57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

UMA ABORDAGEM HÍBRIDA BASEADA EM META-HEURÍSTICA E *MACHINE LEARNING* PARA MELHORIA DA EFICIÊNCIA COMPUTACIONAL EM PROBLEMAS DE *JOB SHOP SCHEDULING*

Pablo Ramom Matias de Andrade
Universidade Federal da Paraíba (UFPB)
Campus I Lot. Cidade Universitária, João Pessoa, PB, 58051-900
andrade.prm@gmail.com

Lucídio dos Anjos Formiga Cabral
Universidade Federal da Paraíba (UFPB)
Campus I Lot. Cidade Universitária, João Pessoa, PB, 58051-900
lucidio@ci.ufpb.br

RESUMO

Avanços recentes mostram o sucesso de meta-heurísticas na resolução de problemas de agendamento *job shop*. Métodos baseados em aprendizado de máquina surgem como catalisadores das abordagens tradicionais. Assim, investigar a combinação dessas técnicas tornou-se relevante, visando melhorar a qualidade das soluções e a eficiência computacional. Este artigo propõe um algoritmo híbrido chamado *K-means Particle Swarm Optimization* (KPSO) para otimizar o *makespan* no problema clássico de *job shop* flexível. Inicialmente, utiliza-se o mapa caótico Singer para gerar soluções iniciais diversas e bem distribuídas. Logo, o *K-means* agrupa essas soluções e utiliza os centróides como ponto de partida para o PSO, com o objetivo de acelerar a convergência e melhorar os resultados. Testes em dez instâncias da literatura indicam que, em seis delas, o makespan foi reduzido e, em 70% dos casos, o makespan médio foi melhorado. Conclui-se que o uso da clusterização agrega valor e sugere consistência no algoritmo proposto.

PALAVRAS CHAVE. Aprendizado de Máquina, PSO, *Job Shop Scheduling*.

Tópicos: OC - Otimização Combinatória, MH - Meta-heurísticas.

ABSTRACT

Recent advances show the success of metaheuristics in solving job shop scheduling problems. Methods based on machine learning have emerged as catalysts for traditional approaches. Thus, investigating the combination of these techniques has become relevant in order to improve the quality of solutions and computational efficiency. This article proposes a hybrid algorithm called K-means Particle Swarm Optimization (KPSO) to optimize makespan in the classic flexible job shop problem. Initially, the Singer chaotic map is used to generate diverse and well-distributed initial solutions. K-means then groups these solutions together and uses the centroids as a starting point for PSO, with the aim of speeding up convergence and improving results. Tests on ten instances from the literature show that, in six of them, the makespan was reduced and, in 70% of cases, the average makespan was improved. The conclusion is that the use of clustering adds value and suggests consistency in the proposed algorithm.

KEYWORDS. Machine Learning. PSO. Job Shop Scheduling.

Paper topics: OC - Combinatorial Optimization, MH - Metaheuristics.

https://proceedings.science/p/212226?lang=pt-br

1. Introdução

Um dos problemas enfrentados pelas empresas industriais é o agendamento de tarefas, também conhecido como *schedule* ou *scheduling* [Baker e Trietsch, 2019]. Conforme define Pinedo [2022], o agendamento de tarefas é o processo de tomada de decisão que trata da alocação de tarefas (operações em um processo de produção) em recursos (máquinas) em determinados períodos de tempo e tem como objetivo otimizar um ou mais objetivos, adaptando-os às necessidades específicas de cada organização e setor.

O agendamento ou programação de tarefas faz parte essencial do planejamento de empresas modernas, visto que é através dele que pode-se melhorar a produtividade e utilização eficiente dos recursos empresariais [Xie et al., 2021]. Devido a tal importância, a programação de tarefas, tem atraído muito interesse por parte dos pesquisadores.

Além disso, os Problemas de *Job Shop Scheduling* (PJSS) são considerados os mais clássicos problemas de *schedule* e têm diversas aplicações no mundo real, como, por exemplo, em contexto de saúde [Pham e Klinkert, 2008], manufatura [Tamssaouet et al., 2022], transporte [Lamorgese e Mannino, 2019], etc. Devido à sua vasta aplicação, diversos métodos de resolução têm sido desenvolvidos para solucionar esses problemas de maneira mais eficiente.

