ISSN 2596-2809

ANÁLISE COMPARATIVA DE ALGORITMOS APLICADOS AO FJSSP

COMPARATIVE ANALYSIS OF ALGORITHMS APPLIED TO FJSSP

Walter Schmidt Marinho[1]
Lupercio Fuganti Luppi[2]

RESUMO

Este artigo é uma republicação do trabalho de conclusão de curso apresentado ao Centro Universitário Filadélfia como requisito para obtenção de diploma de bacharel em Ciência da Computação. Esta investigação levantou os algoritmos aplicados a problemas FJSSP e suas principais características. Foram também desenvolvidos dois Algoritmos Genéticos (AG), um clássico e um híbrido com Q-learning para verificação das características levantadas e os resultados mostraram que houve uma melhora de 31,5% no tempo de execução do algoritmo usando a hibridização. O critério usado para avaliação da melhora foi o tempo total de produção, ou *makespan* e os algoritmos genéticos aplicados usaram operadores de seleção, cruzamento e mutação.

Palavras-chave: FJSSP; escalonamento; algoritmo genético; reinforcement learning.

ABSTRACT

This article is a republication of the final paper presented to Centro Universitário Filadélfia as a fulfilment requirement of the Bachelor in Computer Science program. This investigation analyzed some algorithms applied to the solution of FJSSP problems and their main characteristics. To validate the findings, two Genetic Algorithms (GAs) were developed, one using a classic GA approach and the second using a hybrid GA with Q-Learning. The results showed that by using the Q-Table to regulate the mutation coefficient on the genetic algorithm, the total execution time was improved by 31.5%. The criteria for evaluating the solutions was the total processing time, also known as *makespan*, and the genetic algorithms developed used selection, crossover and mutation operators.

Keywords: FJSSP; scheduling; genetic algorithm; reinforcement learning.

[1] Graduando do Curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil. Email: waltersmarinho@edu.unifil.br)
[2] Orientador: Professor Me. do curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil. Email: lupercio.luppi@unifil.br

ISSN 2596-2809

1 INTRODUÇÃO

O problema de escalonamento de linhas de produção é um dos problemas de otimização mais interessantes para a indústria, pois impacta diretamente a utilização de recursos e minimiza os custos ao mesmo tempo em que maximiza a capacidade produtiva da fábrica.

Para isso, métodos para escalonamento de produção foram desenvolvidos. Depois de anos de pesquisas, diversos algoritmos foram propostos, podendo ser classificados em diversas categorias. E para aplicar os algoritmos, foi necessário desenvolver uma modelagem computacional da linha de produção. Com isso, surgiu o problema conhecido como *Job Shop Scheduling Problem* (JSSP).

1.1 Job Shop Scheduling Problem - JSSP

A primeira modelagem de ambiente de produção da classe *job shop* foi esta. Neste problema, modelamos a linha de produção em um conjunto de *j* jobs, compostos por *n* operações, processados por *m* máquinas.

Além disso, algumas restrições são propostas:

- Cada operação de cada job pode ser processado em apenas uma máquina;
- Uma operação de um job só pode ser processada caso a sua operação anterior já tenha sido processada.
- Uma máquina não pode processar duas operações simultaneamente.
- Uma operação, uma vez iniciada, não pode ser interrompida.

Para avaliar soluções JSSP, a métrica mais comum é minimizar o tempo total de processamento dos jobs, chamado também de *makespan***.**

1.2 Flexible Job Shop Scheduling Problem - FJSSP

O FJSSP surgiu da necessidade de aproximar os sistemas de produção dos sistemas reais encontrados nas fábricas. Mais especificamente levar em consideração a possibilidade de existir mais de uma máquina capaz de processar a mesma

operação do job, com o mesmo tempo de processamento da operação ou não.

Com isso, há um aumento na complexidade do problema e dividimos sua solução em 2 etapas, a primeira determinamos quais máquinas irão processar cada operação, e em seguida resolvemos o melhor escalonamento da produção para obter uma solução.

1.3 Motivação do trabalho

Tanto o JSSP quanto o FJSSP são considerados problemas NP-hard, ou seja, com complexidade de solução com aumento não polinomial conforme aumentamos o tamanho do problema.

