UM ESTUDO EM HIPER-HEURÍSTICA BASEADA EM MULTI-ARMED BANDITS APLICADA AO FLEXIBLE JOB SHOP SCHEDULING PROBLEM

A STUDY ON HYPER-HEURISTICS BASED ON MULTI-ARMED BANDITS APPLIED TO THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Eron P. Pereira [1]
Sergio Akio Tanaka [2]
Sergio Kenji S. Tanaka[3]

RESUMO

Na indústria manufatureira um dos fatores-chave para a eficiência e rentabilidade da empresa é o escalonamento ideal do processamento do produto. Tal escalonamento consiste na escolha das máquinas da fábrica que irão realizar cada operação para processamento do produto. Esse problema pode se estender a qualquer viés de escalonamento, tanto para o campo da educação com a escala de professores nas disciplinas, quanto para a produção agroindustrial e indústrias têxteis, para solucionar esse tipo de problema foi desenvolvida uma inteligência artificial para solucionar o problema. Este estudo apresenta o desenvolvimento para aplicar uma hiper-heurística baseada em *multi-armed-bandits (MAB)* aplicada ao *flexible job shop scheduling problem (FJSP).*

Palavras-chave: multi-armed-bandits; flexible-job-shop; hiper-heurística; algoritmo-genético.

ABSTRACT

In the manufacturing industry, one of the key factors for the company's efficiency and profitability is the optimal scheduling of product processing. Such scheduling consists of choosing the machines in the factory that will perform each operation for processing the product. This problem can extend to any scaling bias, both for the field of education with the scale of teachers in the disciplines, and for agro-industrial production and textile industries, to solve this type of problem, an artificial intelligence was developed to solve the problem. This study presents the development to apply a hyper-heuristic based on *multi-armed-bandits* (*MAB*) applied to the *flexible job shop shop scheduling problem* (*FJSP*).

Keywords: multi-armed-bandits; flexible-job-shop; hyper-heuristic; genetic-algorithm.

[1] Graduando do Curso de Ciência da computação do Centro Universitário Filadélfia - UniFil. eronponcepereira@gmail.com
[2] Graduando do Curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil. serginho.tanaka@edu.unifil.br
[3] Professor e orientador do Curso de Ciência da Computação do Centro Universitário Filadélfia - UniFil. sergio.tanaka@unifil.br

1 INTRODUÇÃO

Este trabalho foi estruturado de uma forma com que o leitor, entenda os pontos principais da pesquisa desenvolvida e os conceitos estudados durante o processo de pesquisa, posteriormente explicados na seção de desenvolvimento.

Problemas combinatórios fazem parte do conjunto de problemas NP-HARD (GOLDBARG, 2005), dentre eles se destacando problemas como o Caixeiro Viajante (*Wandering Salesman*), escalonamentos de fábrica (e.g. *Job Shop Scheduling, Flow Shop Scheduling, Flexible Job Shop*) e o problema da mochila (*Knapsack Problem*).

Estes problemas se destacam pela sua complexidade e, por consequência, o elevado tempo necessário para sua resolução. Assim, pesquisadores e desenvolvedores têm criado algoritmos e metodologias para encontrar soluções de uma maneira mais rápida e eficaz.

Existem diversas pesquisas sendo feitas sobre Problemas combinatórios, desde 1985 (BERRY; FRISTEDT,1985) até os anos de 2021 (VIANA,2021), trazendo assim a motivação para continuidade de pesquisas mais a fundo destes tópicos.

Além de artigos sendo escritos, existem também aplicações na no cotidiano, como, por exemplo, na criação de padrões têxteis de culturas quase perdidas em tecido (OBE; EGWUCHE, 2018), além deste padrão é possível criar padrões em indústrias, como, por exemplo, qual produto colocar em um maquinário específico para melhor resultado. Dentre estas aplicações e motivações existe um dos algoritmos criados dentro do grupo de pesquisa em inteligência artificial da *UNIFIL*, de Andrade (2020).