Dentre essas abordagens, as meta-heurísticas (MH) têm sido extensivamente utilizadas pelo seu poder computacional, entregando soluções subótimas mais rápidas do que os métodos tradicionais [Ngoo et al., 2024]. Sendo assim, essa abordagem de resolução permite uma maior flexibilidade na modelagem desses problemas e é capaz de lidar com incertezas e variabilidades existentes no ambiente produtivo.

Todavia, a eficácia das técnicas meta-heurísticas é totalmente dependente da adaptação dos parâmetros específicos dos PJSS, através da seleção das variáveis de decisão, das restrições e dos objetivos, dentre outros [Kommadath et al., 2022]. Dessa forma, alguns desafios têm sido encontrados em sua aplicaç ao no mundo real, devido aos problemas serem considerados descontínuos, não diferenciais, com muitos ótimos locais, além da convergência prematura de soluções, que podem limitar a qualidade da solução fornecida [Li et al., 2024].

Para mitigar essas limitações, abordagens híbridas que combinam algoritmos de otimização com técnicas de aprendizado de máquina têm sido objeto de pesquisa nos últimos anos [Hajariwala et al., 2025]. Essa atenção dada é oriunda da sua capacidade de agregar conhecimento de domínio durante o processo de busca, orientando as MH para a tomada de melhores decisões e tornando-as mais inteligentes, melhorando significativamente seu desempenho em termos de qualidade de solução, taxa de convergência e robustez [Da Costa Oliveira et al., 2023; Karimi-Mamaghan et al., 2022].

Apesar dos avanços recentes no uso de aprendizado de máquina para melhorar as meta-heurísticas, poucos estudos têm explorado as potenciais aplicações no ambiente *job shop* [Nasiri et al., 2019]. Em particular, não há uma análise sistemática sobre o comportamento dessas técnicas híbridas na otimização do agendamento de tarefas industriais. Assim, este trabalho tem como objetivo analisar como se comporta um modelo de aprendizado de máquina dentro de um modelo meta-heurístico de otimização para o PJSS, avaliando seu impacto em termos de qualidade da solução e tempo computacional.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 são apresentadas as características específicas do PJSS consideradas no artigo. Na Seção 3 são descritos a meta-heuística proposta e o algoritmo de hibridização para resolver o PJSS. Os experimentos e os resultados computacionais são descritos na Seção 4. Por fim, a Seção 5 apresenta as principais conclusões baseadas nos resultados experimentais obtidos e possíveis trabalhos futuros.

2. Definição do problema

Esse estudo será desenvolvido num ambiente *job shop scheduling* flexível (JSSF). O JSSF é uma extensão do modelo clássico *job shop*, introduzindo flexibilidade na seleção de máquinas e sequenciamento de operação, tornando-o mais representativo nos cenários de produção do mundo real [Jiang e Liu, 2024].

Essa flexibilidade está associada ao fato de que não apenas as operações devem ser sequenciadas nas máquinas, como nos problemas de *job shop* clássico, mas também as operações devem ser atribuídas às máquinas, permitindo, assim, que a carga de trabalho, ou seja, a soma dos tempos de processamento das operações atribuídas à máquinas, seja melhor equilibrada em comparação com os problemas clássicos [Dauzére-Pérès et al., 2024].

Embora a flexibilidade permita uma utilização mais eficiente dos recursos, ofereça o potencial de reduzir o *makespan* e melhorar o desempenho geral do sistema, também expande o espaço de busca para possíveis soluções, tornando a tarefa de encontrar um cronograma ideal ou quase ideal ainda mais computacionalmente desafiadora [Xu et al., 2025].