Como estes problemas representam uma situação muito desejada pela indústria, muitos algoritmos estão sendo desenvolvidos e melhorados por pesquisadores. No entanto, o estudo do FJSSP é um estudo recente, portanto este trabalho busca compilar as principais categorias de algoritmos aplicados para solução deste problema e suas características. No entanto, os algoritmos recentes acumulam uma grande complexidade, então a discussão deste trabalho também serve para que pesquisadores ingressantes nesta área possam ter um panorama geral das soluções possíveis.

Para verificar e testar algumas hipóteses levantadas na comparação dos algoritmos levantados durante a pesquisa, foram desenvolvidos dois algoritmos para uma instância simples do problema para comparação dos seus tempos de execução e resultados obtidos.

2 DESENVOLVIMENTO

2.1 Levantamento de algoritmos aplicados e suas características

Para consulta dos artigos, foram utilizadas três bases de dados, Springer, IEEE e ACM, garantindo uma boa qualidade dos artigos base para levantamento das informações. Nessas bases de dados foi usada uma string de busca padronizada, e

os artigos foram selecionados a partir de seus títulos, resumos e datas de publicação. Somente foram selecionados artigos publicados entre janeiro de 2019 e julho de 2021. No final desta seleção, sobraram 14 artigos selecionados.

Após a leitura detalhada de todos os artigos levantados, foi possível notar a presença de 3 categorias de algoritmos aplicados ao FJSSP. A tabela 1 mostra os algoritmos encontrados e sua frequência de aparição nos artigos.

Tabela 1 – Relação dos algoritmos encontrados nos artigos selecionados

Categoria	Número de referências nos artigos
Algoritmo genético	4
Algoritmos evolucionários	3
Aprendizado por reforço	9

Fonte: Marinho (2021)

A categoria de algoritmo mais proeminente nos artigos selecionados foi a de aprendizado por reforço, com 9 artigos, seguido do algoritmo genético com 4 ocorrências, e com os algoritmos evolucionários aparecendo 3 vezes. Muitos artigos selecionados investigaram mais de um algoritmo.

A seguir são discutidos os pontos positivos e negativos de cada categoria de algoritmos encontrados.

2.1.1 Algoritmo Genético

O algoritmo genético (AG) é o algoritmo mais investigado para aplicação na solução de escalonamento do FJSSP. Por isso, ele é muito usado como parâmetro de comparação com outros algoritmos propostos.

Ao aplicar algoritmos desta categoria no problema, eles foram capazes de obter as melhores soluções para o escalonamento da produção, e estão entre os melhores tipos de algoritmo, com soluções muito próximas do melhor conhecido, principalmente para grandes instâncias de problemas.

Outra vantagem dos AGs são suas etapas, de forma que este algoritmo pode ser melhorado através da aplicação de outros algoritmos (hibridização).

ISSN 2596-2809

No entanto, a solução é demorada e é preciso configurar todos os parâmetros para o problema ao qual estamos solucionando. Portanto, a solução final é extremamente dependente destes parâmetros e da solução inicial fornecida.

2.1.2 Algoritmos Evolucionários

Algoritmos evolucionários (AEs) possuem muitas características em comum com algoritmos genéticos devido a serem baseados em comportamentos da natureza. Dessa forma, AEs também encontram soluções boas para diversas instâncias do FJSSP e são fáceis de hibridizar com outros algoritmos. Por terem inspirações diversas, esta categoria de algoritmos possui uma maior adaptabilidade a diversas configurações de problema. Mas essa adaptabilidade faz necessário saber escolher qual o melhor algoritmo para a solução.

Um segundo problema na aplicação de AEs é a demora na obtenção de solução em alguns ambientes industriais, e, por fim, a regulagem dos parâmetros do algoritmo também ser trabalhosa e sensível, alterando o resultado final dependendo dos parâmetros e da solução inicial usados.

2.1.3 Aprendizado por reforço

Algoritmos de aprendizado por reforço (RL, do inglês R*einforcement Learning*), são capazes de aprender como resolver problemas enquanto buscam uma solução.

Algoritmos de RL foram capazes de apresentar soluções próximas das melhores soluções conhecidas pelas outras categorias de algoritmos (AGs e AEs). Sua principal vantagem está no tempo necessário para obter esta solução, que é menor em relação às outras categorias. Além disso, se aplicados de forma híbrida com outros algoritmos, o RL foi capaz de acelerar a solução.

