	**333,20**	333,92	**0,18**	**333,20**	**333,20**	5,97	0,00%	-0,22%
3	25	7	**330,00**	335,00	**0,88**	**330,00**	**330,00**	5,63	0,00%	-1,49%
4	25	5	259,80	265,12	**0,30**	**259,40**	**259,40**	8,16	-0,15%	-2,16%
4	25	7	**242,40**	253,64	**0,70**	**242,40**	**242,40**	8,62	0,00%	-4,43%
4	25	11	**286,00**	293,20	**1,05**	**286,00**	**286,00**	6,37	0,00%	-2,46%

Os dados sumarizados na Tabela 3 estão relacionados com as instâncias de menores dimensões. Embora o conjunto RCPMS-I não represente mais um desafio computacional significativo, como esperado, os resultados obtidos pelo PT demonstram que o método foi capaz de gerar soluções similares às geradas pelo BRKGA. Entretanto, o PT determinou 4 novas melhores soluções. A Tabela 4 apresenta os resultados para as instâncias do conjunto RCPMS-II.

No tocante à qualidade das soluções, o PT obteve resultados iguais ou superiores ao BRKGA para todos os 36 conjuntos de instâncias ao se analisar a melhor solução encontrada. Em 33 conjuntos de instâncias o PT foi melhor que o BRKGA variando o *gap* entre -0,05% e -43,13% e em 3 conjuntos de instâncias houve empate entre os métodos. Esse resultado demonstra que o PT é extremamente competitivo em relação ao BRKGA, obtendo resultados significativamente melhores. Quando se observa o valor médio de todas as soluções, o PT passa a ser melhor que o BRKGA em todos os 36 conjuntos de instâncias, variando o *gap* entre -0,02 e -48,67%, indicando uma melhor consistência do PT em relação ao BRKGA. Em nenhuma das 180 instâncias do conjunto RCPMS-II o PT foi inferior ao BRKGA, obtendo 144 novas melhores soluções.

Ao considerar o tempo de execução, apesar do PT possuir diversos mecanismos paralelos, o BRKGA obteve resultados melhores para 27 dos 36 conjuntos de instâncias. Em somente 9 conjuntos de instâncias o PT obteve um tempo computacional menor. Entretanto, dentre esses 9 conjuntos

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

Tabela 4: Resultado dos experimentos para os grupos com 50, 100 e 200 tarefas (RCPMS-II).

m	j	t	BRKGA			PT			*gap*	
			Melhor	Avg	T(s)	Melhor	Avg	T(s)	Melhor(%)	Avg(%)
3	50	4	**536,00**	536,10	**0,22**	**536,00**	**536,00**	10,91	0,00%	-0,02%
3	50	5	**578,00**	584,14	**3,10**	**578,00**	**578,00**	8,56	0,00%	-1,05%
3	50	7	595,40	601,54	**3,96**	**594,80**	**594,80**	8,80	-0,10%	-1,12%
4	50	5	424,80	440,48	**3,04**	**423,80**	**423,80**	14,25	-0,24%	-3,79%
4	50	7	438,40	460,32	**4,31**	**436,60**	**436,60**	14,70	-0,41%	-5,15%
4	50	11	483,00	504,98	**5,43**	**479,60**	**479,60**	10,39	-0,70%	-5,03%
5	50	6	**339,40**	390,02	**3,09**	**339,40**	**339,40**	23,09	0,00%	-12,98%
5	50	7	366,60	413,84	**4,44**	**364,00**	**364,00**	23,29	-0,71%	-12,04%
5	50	11	393,00	422,00	**5,94**	**387,20**	**387,20**	15,79	-1,48%	-8,25%
4	100	5	779,20	806,56	**14,25**	**778,80**	**778,80**	38,82	-0,05%	-3,44%
4	100	7	811,20	856,68	**25,07**	**797,40**	**797,40**	30,11	-1,70%	-6,92%
4	100	11	869,60	919,58	32,41	**848,80**	**848,80**	**25,23**	-2,39%	-7,70%
5	100	6	628,20	704,00	**21,28**	**625,60**	**625,60**	56,25	-0,41%	-11,14%
5	100	7	699,00	743,94	**25,91**	**653,20**	**653,20**	53,18	-6,55%	-12,20%
5	100	11	700,60	777,60	**33,67**	**676,20**	**676,24**	40,48	-3,48%	-13,03%
6	100	7	650,40	732,30	**27,05**	**581,20**	**581,20**	70,83	-10,64%	-20,63%
6	100	11	633,60	683,54	**34,27**	**560,20**	**560,28**	67,54	-11,58%	-18,03%
6	100	13	635,60	713,12	**34,66**	**579,00**	**579,14**	58,46	-8,90%	-18,79%
7	100	8	614,60	718,24	**34,08**	**476,60**	**477,00**	103,01	-22,45%	-33,59%
7	100	11	589,40	685,18	**39,41**	**485,20**	**485,58**	112,16	-17,68%	-29,13%
7	100	13	587,00	658,22	**40,68**	**498,20**	**499,52**	64,74	-15,13%	-24,11%
6	200	7	1165,20	1325,16	**248,13**	**1039,40**	**1039,44**	330,95	-10,80%	-21,56%
6	200	11	1126,40	1158,02	374,47	**1051,20**	**1053,78**	**104,11**	-6,68%	-9,00%
6	200	13	1217,40	1325,60	386,67	**1075,20**	**1078,80**	**156,13**	-11,68%	-18,62%
7	200	8	1116,40	1281,64	281,56	**925,80**	**926,36**	**272,07**	-17,07%	-27,72%
7	200	11	1061,60	1245,46	328,60	**936,20**	**937,42**	**269,88**	-11,81%	-24,73%
7	200	13	1161,80	1276,12	375,53	**923,20**	**927,54**	**260,51**	-20,54%	-27,32%
8	200	9	1152,60	1407,80	**302,54**	**808,60**	**814,50**	396,70	-29,85%	-42,14%
8	200	11	1138,20	1284,62	**327,79**	**811,20**	**813,78**	372,78	-28,73%	-36,65%
8	200	13	1055,80	1226,26	376,92	**801,20**	**806,54**	**350,73**	-24,11%	-34,23%
9	200	10	1282,40	1481,74	**331,87**	**808,00**	**818,06**	517,15	-36,99%	-44,79%
9	200	11	907,60	947,30	358,08	**757,20**	**759,84**	**277,13**	-16,57%	-19,79%
9	200	13	1006,40	1099,56	**364,79**	**712,80**	**716,16**	417,87	-29,17%	-34,87%
10	200	11	1079,00	1174,14	**320,44**	**790,20**	**794,26**	517,93	-26,77%	-32,35%
10	200	13	1202,00	1348,04	**348,21**	**683,60**	**691,90**	695,33	-43,13%	-48,67%
10	200	17	826,60	892,58	373,99	**623,60**	**627,02**	**327,04**	-24,56%	-29,75%