Para o artigo, o modelo matemático utilizado será o mesmo apresentado no trabalho de Chen et al. [2020]. Para formular o problema, alguns símbolos e variáveis necessárias são mostrados da seguinte forma:

n : Número total de trabalhos;
j : Número total de operações;
k : Número total de máquinas;
s_{ijk} : Tempo de início da o_{ij} na máquina m_k;
p_{ijk} : Tempo de processamento O_{ij} na máquina m_k;
x_{ijk} : Variável binária, se a O_{ij} é atribuída na máquina m_k, $x_{ijk} = 1$; caso contrário, $x_{ijk} = 0$;

O modelo matemático é apresentado:

$$\text{Min. } C_{\max} = \text{Min} \left\{ \text{Max.} \sum_{i=1}^{n} \sum_{j=1}^{h} (s_{ij,k} + p_{ij,k}) \right\} \tag{1}$$

Sujeito a:

$$p_{ij,k} > 0, \quad i = 1, 2, \ldots, n; \; j = 1, 2, \ldots, h; \; k = 1, 2, \ldots, m. \tag{2}$$

$$s_{ij,k} + p_{ij,k} \leq s_{ij+1,k}, \quad i = 1, 2, \ldots, n; \; j = 1, 2, \ldots, h; \; k = 1, 2, \ldots, m. \tag{3}$$

$$\sum_{k=1}^{m} X_{i,j,k} \geq 1, \quad i = 1, 2, \ldots, n; \; j = 1, 2, \ldots, h. \tag{4}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{h} X_{i,j,k} = 1, \quad k = 1, 2, \ldots, m. \tag{5}$$

$$X_{i,j,k} = \begin{cases} 1, & \text{se a operação } O_{i,j} \text{ é alocada na máquina } k, \\ 0, & \text{caso contrário.} \end{cases} \tag{6}$$

A equação 1 representa a função objetivo do problema. Para este modelo, a escolha foi pela minimização do *makespan* (tempo total necessário para a conclusão de todas as operações). A equação 2 representa que o tempo de processamento de todas as operações é maior que zero. A equação 3 indica a ordem de precedência entre operações que devem ser executadas. A equação 4 garante que cada operação O_{ij} tem pelo menos uma máquina capaz de processá-la. A equação

5 estabelece que cada máquina deve processar exatamente uma operação por vez. E a equação 6 define a variável binária.

3. Meta-heurística híbrida proposta para resolução de problemas *job shop* flexível

Conforme afirma Abdolrazzagh-Nezhad e Abdullah [2024] para implementar as MH no ambiente *job shop* flexível, faz-se necessário seguir três etapas importantes, sendo: etapa de pré-processamento, etapa de inicialização e etapa de melhoria.

A fase de pré-processamento é constituída de duas tarefas principais, sendo a primeira a representação ou codificação do espaço de soluções (*Encoding*); e a segunda como o projeto de um algoritmo decodificador para gerar o cronograma ativo correspondente ao ponto codificado (*Decoding*). [Abdolrazzagh-Nezhad e Abdullah, 2024]. Para o artigo, será utilizado a codificação e decodificação apresentada no trabalho de Xu et al. [2025] que introduz esquemas integrados de codificação de números reais baseados em mapas caóticos para superar os desafios associados aos esquemas de codificação tradicionais que incluem altas despesas computacionais, dependências intercamadas e sensibilidade aos parâmetros de codificação, assim como a geração de cronogramas ativos a partir da sequência codificada.

A fase de incialização é a etapa onde se constrói a população ou solução inicial do problema. É importante destacar que esta etapa tem uma influência significativa na velocidade de convergência e na qualidade da solução dos problemas [Cui et al., 2024]. Por conta disso, foi definido que a hibridização da técnica de aprendizado de máquina irá acontecer nesta etapa. Alguns trabalhos na literatura aplicaram técnicas de *machine learning* nesta fase, como pode ser visto em Li et al. [2019]; Cheng et al. [2021]; Nguyen et al. [2021] e obtiveram bons resultados na aplicação.

Devido a vasta quantidade de técnicas de aprendizado de máquina existentes na literatura, para este artigo foi proposto o método *k-means* para identificar grupos de soluções promissoras no espaço de busca definido pela codificação caótica e, então, usar os centroides desses grupos como suas soluções iniciais. Ao usar os centroides como a população inicial, você está direcionando a busca inicial para regiões que já mostraram conter um número razoável de soluções candidatas.

Essa abordagem foi escolhida devido à sua simplicidade, ao tempo computacional relativamente curto e à facilidade de adaptação a novas áreas de aplicação, além de ser um dos mais importantes algoritmos de mineração de dados [Cheng et al., 2021].Além disso, conforme afirmam Tao et al. [2024] faltam pesquisas sobre a utilização dos centros de agrupamento em problemas de *job shop*.

