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

UMA ABORDAGEM HEURÍSTICA PARA UM PROBLEMA MULTI-OBJETIVO DE SEQUENCIAMENTO EM MÁQUINAS PARALELAS COM CONSIDERAÇÕES AMBIENTAIS

Clarissa Maria Rodrigues de Oliveira, Matheus Lopes Bittencourt, Raphael Kramer
Universidade Federal de Pernambuco
Departamento de Engenharia de Produção
Av. da Arquitetura, s/n – Cidade Universitária, Recife - PE, Brasil, CEP: 50740-550
{clarissa.maria, matheus.bittencourt, raphael.kramer}@ufpe.br

RESUMO

Este artigo aborda um problema multi-objetivo de sequenciamento em máquinas paralelas com foco em eficiência energética, visando minimizar o $makespan$ e o consumo de energia, considerando tarifas que variam ao longo do dia, política comumente conhecida como *Time-of-use* (TOU). Propomos uma heurística que combina uma proposta recente com a meta-heurística Busca Local de Pareto de Duas Fases, integrando estratégias específicas para este desafio. Avaliamos a qualidade das soluções usando indicadores de desempenho como *Hipervolume*, *Pureza* e D_r, que mensuram a qualidade da fronteira de soluções. A comparação dos algoritmos foi realizada com o Teste de Wilcoxon, evidenciando a superioridade da abordagem proposta neste trabalho em relação à anterior em todas as medidas de performance, para instâncias pequenas e grandes. Esta pesquisa amplia o conhecimento científico e fomenta a aplicação prática em sistemas de manufatura, visando a otimização dos recursos e a redução dos impactos ambientais.

PALAVRAS CHAVE. Sequenciamento de Produção. Sequenciamento Verde. Otimização Bi-Objetivo. Consumo Energético. Heurísticas.

OC – Otimização Combinatória; MH – Metaheurísticas.

ABSTRACT

This article addresses a multi-objective scheduling problem on parallel machines with a focus on energy efficiency, aiming to minimize the makespan and energy consumption, considering tariffs that vary throughout the day, a policy commonly known as Time-of-Use (TOU). We propose a heuristic that combines a recent proposal with the meta-heuristic Two-Phase Pareto Local Search, integrating specific strategies for this challenge. We assess the quality of solutions using performance indicators such as *Hipervolume*, *Purity*, and D_r, which measure the quality of the solution frontier. The comparison of algorithms was performed with the Wilcoxon Test, highlighting the superiority of the proposed approach in this work over the previous one in all performance measures, for small and large instances. This research expands scientific knowledge and promotes practical application in manufacturing systems, aiming at resource optimization and reduction of environmental impacts.

KEYWORDS. Production Scheduling. Green Scheduling. Bi-Objective Optimization. Energy Consumption. Heuristics.

CO - Combinatorial Optimization; MH - Metaheuristics.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

1. Introdução

Um dos problemas mais estudados na otimização combinatória é o sequenciamento da produção, que visa determinar a ordem ideal das atividades [Afzalirad e Shafipour, 2018]. Os ambientes de manufatura, como máquina única, máquinas paralelas, *flow shop*, *job shop* e *open shop*, influenciam o sequenciamento adequado [Pinedo, 2016]. O problema de sequenciamento em máquinas paralelas, comum em diversas indústrias (e.g., usinagem e soldagem [Wang et al., 2018] e indústria química [Wu et al., 2022]), busca otimizar medidas de desempenho como o atraso ponderado de entrega e o $makespan$ [Pinedo, 2016].

Além disso, a programação da produção pode incorporar questões ambientais para reduzir custos e impactos, considerando indicadores como emissões de carbono [Xue et al., 2019] e consumo total de energia. A última pode ser avaliada por meio de uma política de tarifação de energia por horário de uso (*Time-of-use*, TOU), simulando a situação prática do mercado de eletricidade de períodos de pico, vale e normal [Che et al., 2017], pelo ajuste de níveis de velocidade das máquinas, de forma a regular o consumo de energia e o tempo de processamento das tarefas [Wu e Che, 2019] e pela decisão sobre os diferentes estados das máquinas, como em processamento, desligada e ociosa, com atribuição de diferentes níveis de consumo energético para cada estado [Antoniadis et al., 2020]. As medidas de desempenho operacionais, em geral, são conflitantes com os objetivos ambientais. Visto isso, a literatura recente utiliza modelagens matemáticas multiobjetivo para encontrar soluções ótimas que equilibram eficiência operacional e sustentabilidade ambiental [Wang e Qi, 2023].

O Problema de sequenciamento em máquinas paralelas com considerações ambientais (PSMPCA) é uma extensão do Problema de sequenciamento em máquinas paralelas (PSMP), o qual é classificado como NP-difícil, i.e., a resolução desse tipo de problema demanda um tempo de processamento computacional não polinomial para auferir a solução ótima [Yin et al., 2017]. Visto isso, métodos heurísticos são comumente empregados para resolvê-lo. Essa pesquisa apresenta como objetivo geral propor um algoritmo heurístico para resolver um PSMPCA multi-objetivo.

2. Descrição do problema

O modelo selecionado para estudo foi proposto inicialmente por Wang et al. [2018] e estudado em Anghinolfi et al. [2021], no qual envolve o sequenciamento verde a medida que busca a minimização do custo total energético, chamado de TEC, e do *makespan*.