A principal desvantagem está na necessidade de uma etapa de treinamento prévia do modelo, inviabilizando sua aplicação onde há muita variação no ambiente de produção.

ISSN 2596-2809

2.2 Algoritmos desenvolvidos

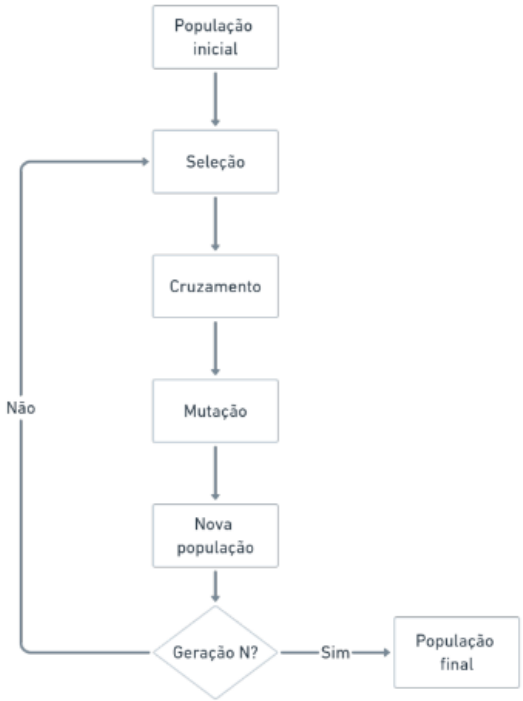
2.2.1 Genético clássico

No algoritmo genético clássico, foram usados operadores de seleção, cruzamento e mutação dos cromossomos.

Na seleção, copiamos alguns cromossomos da geração anterior para a próxima geração. O restante da população é gerada usando o cruzamento e alguns cromossomos são alterados usando mutação.

A figura 3 mostra as etapas do algoritmo genético.

Figura 3 - Etapas de um algoritmo genético clássico



Fonte: Marinho (2021).

O cromossomo é composto por uma sequência de operações e uma sequência de máquinas que irá processar cada operação, representadas pela parte cinza na figura 4 e pela parte branca na figura 4, respectivamente.

Figura 4 - Representação de um cromossomo

1	2	1	3	1	2	3	2
1	2	1	1	3	2	1	1

Fonte: Marinho (2021).

Para a seleção, foram selecionados randomicamente uma parcela dos cromossomos da população atual. O cruzamento dos cromossomos foi feito usando o cruzamento IPOX aplicado por Luo Quian e Fu (2020) que não gera soluções inválidas para o problema, pois respeita a ordem de processamento das operações.

A mutação é feita usando o operador de mutação *swap*, onde apenas trocamos duas posições de lugar no cromossomo e só serão sorteadas posições válidas para troca, a fim de se respeitar a validade do cromossomo.

A tabela 2 apresenta os parâmetros usados na execução do algoritmo genético clássico.

Tabela 2 – Parâmetros de execução do algoritmo genético clássico

Parâmetro	Valor
Número de gerações	500
Tamanho da população	100
Porcentagem de seleção	0,5
Probabilidade de mutação	0,3
Operador de cruzamento	IPOX

Fonte: Marinho (2021)

2.2.2 Genético híbrido

O algoritmo híbrido se diferencia do clássico pois usa uma tabela-Q baseada no número de gerações que se passaram para regular a mutação do algoritmo. Os valores da tabela para seleção da taxa de mutação são atualizados segundo a equação 2.1.

Equação 2.1 - Equação de atualização de valores na tabela Q

$$Q(S_t, a_t) \leftarrow (1-\alpha)Q(S,a) + \alpha(r_{t+1} + \gamma max Q(S_{t+1}, a_{t+1}))$$

Fonte: Chen *et al*. (2020)

ISSN 2596-2809

Onde Q(St, at) representa o valor de Q atual da tabela, α é a taxa de aprendizado do algoritmo, rt+1 é a recompensa de aplicar a ação neste estado, γ é a taxa de desconto e maxQ(St+1, at+1) representa o maior valor de Q do estado subsequente após a aplicação da ação.

A tabela 3 mostra os parâmetros usados na execução do algoritmo híbrido.