A fase de melhoria é a parte onde os algoritmos de otimização são utilizados. Para o estudo em questão será utilizado o algoritmo *Particle Swarm Optimization* (PSO). A simplicidade, a facilidade de implementação e a capacidade de lidar com problemas de otimização complexos e multidimensionais fazem do PSO uma escolha popular tanto na pesquisa acadêmica quanto em aplicações pr'aticas [Fontes et al., 2023]. Além de ser flexível para hibridizar com outros algoritmos de otimização [Shami et al., 2022].

Inspirado pelo comportamento social observado na natureza, particularmente em enxames de aves e cardumes de peixes, o PSO é inicializado com uma população de partículas, sendo que cada uma das partículas representa uma solução candidata para o problema e tem três atributos principais: a posição no espaço de busca $x_i(t)$, a velocidade atual $v_i(t)$ e a melhor posição já encontrada pela partícula durante o processo de busca $Pbest_i(t)$. [Kennedy e Eberhart, 1995].

Conforme Nouiri et al. [2018] afirmam que o princípio do algoritmo é mover partículas para encontrar a solução ideal, regulando a trajetória delas de acordo com as experiências da própria partícula e das suas vizinhas. Durante a busca, cada partícula atualiza sua velocidade e posição de acordo com as seguintes equações:

$$v_i(t+1) = w * v_i(t) + c_1 r_1 * (\mathit{Pbest}_i(t) - x_i(t)) + c_2 r_2 * (\mathit{Gbest}_i(t) - x_i(t)), \quad (7)$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (8)$$

onde $\mathit{Pbest}_i(t)$ é a melhor posição de cada partícula, que representa o melhor valor particular da função objetivo obtida até o momento; o $\mathit{Gbest}_i(t)$ é a melhor partícula global, que denota a melhor posição entre todas as partículas da população; w é o peso de inércia que é usado para manter a partícula; c_1 e c_2 são o coeficientes de aceleração cognitiva e social, e r_1 e r_2 são dois valores aleatórios uniformes gerados dentro de um intervalo $[0, 1]$.

Dessa forma, a meta-heurística proposta denominada KPSO para o artigo seguirá o pseudo códido apresentado na Algoritmo 1.

Algorithm 1 Algoritmo 1: KPSO

Entrada: N; $T_{\max}$
Saída: Melhor cronograma (menor makespan)

1. $t \leftarrow 0$
2. Gerar população inicial com codificação caótica
3. Clusterizar partículas via *K-means*
4. Calcular centróides dos clusters
5. **Para cada** partículas na população:
   5.1 Decodificar posição
   5.2 Calcular makespan
6. Inicializar:
   - *pbest* de cada partículas ← sua posição atual
   - *gbest* ← melhor posição global
7. **Enquanto** $t \leq T_{\max}$:
   7.1 **Para cada** partículas:
      7.1.1 Atualizar velocidade (Equação 7)
      7.1.2 Atualizar posição (Equação 8)
   7.2 Decodificar novas posições
   7.3 Calcular makespan para todas as partículas
   7.4 Atualizar *pbest* e *gbest*:
      Se $\text{makespan}_{\text{atual}} < \text{makespan}_{\mathit{pbest}}$
         pbest ← posição atual
      Se $\text{makespan}_{\text{atual}} < \text{makespan}_{\mathit{gbest}}$
         gbest ← posição atual
   7.5 $t \leftarrow t + 1$
8. **Retorna** *gbest*

4. Resultados e Discussões

Aqui são apresentados os resultados alcançados nos experimentos, considerando o *makespan*, tempo de execução e a interação de convergência encontrada para cada instância.

4.1. Definição das instâncias

As instância que serão utilizadas neste trabalho são definidas e apresentadas através do trabalho de Brandimarte [1993]. São instância conhecidas na literatura e validadas em muitos tra-

balhos no contexto de *job shop scheduling* flexível. A Tabela 1 mostra a quantidade de trabalhos (j_i) de cada instância e a quantidade de máquinas (m) de cada trabalho.