O problema consiste em sequenciar N tarefas independentes em M máquinas paralelas idênticas. Todas as tarefas estão disponíveis para processamento no tempo 0 (i.e., no início do horizonte de tempo), e cada máquina pode processar apenas uma tarefa por vez. Seja $J = \{1, \ldots, N\}$ o conjunto de tarefas a serem processadas, e $H = \{1, \ldots, M\}$ um conjunto de máquinas. Para cada tarefa $j \in J$, o tempo de processamento p_j é idêntico para todas as máquinas. As tarefas devem ser sequenciadas em um horizonte de tempo que consiste em um conjunto $T = \{1, \ldots, K\}$ de intervalos de tempo. As tarefas não podem ser interrompidas, i.e., cada tarefa j deve ser processada em um conjunto S_j de p_j intervalos de tempo consecutivos em uma única máquina $h \in H$.

Devido aos custos variáveis de energia na abordagem TOU, os intervalos de tempo em T são organizados em grupos, onde cada grupo engloba uma série de intervalos de tempo que compartilham o mesmo preço de energia. Ou seja, um determinado preço da energia c_t é atribuído a cada intervalo de tempo $t \in T$. Cada máquina $h \in H$ está associada a uma taxa fixa de consumo de energia e_h, e o custo do consumo de energia associado ao processamento da tarefa j na máquina h é $e_h \sum_{t \in S_j} c_t$.

O tempo de conclusão C_j de uma tarefa j é o tempo final do último intervalo de tempo designado para a tarefa. Além disso, o C_{max} de um sequenciamento é o tempo de término do último $j \in J$, ou seja, $C_{max} = max\{C_j : j \in J\}$. O valor do TEC está associado ao sequenciamento definido como $\sum_{h \in H} e_h \sum_{j \in J_h} \sum_{k \in S_j} c_k$, onde J_h é o conjunto de tarefas sequenciadas na máquina

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

h. Dessa forma, as variáveis desse problema são: $X_{jht} \in \{0,1\}, j \in J, h \in H, t \in T$, que assume valor 1 se a tarefa j for processada na máquina h começando no início do intervalo de tempo t, e 0 caso contrário; $C_j \geq 0, j \in J$, referente ao tempo de conclusão da tarefa j; $C_{max} \geq 0$; e $TEC \geq 0$. Diante disso, este problema pode ser formulado da seguinte maneira:

$$\text{Min } C_{max} \tag{1}$$

$$\text{Min } TEC \tag{2}$$

$$\text{sujeito a } TEC = \sum_{h \in H} e_h \sum_{j \in J} \sum_{t=1}^{k-p_j+1} X_{jht} \left(\sum_{i=t}^{t+p_j-1} c_i \right), \tag{3}$$

$$\sum_{h \in H} \sum_{t=1}^{K-p_j+1} X_{jht} = 1, \qquad \forall\, j \in J, \tag{4}$$

$$\sum_{j \in J} \sum_{i=max\{0,t-p_j+1\}}^{t} X_{jhi} \leq 1, \qquad \forall\, t \in T, h \in H, \tag{5}$$

$$C_j = \sum_{h \in H} \sum_{t=1}^{K-p_j+1} (t - p_j + 1)\, X_{jht}, \qquad \forall\, j \in J, \tag{6}$$

$$C_{max} \geq C_j, \qquad \forall\, j \in J, \tag{7}$$

$$C_{max} \leq K, \tag{8}$$

$$C_{max} \geq 0, TEC \geq 0, C_j \geq 0, \qquad \forall\, j \in J, \tag{9}$$

$$X_{jht} \in \{0,1\}, \qquad \forall\, j \in J, h \in H, t \in T. \tag{10}$$

As Expressões (1) e (2) representam as funções objetivos a serem minimizadas, sendo elas C_{max} e o TEC, respectivamente. A Equação (3) define o cálculo do TEC. As Restrições (4) impõem que cada tarefa seja atribuída a uma única máquina, garantindo que o processamento de qualquer tarefa comece em um único intervalo de tempo em uma única máquina. O conjunto de Restrições (5) garantem que no máximo uma única tarefa seja processada em cada intervalo de tempo em cada máquina. As Restrições (6) fornecem os horários de conclusão das tarefas, impondo implicitamente a não preempção. As Inequações (7) definem o C_{max}, enquanto que a Inequação (8), impõe que o C_{max} não deva exceder o horizonte temporal disponível (observe que, devido a Equação (8), o problema pode não ser solucionável se K não for suficientemente grande). Finalmente, as Restrições (9) e (10) definem os domínios das variáveis.

O modelo abordado é definido como um Problema de Otimização Multiobjetivo (POM). Conforme Xue et al. [2019], um POM pode ser formalmente descrito da seguinte maneira: considerando um problema de minimização, para uma dada solução $x \in B$, i.e., x é um vetor de decisão pertencente ao espaço de soluções viáveis B, deseja-se minimizar $f(x) = f_1(x), f_2(x), \ldots, f_q(x)$, sendo $f_1, f_2, \ldots, f_q$ objetivos conflitantes. Diante disso, uma solução viável a domina a solução viável b (com notação $a \succ b$), se $f_l(a) \leq f_l(b)$ para todo $l \in \{1, 2, \ldots, q\}$ e se $f_l(a) < f_l(b)$ para pelo menos uma função objetivo $l \in \{1, 2, \ldots, q\}$.