Tabela 3 – Parâmetros de execução do algoritmo genético híbrido

Parâmetro	Valor
Número de gerações	500
Tamanho da população	100
Porcentagem de seleção	0.5
Probabilidade de mutação	variável entre 0,1 e 0,5
Operador de cruzamento	IPOX

Fonte: Marinho (2021)

2.3 Resultados obtidos

A máquina usada para executar os algoritmos é equipada com processador intel i7-10750H e 16GB de RAM. Cada algoritmo foi executado 20 vezes. Os resultados estão mostrados na tabela 4.

Tabela 4 – Resultados de makespan e tempos de execução dos algoritmos

	Genético puro		Genético híbrido	
Iteração	Makespan	Tempo de execução [s]	Makespan	Tempo de execução [s]
1	53	7.562	49	5.450
2	56	8.133	49	5.333
3	57	8.387	54	5.473
4	50	7.755	58	5.446
5	47	7.752	53	5.533
6	52	7.789	54	5.324
7	51	7.729	56	5.300
8	55	7.555	52	5.187
9	55	7.534	53	5.423
10	55	7.585	58	5.369
11	54	8.262	55	5.166
12	56	7.516	51	5.299
13	56	7.655	50	5.489
14	49	7.542	56	5.250
15	53	7.815	51	5.173
16	51	7.868	53	5.460
17	58	7.787	48	5.250
18	62	8.195	58	5.008
19	55	7.698	53	5.360
20	56	7.459	54	5.357

Fonte: Marinho(2021)

ISSN 2596-2809

Podemos visualizar que o algoritmo híbrido foi mais rápido que o algoritmo puro (clássico) em todas as execuções, alinhando-se com a hipótese levantada anteriormente de aceleração de algoritmos fornecida por métodos de reinforcement learning.

Na tabela 5, podemos visualizar os melhores resultados de makespan dos algoritmos implementados e dos algoritmos propostos na literatura.

Tabela 5 – Melhores *makespans* obtidos pelos algoritmos e da literatura

Genético puro	Genético híbrido	Luo, Qian e Fu (2020)	Chen et al. (2020)
47	48	40	40

Fonte: Marinho (2021)

Na tabela 6, é mostrada uma comparação do tempo total de execução dos algoritmos, com uma diferença de 31,5% no tempo total de execução, denunciando que em sistemas de escalonamento recorrente, algoritmos com hibridização com reinforcement learning oferecem soluções boas e rápidas o suficiente para que a indústria obtenha resultados.

Tabela 6 – Comparação dos tempos de execução total dos algoritmos

Genético puro	Genético híbrido	Diferença porcentual
155.6	106.6	31,5%

Fonte: Marinho (2021)

Através das observações encontradas e pontuadas neste artigo, pode-se evidenciar a importância da investigação de algoritmos para solução deste problema, e mostrar que com demais investigações na área, novos métodos podem surgir que trarão resultados muito bons para a indústria.

3 CONCLUSÃO

Neste trabalho, foram investigadas as principais categorias de algoritmos

ISSN 2596-2809

aplicados para a obtenção de escalonamentos para sistemas de produção modelados como FJSSP.

Foram encontradas três categorias de algoritmos, sendo elas, algoritmos genéticos (AGs), algoritmos evolucionários (AEs) e algoritmos de reinforcement learning. Onde cada um possui seus pontos fortes e fracos quando aplicados. AGs e AEs encontram as melhores soluções, mas tem aplicação complexa devido a necessidade de ajuste de parâmetros para cada instância do problema. E por fim, algoritmos de reinforcement learning (RLs) são capazes de obter soluções viáveis por fornecerem soluções com boa qualidade em um tempo de execução menor. Além disso, o RL é capaz de acelerar a execução de outros algoritmos quando aplicado em conjunto.

Para verificar essas hipóteses, foram desenvolvidos dois algoritmos genéticos, um clássico com e outro hibridizado com Q-learning para regulagem da mutação do algoritmo. Com eles verificou-se que as hipóteses estavam corretas. O AG clássico obteve a melhor solução de *makespan* mas demorou mais tempo para resolver cada solução. O AG híbrido obteve soluções próximas às obtidas pelo AG clássico, mas em um tempo 31,5% menor.

Esta melhora é uma vantagem muito importante para as linhas de produção, pois com um tempo de escalonamento menor, a linha fica menos tempo ociosa e reduz desperdícios de energia e gastos com armazenamento de estoque e produção excedente.