Tabela 1: Tabela dos dados MK01 a MK10

MK01	MK02	MK03	MK04	MK05	MK06	MK07	MK08	MK09	MK10
10×6	10×6	15×8	15×8	15×4	10×15	20×5	20×10	20×10	20×15

O conjunto de dados Brandimarte consiste em instâncias de médio e grande porte, projetadas para testar a escalabilidade de um algoritmo e a capacidade de manter o desempenho à medida que a complexidade dos trabalhos e das máquinas aumenta [Xu et al., 2025].

4.2. Definição dos parâmetros

Para o mapa caótico foi utilizado o mapa Singer [Peitgen et al., 2004]. O parâmetro μ que consta neste mapa caótico teve valor estático assumido, sendo o mesmo apresentado no trabalho de Xu et al. [2025], que define o $\mu = 1.07$.

Como já foi visto, o PSO tem três parâmetros de controle: peso de inércia w, o componente cognitivo c_1 e o componente social c_2. Para os resultados, o peso de inércia foi considerado estático, com valor de $w = 0.7$ e os coeficientes c_1 e c_2 serão considerados como 2.1, conforme definido em Ding e Gu [2020].

O tamanho da população N selecionada depende do problema, e o trabalho de Xiaohui Hu et al. [2004] afirma que de 20 a 50 partículas são os tamanhos mais comuns nos problemas relacionados a PSO. Dessa forma, os testes foram realizados com N = 40 partículas no PSO. Além disso, o critério de parada foi definido pelo número máximo de interações, que para os testes foi de $T_{max} = 200$ interações.

Para o número de clusters, foram feitos alguns testes iniciais e foi verificado que o $K = 5$ retorna valores melhores. Além disso, a etapa de clusterização depende das características atribuídas a cada cluster. Dessa forma foram feitos dois testes. Para o primeiro teste denominado de KPSO (Tempo), foi feita a clusterização com base na máquina preferida, ou seja, para cada operação o_{ij} criou-se um vetor característica dos tempos de processamento de cada máquina e os agrupou com perfis similares. Para cada cluster, calculou-se o tempo médio de processamento em cada máquina e selecionou-se a máquina com o menor tempo (excluindo máquinas indisponíveis). Dessa forma, a inicialização é ajustada para favorecer a máquina preferida do cluster correspondente.

O segundo teste, denomiando de KPSO (Tempo + Carga), avaliou o comportamento do algoritmo com características adicionais. Foi escolhida a carga de trabalho da máquina para clusterização, ou seja, foi acrescido ao vetor característica a soma dos tempos de processamento de todas as operações que podem ser executadas nas máquinas, intercalando com os tempos de processamento. A combinação das duas características seguiu uma ponderada, representando 70% para os tempos de processamento e 30% para a carga, na escolha da máquina preferida do cluster.

4.3. Resultados dos testes

O algoritmo apresentado foi implementado inicialmente na versão Python 3.12.4 em um processador Intel Core i5, 12ª geração, executado a 1.30 GHz, com o sistema operacional Windows 11 e 16GB de memória RAM. Para os testes, cada instância foi executada de forma independente 30 vezes, com as soluções resultantes registradas para comparação. A Tabela 2 mostra os melhores *makespans* encontrados para cada instância nos dois teste.

Fazendo uma comparação dos makespans encontrados, o algoritmo retornou melhores resultados quando foi acrescentada a característica adicional, sendo superior nas instâncias MK03,

Tabela 2: Resultados do algoritmo KPSO para as instâncias MK01 a MK10

Teste	**MK01**	**MK02**	**MK03**	**MK04**	**MK05**	**MK06**	**MK07**	**MK08**	**MK09**	**MK10**
Tempo	**48**	**32**	232	102	178	83	**154**	577	**406**	306
Tempo + Carga	**48**	**32**	**224**	**98**	**177**	**78**	155	**566**	410	**279**

MK04, MK05, MK06, MK08 e MK10, permanecendo com o mesmo valor em MK01 e MK02. Essa melhoria representa um fator de 75% no encontro de melhores soluções pelo algoritmo proposto.

Para comparar o impacto da hibridização da proposta, foi feito um teste do algoritmo, retirando a técnica *K-means*. A Tabela 3 mostra os melhores makespans encontrados.