De acordo com Bérubé et al. [2009], alguns conceitos importantes são apresentados:

Definição 2.1 (Solução eficiente e Conjunto eficiente) *Diz-se que uma solução $x \in B$ é (Pareto) eficiente se não existe uma outra solução $x' \in B$ de tal modo que $x' \succ x$. E conjunto eficiente agrupa todas as soluções ditas eficientes, isto é, $A = \{x \in B : x$ é eficiente em $B\}$.*

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Definição 2.2 (Fronteira de Pareto) *Agrupa as imagens do conjunto A no espaço dos objetivos, ou seja, $P = \{f_l(x) \; \forall \, l \in \{1, 2, ..., q\} : x \in A\}$.*

Definição 2.3 (Ponto Ideal e Ponto de Nadir) *Para um problema bi-objetivo, o ponto ideal $p^I = (f_1^I, f_2^I)$, no qual, $f_1^I = min\{f_1(x)\}$ e $f_2^I = min\{f_2(x)\}$, $x \in A$. E o ponto de Nadir é entendido como $p^N = (f_1^N, f_2^N)$, no qual, $f_1^N = \{f_1(x) : f_2(x) = f_2^I\}$ e $f_2^N = \{f_2(x) : f_1(x) = f_1^I\}$, $x \in A$.*

O conjunto de soluções encontradas pelos métodos de otimização multi-objetivo podem ser comparados conforme métricas de performance. Para esse trabalho, os indicadores de performance considerados são: **(a) *Pureza***: Mede a quantidade de soluções que têm os mesmos valores de TEC e C_{max} quando comparadas a uma fronteira de referência. Os valores de *Pureza* variam de 0 a 1, com valores mais altos indicando uma fronteira de melhor qualidade; **(b) $\mathbf{D_r}$**: Calcula a distância euclidiana entre os pontos do método avaliado e uma fronteira de referência. Valores menores de D_r são desejáveis, indicando soluções mais próximas da fronteira ideal; **(c) *Hipervolume***: Avalia a aproximação e dispersão das soluções em relação a um ponto de referência, que é o Ponto de Nadir da união de todas as fronteiras obtidas. Um maior *Hipervolume* indica soluções menos dispersas e mais próximas da fronteira de referência.

Nesse trabalho, a fronteira de referência foi definida por meio de um algoritmo restrição-ϵ exato para pequenas instâncias ou pela união das soluções encontradas por todos os métodos, removendo as soluções dominadas, para grandes instâncias.

3. Algoritmos implementados

A heurística de sequenciamento apresentada a seguir foi desenvolvida por Anghinolfi et al. [2021]. Testes computacionais revelaram que essa abordagem supera significativamente outras alternativas comparadas, como as abordagens evolutivas multi-objetivo NSGA-III e MOEA/D. Esse algoritmo é definido como Sequenciamento Guloso Dividido (SGS) e combina uma heurística construtiva gulosa aleatória que considera a alocação preemptiva de tarefas, buscando aproveitar períodos de menor custo energético, com uma heurística de refinamento (ES) que busca melhorar o TEC sem comprometer o C_{max} por meio de trocas de blocos. Para facilitar a compreensão, algumas definições de Anghinolfi et al. [2021] são necessárias.

Definição 3.1 (Unidades de tempo adjacentes e consecutivas) *Duas unidades de tempo k e $k+1$ na mesma máquina são consideradas adjacentes, e duas unidades de tempo k e $k+t$ na mesma máquina são consideradas como unidades de tempo consecutivas, se todas as unidades de tempo de $k+1$ até $k+t-1$ forem utilizadas para processar tarefas, ou seja, não estiverem livres.*

As unidades de tempo são consideradas livres se não estiverem sendo processadas tarefas nelas. Uma localização l que contém as unidades de tempo $F \subseteq T$ em uma máquina $h \in H$ é denotada como $l = (h, F)$. O custo energético dessa localização, EC_l, é dado por $e_h \sum_{t \in F} c_t$.

Definição 3.2 (Localização livre e Localização dividida) *Uma localização livre para uma tarefa j é uma localização (h, F) que agrupa apenas unidades de tempo adjacentes e/ou consecutivas livres da máquina h, de modo que $|F| = p_j$. E uma localização dividida é uma localização que inclui pelo menos duas unidades de tempo livres consecutivas.*

Uma localização atribuída para uma tarefa j na máquina h é representada como (h, A_j), onde A_j é o conjunto de unidades de tempo adjacentes atribuídas à tarefa j e $|A_j| = p_j$. Observe que o horário de início de uma tarefa j, s_j, é igual a $min_{t \in A_j}\{t\}$.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Definição 3.3 (Sequenciamento Viável) *Um sequenciamento viável* $S = \{(h, A_j), j \in J : h \in H, A_j \subseteq T\}$ *é um conjunto de atribuições de tarefas a localizações não divididas, de modo que* (h, A_j) *e* $(h, A_{j'}), A_j \cap A_{j'} = \emptyset$.

O exemplo da Figura 1a) mostra duas tarefas, j e j', com tempos de processamento $p_j = 3$ e $p_{j'} = 2$, respectivamente. Essas tarefas são atribuídas a localizações com pares de unidades de tempo adjacentes, formadas pela localização $l = (h, I_j)$, onde $I_j = \{3, 4, 5\}$ e (h, I'_j), onde $I_{j'} = \{9, 10\}$, enquanto que j'', com tempo de processamento $p_{j''} = 2$, é atribuída a uma localização com um par de unidades de tempo consecutivas, chamada de localização dividida, $l = (h, I_{j''})$, onde $I_{j''} = \{8, 11\}$.

Se uma tarefa j é atribuída a uma localização dividida, diz-se que j é agendada dividida em l. Se pelo menos uma tarefa é agendada dividida em l, temos um sequenciamento dividido, que é o equivalente a um sequenciamento com preempção.

Definição 3.4 (Sequenciamento Dividido) *Um sequenciamento dividido* $S = \{(h, I_j), j \in J : h \in H, I_j \subseteq T\}$ *é um conjunto de localizações atribuídas e localizações divididas atribuídas de tal forma que, para toda localização* (h, I_i) *e* (h, I_j) *atribuídas às tarefas* $i, j \in J$, $I_i \cap I_j = \emptyset$, *se* $i \neq j$.