Em futuras investigações, é interessante investigar a aplicação deste algoritmo desenvolvido a mais instâncias do problema FJSSP e investigar seu comportamento, investigar diferentes operadores de cruzamento, e aplicar uma forma de escolher o melhor operador para cada momento da produção. Outra possibilidade seria comparar AGs com componentes mais robustos, como elitismo e geração da população que não dependam de randomicidade.

Uma última recomendação seria aplicar um algoritmo de RL para escalonamento de forma isolada a fim de verificar o comportamento e os resultados obtidos por uma aplicação única de um algoritmo dessa categoria para um problema FJSSP.

REFERÊNCIAS

BAER, S. *et al*. Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems. In: **2019 Second International Conference on Artificial Intelligence for Industries (AI4I)**. [S.l.: s.n.], 2019. p. 22–25.

BOUAZZA, W.; SALLEZ, Y.; BELDJILALI, B. A distributed approach solving partially flexible job-shop scheduling problem with a q-learning effect. **IFAC-PapersOnLine**, v. 50, n. 1, p. 15890–15895, 2017. ISSN 2405-8963. 20th IFAC World Congress. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405896317331701.

BRANDIMARTE, P. Routing and scheduling in a flexible job shop by tabu search. **Annals of Operations Research**, v. 41, p. 157–183, 1993.

CHEN, R. *et al*. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. **Computers & Industrial Engineering,** v. 149, p. 106778, 2020. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/S0360835220304885.

COELHO, P. *et al*. Thirty years of flexible job-shop scheduling: A bibliometric study. **Procedia Computer Science,** v. 180, p. 787–796, 2021. ISSN 1877-0509. Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020). Disponível em: https://www.sciencedirect.com/science/article/pii/S187705092100380X.

FAHLE, S.; PRINZ, C.; KUHLENKöTTER, B. Systematic review on machine learning (ml) methods for manufacturing processes – identifying artificial intelligence (ai) methods for field application. **Procedia CIRP,** v. 93, p. 413–418, 2020. 53rd CIRP Conference on Manufacturing Systems 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S2212827120307435.

JIMÉNEZ, Y. M.; PALACIO, J. C.; NOWÉ, A. Multi-agent reinforcement learning tool for job shop scheduling problems. In: DORRONSORO, B. *et al*. (Ed.). **Optimization and Learning**. Cham: Springer International Publishing, 2020. p. 3–12.

LUNARDI, W. T.; VOOS, H. An extended flexible job shop scheduling problem with parallel operations. **SIGAPP Appl. Comput. Rev., Association for Computing Machinery,** New York, NY, USA, v. 18, n. 2, p. 46–56, jul. 2018. Disponível em: https://doi.org/10.1145/3243064.3243068.

LUO, X.; QIAN, Q.; FU, Y. F. Improved genetic algorithm for solving flexible job shop scheduling problem. **Procedia Computer Science,** v. 166, p. 480–485, 2020. ISSN 1877-0509. Proceedings of the 3rd International Conference on Mechatronics and Intelligent Robotics (ICMIR-2019). Disponível em: https:

//www.sciencedirect.com/science/article/pii/S1877050920301836.

MANERBA, D. *et al*. Machine learning and optimization for production rescheduling in industry 4.0. **International Journal of Advanced Manufacturing Technology**, v. 10 2020.

PARK, I.-B. *et al*. A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. **IEEE Transactions on Automation Science and Engineering**, v. 17, n. 3, p. 1420–1431, 2020.

MARINHO, Walter Schmidt. **Análise comparativa de algoritmos aplicados ao FJSSP.** 2021. 39 f. TCC (Graduação) - Curso de Ciência da Computação, Centro Universitário Filadélfia, Londrina, 2021.

SHIUE, Y.-R.; LEE, K.-C.; SU, C.-T. A reinforcement learning approach to dynamic scheduling in a product-mix flexibility environment. **IEEE Access**, v. 8, p. 106542–106553, 2020.

WANG, Y. *et al*. Improving nsga-iii for flexible job shop scheduling using automatic configuration, smart initialization and local search. In: . New York, NY, USA: **Association for Computing Machinery, 2020. (GECCO '20)**, p. 181–182. ISBN 9781450371278. Disponível em: https://doi.org/10.1145/3377929.3389924.

ZHU, J.; WANG, H.; ZHANG, T. A deep reinforcement learning approach to the flexible flowshop scheduling problem with makespan minimization. In: **2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS).** [S.l.: s.n.], 2020. p. 1220–1225.