Tabela 3: Dados do algoritmo PSO sem K-means para as instâncias MK01 a MK10

MK01	**MK02**	**MK03**	**MK04**	**MK05**	**MK06**	**MK07**	**MK08**	**MK09**	**MK10**
49	33	231	102	177	80	154	566	401	297

Percebe-se que o algoritmo KPSO retornou melhores resultados de *makespan* em 60% das instâncias. Além disso retornou resultados idênticos nas instâncias MK05 e MK08, o que sugere uma boa adequação da técnica *k-means* dentro de problemas de *job shop scheduling*.

De forma a avaliar a consistência do algoritmo, foi coletado das instâncias de cada teste o *makespan* médio, o desvio padrão e o tempo médio de execução. A Tabela 4 mostra os resultados nos três testes realizados.

Tabela 4: Resultados comparativos dos algoritmos

	KPSO (Tempo)			**KPSO (Tempo + carga)**			**PSO Sem K-means**		
MK	*Makespan*	*Desvio*	*Tempo (s)*	*Makespan*	*Desvio*	*Tempo (s)*	*Makespan*	*Desvio*	*Tempo (s)*
MK1	53,3	**2,10**	2,13	**50,4**	2,28	2,11	52,3	2,79	**1,94**
MK2	34,6	2,15	2,64	**33,9**	**1,30**	2,53	35,4	1,56	**2,40**
MK3	253,0	8,29	7,03	**236,7**	**6,59**	6,57	256,1	12,84	**6,31**
MK4	117,4	12,05	3,75	**101,0**	**3,00**	**3,63**	118,0	8,00	3,64
MK5	185,2	3,76	4,15	**180,0**	**2,02**	4,12	185,6	4,82	**4,02**
MK6	89,0	3,28	7,33	**84,9**	3,60	7,27	85,2	**2,41**	**7,07**
MK7	164,7	6,13	4,10	171,0	6,82	4,21	**163,0**	**4,51**	**4,00**
MK8	597,0	10,84	10,62	584,0	10,49	**9,74**	**581,3**	**6,93**	10,63
MK9	429,9	9,62	12,32	427,5	**8,07**	12,36	**419,1**	9,44	**11,66**
MK10	336,2	13,78	12,83	**303,0**	**9,67**	12,96	317,2	14,62	**12,19**

Os resultados mostram que, na maioria das instâncias, o algoritmo KPSO (Tempo + Carga) é mais consistente nos dados apresentados (desvio padrão menor), sugerindo um algoritmo

mais confiável, ou seja, um algoritmo baseado nos centróides para inicialização reduz positivamente os impactos negativos de uma inicialização aleatória, direcionando melhor a meta-heurística PSO a explorar o espaço de busca mais relevante. Além disso, pode-se perceber que o *makespan* médio é menor em 70% das instâncias utilizadas, sugerindo que a clusterização agrega valor ao estudo. Esses resultados corroboram com os resultados encontrados em Van Der Merwe e Engelbrecht [2003] que mostraram uma melhoria geral no desempenho quando o PSO é inicializado dessa forma.

5. Conclusões e Trabalhos Futuros

Os resultados apresentados sugeriram consistência em *makespan* médio e desvio padrão para o teste realizado no algoritmo utilizando tempo de processamento e carga da máquina, além de apresentar em sua grande maioria as melhores soluções. No entanto, foram verificados desafios em instâncias de grande porte como o MK06 a MK09, destacando a necessidade de mais estudos acerca dos ajustes dos parâmetros do algoritmo.

Em resumo, os resultados indicam que o algoritmo desenvolvido, utilizando *k-means* na inicialização da meta-heurística PSO sugere robustez para a melhoria das soluções e convergência do método. A contribuição numérica destes resultados, aliada à aplicação prática desta abordagem no ambiente *job shop*, destaca o potencial desta pesquisa para avanços significativos na área de otimização combinatória.

Para trabalhos futuros sugere a combinação de outras características para clusterização como prioridade dos trabalhos, flexibilidade de operações, criticidade da operação, tempo de início mais cedo, dentre outras. Uma outra linha de investigação que pode ser iniciada é a hibridização com outras técnicas de *machine learning*. Além disso, está em andamento na pesquisa a utilização de novas estruturas de vizinhança.