Um sequenciamento viável apresenta apenas locais atribuídos às tarefas com unidades de tempo adjacentes na mesma máquina e satisfaz a condição de não sobreposição para tarefas atribuídas à mesma máquina, ou seja, não pode haver intervalos de tempo comuns. Visto isso, um sequenciamento dividido S' pode ser convertido em um sequenciamento viável S (com tarefas atribuídas a localizações com unidades de tempo adjacentes apenas) sem alterar os valores de TEC e *makespan*. Portanto, pode-se afirmar que um sequenciamento dividido S' é equivalente a S, se $C_{\max}(S') = C_{\max}(S)$ e $TEC(S') = TEC(S)$.

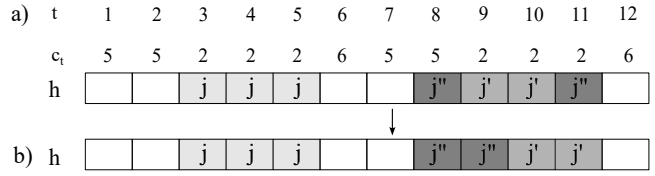

Figura 1: Exemplo de conversão de um sequenciamento dividido em um viável.

O sequenciamento dividido pode ser convertido em um viável sem alterar os valores de TEC e C_{max}, uma vez que não altera os períodos de tempo utilizados para processar tarefas. Na Figura 1a), apresenta-se um sequenciamento com a atribuição das tarefas j e j' à máquina h em localizações $(h, \{3, 4, 5\})$ e $(h, \{9, 10\})$, com custo energético $EC_{I_j} = 6$ e $EC_{I_{j'}} = 4$, assumindo $e_h = 1$, respectivamente. A tarefa j'' é atribuída à mesma máquina na localização dividida $(h, \{8, 11\})$, com $EC_{I_{j''}} = 7$. Portanto, temos um sequenciamento dividido com $TEC = 17$ e $C_{max} = 11$, o qual pode ser convertido em um sequenciamento viável equivalente, conforme mostrado na Figura 1b), com a atribuição das tarefas j, j' e j'' aos respectivos locais $(h, \{3, 4, 5\})$, $(h, \{10, 11\})$ e $(h, \{8, 9\})$, que apresentam apenas unidades de tempo adjacentes.

A heurística construtiva é chamada de Heurística Gananciosa Dividida (SGH), apresentada no Algoritmo 1, e constrói uma solução inicial minimizando o TEC dentro de um horizonte de tempo definido. O processo envolve definir localizações livres e divididas nas máquinas. Tarefas são alocadas em localizações, e sequenciamentos divididos são convertidos em sequenciamentos viáveis, que não alteram os valores de TEC e C_{max}.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Algoritmo 1: `Heurística_gananciosa_dividida` (J, H, T)

1 $S \leftarrow \{S_h, h \in H\}, S_h = \emptyset, h \in H$;
2 Construir uma lista $\hat{P}$ dos elementos de P_J e os ordenar de forma não-crescente;
3 Para todo $p \in P_J$, construir uma lista $F_p \leftarrow \{j \in J : p_j = p\}$;
4 **for** $p \in \hat{P}$ **do**
5 **for** $h \in H$ **do**
6 Construir uma lista L_{ph} com as localizações alternativas na máquina h para alguma tarefa j, no qual, $p_j = p$;
7 **end**
8 **for** $j \in F_p$ **do**
9 **if** $L_{ph} = \emptyset \; \forall \; h \in H$ **then**
10 **return** *"Não existe solução viável"*
11 **end**
12 **else**
13 Selecionar aleatoriamente a localização $\hat{l} = (\hat{h}, \hat{I})$ dentre as localizações de menor custo energético provenientes de $\cup_{h \in H} L_{ph}$;
14 Designar a localização $\hat{l}$ para a tarefa j e adicionar $\hat{l}$ em $S_{\hat{h}}$;
15 Remover de $L_{p\hat{h}}$ qualquer localização $(\hat{h}, I)$ no qual $I \cap \hat{I} \neq \emptyset$;
16 Construir as novas localizações divididas $(\hat{h}, I')$, com $|I'| = p$ e inserir em $L_{p\hat{h}}$;
17 **end**
18 **end**
19 **end**
20 **if** *O sequenciamento S construído é dividido* **then**
21 `Converter_sequenciamento`(S);
22 **end**
23 **return** S

Com base no Algoritmo 1, a construção de uma solução inicial começa com a criação de uma lista $\hat{P}$ dos tempos de processamento distintos $p \in P_J$ das tarefas $j \in J$, ordenados de forma não-crescente (Linha 2). Em seguida, é construída uma lista de listas F, onde F_p armazena as tarefas $j \in J$ com o tempo de processamento p (Linha 3). Para cada tempo de processamento distinto $p \in \hat{P}$, as tarefas correspondentes são alocadas em localizações livres (possivelmente divididas) visando obter o menor custo energético (Linhas 4 a 19).

Esse procedimento começa com a construção de uma lista L_{ph} das localizações livres e livres divididas disponíveis para alocar uma tarefa com tempo de processamento p para cada máquina $h \in H$ (Linhas 5 a 7). As localizações livres (e divididas) são construídas ao longo dos intervalos de tempo, iterando uma unidade até alcançar K^{max}, i.e., o maior período do horizonte de tempo. Uma fila é definida e, se a unidade de tempo t é considerada livre, ela é enfileirada até que se alcance uma capacidade de p unidades, indicando que uma nova localização (possivelmente dividida) foi encontrada. O custo energético desse local pode ser definido pelo custo do local anterior retirando o custo da unidade de tempo mais antiga da fila e acrescentando a unidade de tempo mais atual da fila à nova localização.