O *makespan* foi escolhido como o principal fator *fitness* a ser considerado para um bom algoritmo por conta de sua frequente adoção na maioria dos casos no domínio *FJSP*, pois, mesmo que o foco do algoritmo seja menor poluição ambiental ou gasto energético, o fator tempo é o mais levado em conta.

A universalidade do *makespan* faz com que o algoritmo seja facilmente desenvolvido em diferentes cenários, pois, não é necessário remover nenhum fator, somente adicionar caso seja necessário, como, por exemplo, considerar o fator de poluição como *fitness* secundário da operação.

Em seu estudo, Andrade (2020) aponta que os resultados obtidos pelo algoritmo foram competitivos, entretanto não excederam os de outros estudos, tal como Viana (2019). Os autores apontam, então, que a aplicação de outra hiper-heurística com seu *HGA* pode potencialmente gerar resultados ainda mais promissores.

Dentre as hiper-heurísticas de seleção, a heurística *Multi-Armed Bandits* (*MAB*) se mostrou muito eficiente em outros problemas combinatórios, em especial o *Flow Shop Problem* (*FSP*) (ALMEIDA et al. 2020). Foi levantada então a hipótese que sua aplicação sobre o *JSSP* pode tornar o algoritmo ainda mais eficiente. Caso este algoritmo obtenha bons resultados, é possível adaptar o algoritmo a problemas mais complexos, como o *Flexible Job Shop Problem* (*FJSP*).

O algoritmo *MAB* foi desenvolvido para solucionar um problema de recompensas desconhecidas do mesmo nome, o qual representava uma série de máquinas caça-níqueis, visando maximizar a quantidade de dinheiro ou fichas obtidas durante uma quantidade de jogadas (GLIMNE, 2015).

O algoritmo *MAB* se adaptou muito bem como uma hiper-heurística de seleção, selecionando quais heurísticas de baixo nível devem ser aplicadas, análogas com os braços das máquinas de caça-níqueis. Desde então, foram criadas diversas versões do *MAB*, tais como o *Fitness Rate Rank MAB* (*FRRMAB*), *Restless MAB* (*RMAB*) e *MABs* Contextuais.

2 DESENVOLVIMENTO

O estudo desenvolvido na *UNIFIL*, aplicado ao grupo de pesquisa em inteligência artificial, foi desenvolvido a partir da leitura de diversos artigos científicos apresentados nas referências e codificação de métodos para desenvolvimento do conhecimento matemático.

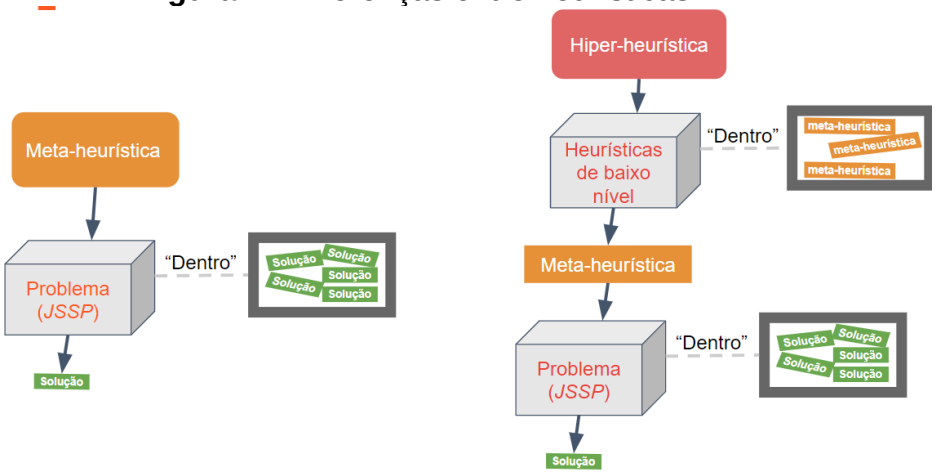
Heurísticas, meta-heurísticas, hiper-heurísticas e heurísticas de baixo nível

As heurísticas podem ser classificadas em três tipos. As heurísticas de baixo nível (*low-level heuristics*) são desenvolvidas para problemas específicos, e

diretamente geram soluções para os problemas. Meta-heurísticas, por sua vez, são heurísticas genéricas, e podem ser adaptadas para diversos problemas. Por fim, as hiper-heurísticas trabalham com o controle e geração de heurísticas de baixo nível, sem trabalhar com o problema diretamente (FERREIRA et al. 2015).