6. Agradecimentos

Esta pesquisa foi realizada com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Demanda Social

Referências

Abdolrazzagh-Nezhad, M. e Abdullah, S. (2024). A Review on Metaheuristic Approaches for Job-Shop Scheduling Problems. *Data Science: Journal of Computing and Applied Informatics*, 8 (1):45–63. ISSN 2580-829X. URL https://talenta.usu.ac.id/JoCAI/article/view/17138. Number: 1.

Baker, K. R. e Trietsch, D. (2019). *Principles of sequencing and scheduling*. Wiley series in operations research and management science. Wiley, Hoboken, NJ, USA, second edition edition. ISBN 978-1-119-26259-6 978-1-119-26256-5.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183. ISSN 0254-5330, 1572-9338. URL http://link.springer.com/10.1007/BF02023073.

Chen, R., Yang, B., Li, S., e Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 149:106778. ISSN 03608352. URL https://linkinghub.elsevier.com/retrieve/pii/S0360835220304885.

Cheng, C.-Y., Pourhejazy, P., Ying, K.-C., e Lin, C.-F. (2021). Unsupervised Learning-based Artificial Bee Colony for minimizing non-value-adding operations. *Applied Soft Computing*, 105:

107280. ISSN 15684946. URL https://linkinghub.elsevier.com/retrieve/pii/S1568494621002039.

Cui, H., Li, X., Yu, Y., Zhou, W., Fu, L., e Xiang, Z. (2024). An effective metaheuristic with adaptive delay active decoding for the no-wait job-shop scheduling problem. *Engineering Optimization*, p. 1–21. ISSN 0305-215X, 1029-0273. URL https://www.tandfonline.com/doi/full/10.1080/0305215X.2024.2431849.

Da Costa Oliveira, A. L., Britto, A., e GusmÃ£o, R. (2023). Machine learning enhancing metaheuristics: a systematic review. *Soft Computing*, 27(21):15971–15998. ISSN 1432-7643, 1433-7479. URL https://link.springer.com/10.1007/s00500-023-08886-3.

Dauzére-Pérès, S., Ding, J., Shen, L., e Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2):409–432. ISSN 03772217. URL https://linkinghub.elsevier.com/retrieve/pii/S037722172300382X.

Ding, H. e Gu, X. (2020). Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. *Computers & Operations Research*, 121:104951. ISSN 03050548. URL https://linkinghub.elsevier.com/retrieve/pii/S030505482030068X.

Fontes, D. B., Homayouni, S. M., e GonÃ§alves, J. F. (2023). A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *European Journal of Operational Research*, 306(3):1140–1157. ISSN 03772217. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221722007184.

Hajariwala, D. C., Patil, S. S., e Patil, S. M. (2025). A Review of Metaheuristic Algorithms for Job Shop Scheduling. *Engineering Access*, 11(1):65–91. ISSN 27304175.

Jiang, T. e Liu, L. (2024). A Bi-Population Competition Adaptive Interior Search Algorithm Based on Reinforcement Learning for Flexible Job Shop Scheduling Problem. *International Journal of Computational Intelligence and Applications*. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85206325664&doi=10.1142%2fS1469026824500251&partnerID=40&md5=1b72562ae28dcaec285fbc64850b86aa.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., e Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422. ISSN 03772217. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221721003623.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, p. 1942–1948, Perth, WA, Australia. IEEE. ISBN 978-0-7803-2768-9. URL http://ieeexplore.ieee.org/document/488968/.

Kommadath, R., Ramchandani, B., Nagraj, M. S., e Kotecha, P. (2022). Performance Evaluation of Recently Proposed Metaheuristics Algorithms on Solving Job Shop Scheduling Problem. In *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, p. 1051–1058. URL https://ieeexplore.ieee.org/document/10029066.

Lamorgese, L. e Mannino, C. (2019). A Noncompact Formulation for Job-Shop Scheduling Problems in Traffic Management. *Operations Research*, 67(6):1586–1609. ISSN 0030-364X, 1526-5463. URL https://pubsonline.informs.org/doi/10.1287/opre.2018.1837.

Li, J., Li, X., e Gao, L. (2024). An operator-inspired framework for metaheuristics and its applications on job-shop scheduling problems. *Applied Soft Computing*, 157:111522. ISSN 15684946. URL https://linkinghub.elsevier.com/retrieve/pii/S1568494624002965.