Para cada tarefa j com tempo de processamento p (ou seja, $j \in F_p$), uma localização $\hat{l}$ dentre as de menor custo energético, considerando todas as máquinas, é selecionada aleatoriamente (Linha 13). Essa seleção uniforme aleatória foi definida em Anghinolfi et al. [2021], pois observaram que a escolha aleatória produziu melhores resultados de TEC em comparação com uma escolha determinística que prioriza a alocação das tarefas nos primeiros períodos de tempo.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Em seguida, a localização selecionada $\hat{l} = (\hat{h}, \hat{I})$ é designada para a tarefa j e adicionada ao sequenciamento da máquina $\hat{h}$ (Linha 13). Posteriormente, nas Linhas 15 e 16, a lista $L_{p\hat{h}}$ é atualizada removendo as localizações $l' \in L_{p\hat{h}}$ onde $I' \cap \hat{I} \neq \emptyset$ e adicionando as novas localizações divididas que surgem da designação de j para $\hat{l}$. Se não houver localização livre em qualquer uma das máquinas para alocar uma tarefa, isso indica um problema sem solução viável, uma vez que K^{max} é muito restrito para sequenciar todas as tarefas (Linhas 9 a 11).

O algoritmo `Converter_sequenciamento` é usado para ajustar sequenciamentos divididos em viáveis, garantindo que as tarefas sejam alocadas a unidades de tempo adjacentes sem sobreposição. A solução obtida pela heurística construtiva SGH pode ser aprimorada por meio de uma busca baseada em trocas, conhecida como ES. Para uma compreensão mais profunda dessa abordagem, são necessárias as seguintes definições.

Definição 3.5 (Sequência de Períodos Trocáveis - EPS) *Um bloco EPS é um subconjunto E de intervalos de tempo adjacentes em uma máquina específica h dentro de um cronograma viável S. Em um EPS, se uma unidade de tempo é utilizada para processar uma tarefa j, então todas as unidades de tempo atribuídas à tarefa j devem estar contidas em E.*

Na Figura 2 o subconjunto $E_1 = \{1, 2, 3\}$ na máquina h não pode ser considerado um EPS, uma vez que a unidade de tempo 3 é utilizada para executar parte da tarefa 1, porém E_1 não contém todos os períodos utilizados para processar a tarefa 1, ou seja, os períodos 4 e 5 não estão contidos em E_1. Nesse caso, possíveis EPS's de tamanho 3 seriam E_2, que abrange todos os períodos utilizados para processar a tarefa 1, e E_3 e E_4, que englobam todos os períodos usados para processar a tarefa 2.

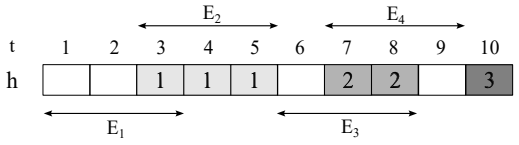

Figura 2: Exemplo para identificar uma EPS.

Definição 3.6 (Troca de EPS) *Uma troca entre dois EPS's E e E', pertencentes às máquinas h e h', respectivamente, tal que $|E| = |E'|$ e $E \cap E' = \emptyset$ se $h = h'$, é um procedimento que permite o sequenciamento das tarefas atribuídas às unidades de tempo em E, na máquina h, para as unidades de tempo em E', na máquina h', e vice-versa, sem alterar a ordem de atribuição das tarefas dentro das unidades de tempo de E e E'.*

Definição 3.7 (Rearranjo de EPS) *Um rearranjo de um EPS E em um sequenciamento viável S é um procedimento que reatribui o conjunto de tarefas sequenciadas em E, denotado por $J_E(S)$, em localizações com períodos de tempo adjacentes.*

A Figura 3 exemplifica uma troca e rearranjo de EPS. Na Figura 3a), temos um sequenciamento viável S de 5 tarefas. Nesta ilustração, o EPS $E = \{8, 9, 10\}$ na máquina h é trocado com o EPS $E' = \{2, 3, 4\}$ na máquina h', resultando no sequenciamento em 3b). Em seguida, o rearranjo em E é realizado, resultando no novo sequenciamento em 3c). Esse rearranjo tem como objetivo atribuir as tarefas envolvidas na troca nos períodos de tempo de menor custo energético.

Ainda, conforme ilustrado nas Figuras 2 e 3, é possível notar que nem todas as unidades de tempo em um EPS são atribuídas às tarefas. Nesse caso, podemos distinguir um EPS-J de um EPS-I.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

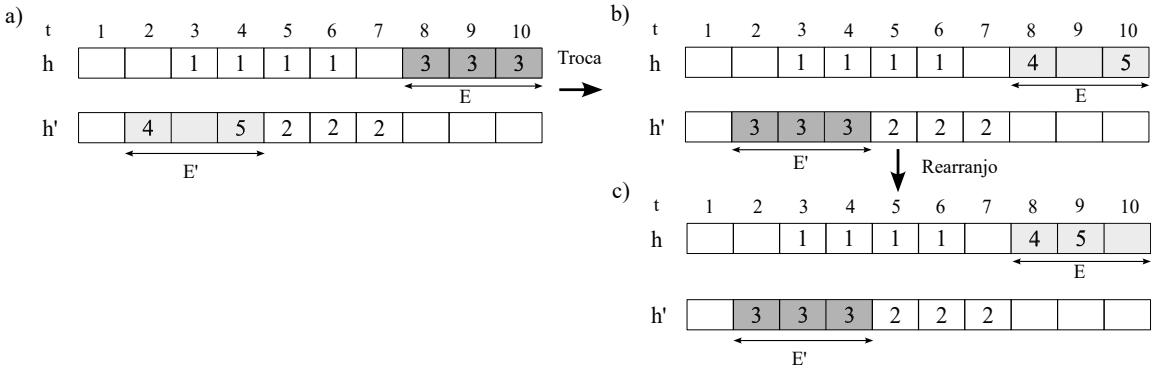

Figura 3: Exemplo para identificar troca e rearranjo.