de instâncias, em 8 a redução foi obtida nas maiores instâncias, o que sugere que o PT é capaz de escalar à medida que a quantidade de tarefas aumenta. Além disso, o PT não obteve um tempo de execução proibitivo, variando na média entre 8,56 à 695,33 segundos, valores aceitáveis em ambientes industriais.

5. Conclusão

Nesta aplicação do revenimento paralelo (ou *parallel tempering*, PT), abordou-se o problema de sequenciamento de tarefas em máquinas flexíveis paralelas com compartilhamento de recursos, encontrado frequentemente na indústria microeletrônica. Uma implementação paralela da metaheurística PT foi desenvolvida e aplicada. Os resultados apresentados representam um avanço significativo no estado da arte, estabelecendo novos melhores resultados para 82,22% das instâncias

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878

avaliadas, em um tempo de execução razoável. Os resultados indicam um desempenho notável do PT em uma variedade de cenários, consolidando sua posição como uma metaheurística eficaz para diferentes problemas de sequenciamento de tarefas em máquinas flexíveis.

6. 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

Atchadé, Y. F., Roberts, G. O., e Rosenthal, J. S. (2011). Towards optimal scaling of metropolis-coupled markov chain monte carlo. *Statistics and Computing*, 21:555–568.

Chung, T., Gupta, J. N., Zhao, H., e Werner, F. (2019). Minimizing the makespan on two identical parallel machines with mold constraints. *Computers & Operations Research*, 105:141–155.

Garey, M. R. e Graham, R. L. (1975). Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4(2):187–200.

Geyer, C. J. (1991). Markov chain monte carlo maximum likelihood. In *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*, p. 156–163. Interface Foundation of North America.

Hong, T.-P., Jou, S.-S., e Sun, P.-C. (2009). Finding the nearly optimal makespan on identical machines with mold constraints based on genetic algorithms. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, Hangzhou, China. World Scientific and Engineering Academy and Society.

Hukushima, K. e Nemoto, K. (1996). Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Rozada, I., Aramon, M., Machta, J., e Katzgraber, H. G. (2019). Effects of setting temperatures in the parallel tempering monte carlo algorithm. *Physical Review E*, 100(4):043311.

Soares, L. C. e Carvalho, M. A. (2022). Application of a hybrid evolutionary algorithm to resource-constrained parallel machine scheduling with setup times. *Computers & Operations Research*, 139:105637.

Syed, S., Bouchard-Côté, A., Deligiannidis, G., e Doucet, A. (2019). Non-reversible parallel tempering: A scalable highly parallel mcmc scheme. *arXiv preprint arXiv:1905.02939*.

Zhao, H. D., Gao, J., e Zhu, F. (2018). Scheduling on parallel machines with mold constraints. *IOP Conference Series: Materials Science and Engineering*, 389(1):012005.

DOI: 10.59254/sbpo-2024-193878
http://live-proceedings.pantheonsite.io/p/193878