Li, Z., Qian, B., Hu, R., Chang, L., e Yang, J. (2019). An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups. *Knowledge-Based Systems*, 173:83–112. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85062651133&doi=10.1016%2fj.knosys.2019.02.027&partnerID=40&md5=22856c091d0771c043966fbe2352afea.

Nasiri, M. M., Salesi, S., Rahbari, A., Salmanzadeh Meydani, N., e Abdollai, M. (2019). A data mining approach for population-based methods to solve the JSSP. *Soft Computing*, 23(21):11107–11122. ISSN 1432-7643, 1433-7479. URL http://link.springer.com/10.1007/s00500-018-3663-2.

Ngoo, C. M., Goh, S. L., Sze, S. N., Sabar, N. R., Hijazi, M. H. A., e Kendall, G. (2024). A survey of mat-heuristics for combinatorial optimisation problems: Variants, trends and opportunities. *Applied Soft Computing*, 164:111947. ISSN 1568-4946. URL https://www.sciencedirect.com/science/article/pii/S156849462400721X.

Nguyen, S., Thiruvady, D., Zhang, M., e Tan, K. (2021). A Genetic Programming Approach for Evolving Variable Selectors in Constraint Programming. *IEEE Transactions on Evolutionary Computation*, 25(3):492–507. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099541702&doi=10.1109%2fTEVC.2021.3050465&partnerID=40&md5=6d25836734bfb56314635b557ae5f35f.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., e Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):603–615. ISSN 0956-5515, 1572-8145. URL http://link.springer.com/10.1007/s10845-015-1039-3.

Peitgen, H.-O., JÃ¼rgens, H., e Saupe, D. (2004). *Chaos and Fractals*. Springer New York, New York, NY. ISBN 978-1-4684-9396-2 978-0-387-21823-6. URL http://link.springer.com/10.1007/b97624.

Pham, D.-N. e Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 185(3):1011–1025. ISSN 0377-2217. URL https://www.sciencedirect.com/science/article/pii/S0377221706005820.

Pinedo, M. L. (2022). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, Cham. ISBN 978-3-031-05920-9 978-3-031-05921-6. URL https://link.springer.com/10.1007/978-3-031-05921-6.

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., e Mirjalili, S. (2022). Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, 10:10031–10061. ISSN 2169-3536. URL https://ieeexplore.ieee.org/document/9680690/.

Tamssaouet, K., DauzÃ¨re-PÃ©rÃ¨s, S., Knopp, S., Bitar, A., e Yugma, C. (2022). Multiobjective optimization for complex flexible job-shop scheduling problems. *European Journal of Operational Research*, 296(1):87–100. ISSN 0377-2217. URL https://www.sciencedirect.com/science/article/pii/S0377221721003751.

Tao, X.-R., Pan, Q.-K., Sang, H.-Y., e Rong, M. (2024). A multiobjective optimizer with a K-means cluster algorithm for a distributed flexible flowshop rescheduling problem. *Applied Soft Computing*, 166:112124. ISSN 15684946. URL https://linkinghub.elsevier.com/retrieve/pii/S1568494624008986.

Van Der Merwe, D. e Engelbrecht, A. (2003). Data clustering using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, p. 215–220, Canberra, ACT, Australia. IEEE. ISBN 978-0-7803-7804-9. URL http://ieeexplore.ieee.org/document/1299577/.

Xiaohui Hu, Yuhui Shi, e Eberhart, R. (2004). Recent advances in particle swarm. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, p. 90–97, Portland, OR, USA. IEEE. ISBN 978-0-7803-8515-3. URL http://ieeexplore.ieee.org/document/1330842/.

Xie, J., Li, X., Gao, L., e Gui, L. (2021). A new neighborhood structure for job shop scheduling problems. URL http://arxiv.org/abs/2109.02843. arXiv:2109.02843 [cs].

Xu, Y., Wang, D., Zhang, M., Yang, M., e Liang, C. (2025). Quantum particle swarm optimization with chaotic encoding schemes for flexible job-shop scheduling problem. *Swarm and Evolutionary Computation*, 93:101836. ISSN 22106502. URL https://linkinghub.elsevier.com/retrieve/pii/S2210650224003742.