Definição 3.8 (Sequência de Períodos Trocáveis - J (EPS-J)) *Um EPS-J é um EPS que contém uma única tarefa, sem nenhuma unidade de tempo ociosa.*

Definição 3.9 (Sequência de Períodos Trocáveis - I (EPS-I)) *Um EPS-I é um EPS que contém no mínimo uma unidade de tempo ociosa.*

Diante disso, a troca de blocos só pode ocorrer entre um EPS-J e um EPS-I que possuem a mesma cardinalidade de E. Durante esse procedimento, todas as tarefas contidas no EPS-J são atribuídas ao EPS-I, enquanto as tarefas contidas no EPS-I são rearranjadas no EPS-J, caso existam. Esse procedimento combinado de troca e rearranjo é denominado "movimento". No caso da Figura 3a), o EPS $E = \{8, 9, 10\}$ na máquina h é caracterizado como um EPS-J e o EPS $E' = \{2, 3, 4\}$ na máquina h' é um EPS-I, ambos com cardinalidade igual a 3.

O pseudocódigo da busca local pode ser visto no Algoritmo 2. Este procedimento busca refinar a solução utilizando uma política de primeira melhoria, reduzindo o TEC sem aumentar o $makespan$. Inicialmente todos blocos denominados de EPS-J e EPS-I para cada tempo de processamento $p \in \hat{P}$ são criados e armazenados em $blocos\_epsi$ e $blocos\_epsj$, respectivamente.

O EPS-J de cada tarefa $i \in F_p$ com tempo de processamento p é formado pelo subconjunto de T, $E_J = \{s_j, s_j + 1, \ldots, s_j + p_j - 1\}$, na máquina h_J, onde s_j é o tempo de início da tarefa j. A identificação de um conjunto EPS-I, formado por p unidades de tempo adjacentes, $E_I = \{s, s+1, \ldots, s+p-1\}$ na máquina h_I é feita de tal modo que o primeiro período do subconjunto E_I, s, seja um tempo de início de uma tarefa $j \in J$ ou um período ocioso, e o último período de E_I, $s + p - 1$, deve ser um tempo de conclusão de uma tarefa $j \in J$ ou um período ocioso. No entanto, caso s seja um tempo de início e $s + p - 1$ seja um tempo de conclusão, deve existir pelo menos uma unidade de tempo livre entre eles. Os subconjuntos de T formados por períodos adjacentes são testados de 1 até K^{max}, iterando de uma em uma unidade até completar um subconjunto de períodos de tempo de tamanho p, para cada máquina.

Para cada tempo de processamento distinto (Linha 4) e para cada EPS-J de $blocos\_epsj$ (Linha 5) e EPS-I de $blocos\_epsi$ (Linha 6) de cardinalidade igual a p, i.e., $|EPS\_J_p| = |EPS\_I_p| = p$, a função `Viabilidade` (Linha 7) verifica se, no melhor caso de movimentação, o TEC proveniente desta ($novo\_tec$) é melhor que o TEC original do sequenciamento S, se sim, temos que uma atribuição de $verdade$ para a variável $viabilidade$, c.c., falso.

O cálculo do $novo\_tec$ é realizado retirando do TEC de S o custo do EPS_J_p e do EPS_I_p e somando o custo da ocupação de todas as unidades de tempo pertencentes ao EPS_I_p e do custo da quantidade de unidades de tempo ocupadas no EPS_I_p das tarifas de menor custo do EPS_J_p ($menor\_custo$) nas suas respectivas máquinas.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Algoritmo 2: `Busca_local`$(J, H, M, S, \hat{P})$

```
1  Criar os blocos EPS-I e EPS-J ∀p ∈ P̂ e armazenar em blocos_epsi e blocos_epsj, respectivamente;
2  while melhoria = verdade do
3      melhoria ← falso;
4      for p ∈ P̂ do
5          for EPS_j_p ∈ blocos_epsj do
6              for EPS_i_p ∈ blocos_epsi do
7                  viabilidade, novo_tec, menor_custo ← Viabilidade(EPS_j_p, EPS_i_p);
8                  if viabilidade = verdade then
9                      θ ← Movimento(novo_tec, menor_custo);
10                     if θ < TEC(S) then
11                         Atualizar o Sequenciamento S;
12                         Atualizar os blocos EPS-I e EPS-J ∀ p ∈ P̂ e armazenar em blocos_epsi e
                             blocos_epsj, respectivamente;
13                         melhoria ← verdade;
14                         Retornar para a linha 5;
15                     end
16                 end
17             end
18         end
19     end
20 end
21 return S
```

Se o $novo\_tec$ for menor que o TEC de S, i.e., $viabilidade = verdade$ então realizamos o movimento entre ambos. A função `Movimento` (Linha 9) realiza a troca e o rearranjo das tarefas de EPS_I_p nos períodos e máquina de EPS_J_p e o cálculo do novo TEC proveniente dessa movimentação, θ, é definido pela redução de $menor\_custo$ de $novo\_tec$ e pela adição do real custo de ocupação das tarefas de EPS_I_p nos períodos de EPS_J_p.

Em seguida, se θ for menor que o TEC de S (Linha 10), o novo sequenciamento e seus respectivos valores de TEC e $makespan$ são atualizados (Linha 11), e $blocos\_epsi$ e $blocos\_epsj$ são atualizados para todo $p \in \hat{P}$ (Linha 12). Após isso, o procedimento retorna para a Linha 5. Assim, a heurística de refinamento adota uma política de "primeira melhoria", repetindo esse processo até que não seja mais possível realizar nenhuma troca de EPS que melhore a solução.

Para resolver o problema de forma bi-objetiva e obter a fronteira aproximada de Pareto, utiliza-se o método de Restrição-ϵ aproximado. Este método envolve iterar através de diferentes horizontes de tempo $\hat{K}$, que variam de K^{max} até K^{min}. O valor de K^{max} representa o maior período do horizonte de tempo, enquanto $K^{min} = \max\{\lfloor \sum_{j \in J} p_j / M \rfloor, \max_{j \in J}\{p_j\}\}$. Para cada horizonte de tempo $\hat{K}$, aplica-se a heurística SGH para construir um sequenciamento viável de N tarefas em M máquinas. Em seguida, utiliza-se a heurística de refinamento ES para encontrar um ótimo local, e a solução resultante é adicionada à fronteira aproximada de Pareto F'. A cada iteração, $\hat{K}$ é reduzido em uma unidade, repetindo o procedimento até que o horizonte de tempo se torne tão pequeno que não seja mais possível encontrar um sequenciamento viável. Por fim, reporta-se F' sem incluir as soluções dominadas.

3.1. Abordagens propostas

A meta-heurística, denominada de Busca Local de Pareto de Duas fases (2PPLS), é proposta por Lust e Teghem [2010] e utiliza algoritmos aproximativos nas duas fases do método. Estas fases são: (1) Buscar um conjunto aproximado de boas soluções eficientes suportadas, empregando um algoritmo aproximativo com um único objetivo (mono-objetivo), resultante da agregação li-

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

near dos múltiplos objetivos do POM inicial; (2) Buscar um conjunto de soluções eficientes não-suportadas aproximadas, obtidas pela aplicação de um procedimento de busca em vizinhança, a partir da fronteira gerada na primeira fase.

No contexto deste trabalho, nos inspiramos nessa meta-heurística aplicando a proposta de Anghinolfi et al. [2021] na primeira fase, com algumas estratégias para melhorar as soluções, e aplicamos na segunda fase uma busca em vizinhança adaptada ao problema de modo a explorar estrategicamente mais soluções no espaço de soluções viáveis. Uma dessas estratégias mencionadas envolve a definição de EPS-K.

Definição 3.10 (Sequência de Períodos Trocáveis - K (EPS-K)) *Um EPS-K é um EPS que contém uma ou mais tarefas, sem nenhuma unidade de tempo ociosa.*

No algoritmo de busca local (1ª fase do 2PPLS) e na busca em vizinhança (2ª fase) considera-se movimentações entre EPS-K e EPS-I. Um subconjunto E é identificado como um EPS-K que pode conter uma ou mais tarefas se o primeiro elemento do subconjunto E é o tempo de início de uma tarefa $j \in J$, se o último elemento de E é o tempo de conclusão de uma tarefa $j' \in J$, e se todas as unidades de tempo do subconjunto E são ocupadas, ou seja, são utilizadas para processar tarefas. A consideração dessa estratégia permite uma maior possibilidade de trocas entre blocos, permitindo explorar mais o espaço de soluções.

Outra estratégia aplicada consiste na troca da política de refinamento, de "primeira melhoria" para "melhor melhoria". Nesse caso, todas as possíveis trocas de EPS-I e EPS-K para cada tempo de processamento distinto $p \in \hat{P}$, são testadas, e o movimento escolhido é aquele que resulta na melhor melhoria. Esse processo é repetido até alcançar um ótimo local.

Por fim, na segunda fase do 2PPLS, a busca por vizinhos x' de uma solução x é conduzida por meio do refinamento que envolve movimentos entre EPS's no sequenciamento de x. No entanto, adaptamos essa busca para permitir trocas que resultam em um aumento no valor de $C_{max}(x)$ em θ unidades, desde que haja uma melhoria no valor de TEC. Esse processo segue uma política de "primeira melhoria", onde os movimentos são realizados até que não seja possível encontrar um TEC menor sem piorar o C_{max} de x', indicando a convergência para um ótimo local dentro dessa estrutura de vizinhança.

A busca na vizinhança considera incrementos graduais de piora (de 1 unidade) no valor de $C_{max}(x)$, começando de 0 (para refinar a solução corrente) e avançando até a diferença entre o menor C_{max} de uma solução q da fronteira aproximada F_e que seja maior que o C_{max} da solução corrente x, e o próprio C_{max} da solução corrente x, i.e., $max\_piora = min\{C_{max}(q) : q \in F_e, C_{max}(q) > C_{max}(x)\} - C_{max}(x)$. Portanto, novas soluções vizinhas x' podem ser exploradas por meio de movimentos de blocos no sequenciamento da solução x que permitem uma piora do $C_{max}(x)$ em $\theta \in \{0, 1, \ldots, max\_piora - 1, max\_piora\}$ desde que $TEC(x') < TEC(x)$. Ainda, caso uma solução vizinha x' de ótimo local encontrada for dominada por alguma outra solução da fronteira aproximada de Pareto F_e a busca dos vizinhos da solução corrente x para e a próxima solução de F' passa a ser a corrente.

4. Resultados computacionais

As instâncias de teste propostas por Wang et al. [2018] são classificadas em pequena e grande escala. As instâncias pequenas possuem entre 6 e 25 tarefas, 3 a 7 máquinas paralelas, e 50 a 80 intervalos de tempo disponíveis por máquina. As instâncias grandes, por sua vez, incluem de 30 a 200 tarefas, 8 a 25 máquinas paralelas, e 100 a 300 intervalos de tempo. Em ambas as escalas, são considerados os tempos de processamento das tarefas, os custos de consumo energético das máquinas e as tarifas associadas aos intervalos de tempo disponíveis.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

Nas primeiras analisamos os indicadores de performance para a fronteira obtida por um algoritmo exato baseado em restrição-ϵ (coluna "MIP"), para a Abordagem 1, proposta por Anghinolfi et al. [2021] e a Abordagem 2, proposta nesse trabalho, e na segunda analisamos apenas os dois últimos. A comparação entre as abordagens foi analisada por meio de testes estatísticos de Wilcoxon. Um valor da estatistica de teste $p\text{-}valor$ menor que 0,05 indica uma diferença significativa entre as abordagens.

Tabela 1: Resultados dos indicadores de performance para Instâncias Pequenas e Grandes

Indicadores de Performance	Métricas	Instâncias Pequenas			Instâncias Grandes	
		MIP	Abord. 1	Abord. 2	Abord. 1	Abord. 2
Hipervolume	Média	0,7552	0,7534	0,7541	0,8314	0,8325
	Mediana	0,7650	0,7585	0,7614	0,8487	0,8508
	GAP	-	0,2371%	0,1523%	-	-0,1321%
	$p\text{-}valor$	-	0,0008		6,56E-07	
Pureza	Média	1,0000	0,8384	0,8902	0,5569	0,7265
	Mediana	1,0000	0,9034	0,9296	0,4844	0,7413
	GAP	-	16,157%	10,978%	-	-30,451%
	$p\text{-}valor$	-	0,0017		1,77E-05	
D_r	Média	0,0000	0,0021	0,0015	0,0034	0,0015
	Mediana	0,0000	0,0007	0,0004	0,0025	0,0009
	GAP	-	-	-	-	-121,70%
	$p\text{-}valor$	-	0,0042		3,98E-06	
Tempo CPU		25,0351	0,3904	0,9659	90,8097	241,2196
$p\text{-}valor$		-	1,86E-09		1,86E-09	

Na Tabela 1, observamos que a Abordagem 2 supera significativamente a Abordagem 1 em todas as métricas de desempenho, tanto para instâncias pequenas quanto grandes. Nas instâncias pequenas, a Abordagem 2 se distancia da "MIP" em média em 0,15% e 10,98% para o *Hipervolume* e *Pureza*, respectivamente, enquanto a 1 se distancia mais, com 0,24% e 16,16%, nessa ordem. Nas instâncias grandes, essas diferenças são ainda mais notáveis, com a média da Abordagem 2 superando a Abordagem 1 em 0,13%, 30,45% e 121,22% para *Hipervolume*, *Pureza* e D_r, respectivamente. No entanto, em termos de tempo computacional, a Abordagem 2 requer 2,47 e 2,66 vezes mais tempo que a Abordagem 1 para instâncias pequenas e grandes, respectivamente. Apesar disso, as estratégias implementadas, como o uso de EPS-K, a política de "melhor melhoria" e a busca em vizinhança, são cruciais para aprimorar as fronteiras.

5. Conclusão

Neste estudo, apresentamos uma heurística para resolver um problema de sequenciamento de máquinas paralelas considerando questões ambientais em um contexto bi-objetivo. Nosso objetivo é minimizar simultaneamente o $makespan$, e o consumo total de energia. Avaliamos a eficácia da nossa abordagem em comparação com a proposta de Anghinolfi et al. [2021] através de indicadores de desempenho que medem a qualidade da fronteira de Pareto. Os resultados dos testes computacionais mostram que nossa heurística supera significativamente a proposta anterior em todas as medidas de desempenho, em instâncias de pequeno e grande porte.

Nosso trabalho promete contribuir na redução dos impactos ambientais e na competitividade das indústrias, além de contribuir para o avanço da comunidade acadêmica. Visto isso, dada a crescente relevância do agendamento de eficiência energética, futuras pesquisas explorarão diversas estratégias nesse sentido. Além disso, planejamos aprimorar nossa proposta desenvolvendo uma busca em vizinhança mais eficiente e reduzindo o tempo computacional do algoritmo.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br

6. Agradecimento

O presente trabalho foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), 404807/2021-6; e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Código de Financiamento 001.

Referências

Afzalirad, M. e Shafipour, M. (2018). Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. *Journal of Intelligent Manufacturing*, 29(2):423–437.

Anghinolfi, D., Paolucci, M., e Ronco, R. (2021). A bi-objective heuristic approach for green identical parallel machine scheduling. *European journal of operational research*, 289(2):416–434.

Antoniadis, A., Garg, N., Kumar, G., e Kumar, N. (2020). Parallel machine scheduling to minimize energy consumption. In *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms*, p. 2758–2769. SIAM.

Bérubé, J.-F., Gendreau, M., e Potvin, J.-Y. (2009). An exact formula not shown-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194(1):39–50.

Che, A., Zhang, S., e Wu, X. (2017). Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697. ISSN 0959-6526. URL https://www.sciencedirect.com/science/article/pii/S0959652617307175.

Lust, T. e Teghem, J. (2010). Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 5 edition.

Wang, L. e Qi, Y. (2023). Scheduling an energy-aware parallel machine system with deteriorating and learning effects considering multiple optimization objectives and stochastic processing time. *CMES-Computer Modeling in Engineering & Sciences*, 135(1):325–339.

Wang, S., Wang, X., Yu, J., Ma, S., e Liu, M. (2018). Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of cleaner production*, 193:424–440.

Wu, X., Guo, P., Wang, Y., e Wang, Y. (2022). Decomposition approaches for parallel machine scheduling of step-deteriorating jobs to minimize total tardiness and energy consumption. *Complex & Intelligent Systems*, p. 1–16.

Wu, X. e Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega*, 82:155–165.

Xue, Y., Rui, Z., Yu, X., Sang, X., e Liu, W. (2019). Estimation of distribution evolution memetic algorithm for the unrelated parallel-machine green scheduling problem. *Memetic Computing*, 11:423–437.

Yin, Y., Wang, Y., Cheng, T., Liu, W., e Li, J. (2017). Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. *Omega*, 69:17–28.

DOI: 10.59254/sbpo-2024-193733
https://proceedings.science/p/193733?lang=pt-br