É possível verificar na Figura 1 a diferença entre meta-heurística e hiper-heurística, onde à esquerda é obtido uma solução do problema *JSSP* por meio de uma meta-heurística, e à direita o problema é resolvido a se obter uma meta-heurística por meio de uma hiper-heurística, que faz a seleção das heurísticas para solução do problema.

Figura 1 - Diferenças entre heurísticas



Dentre as meta-heurísticas, Andrade (2020) desenvolveu um novo algoritmo seguindo a estrutura hiper-heurística/meta-heurística, um Algoritmo Genético híbrido (*HGA*) com o uso da hiper-heurística *Modified Choice Function* (*MCF*) para seleção de operadores de cruzamento e mutação. Este algoritmo, denominado *MCF/HGA*, foi desenvolvido para solucionar problemas do *Job Shop Scheduling Problem* (*JSSP*).

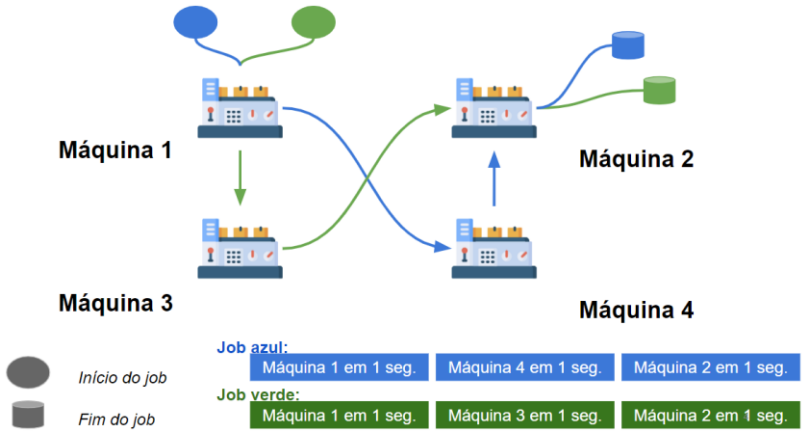
Flexible-Job-Shop-Scheduling-Problem

O Flexible Job Shop Problem (*FJSP*), também conhecido como *Flexible Job Shop Scheduling Problem* (*FJSSP*), é um problema em que *n jobs* devem ser

processados por *m* máquinas. Um *job* é composto por um conjunto de operações que devem ser processadas em uma determinada ordem.

Cada operação pode ser processada em um dado conjunto de máquinas, ao invés de uma máquina específica como no Job Shop clássico, o *JSP* é exemplificado na Figura 2. Esse tipo de problema busca escalonar as operações de modo a otimizar uma função objetivo, como, por exemplo, diminuir o *makespan*, conhecido como tempo total de produção, é o tempo no qual a última etapa do último *job* termina de ser produzida. Quanto menor o *makespan*, mais rápida a produção, contribuindo para a eficiência da fábrica.

Figura 2 - Funcionamento do *JSP*



Fonte: Marinho (2021).

No *FJSP* é necessário fazer a atribuição de cada operação para uma máquina apropriada, tendo em vista que pode existir mais de uma máquina possível para cada operação de um produto, além de fazer o sequenciamento das operações no escalonamento. Desse modo, o *FJSP* mostra-se mais complicado do que o *JSP*, e tem-se provado fortemente *NP-hard.*

Alguns dos *frameworks* criados para resolver este tipo de problema envolvem a combinação de metodologias denominadas meta-heurísticas e hiper-heurísticas. Estes *frameworks* têm recebido atenção pela sua eficiência, como demonstrado em Chaurasia (2018), Lin (2019) e Almeida et al. (2020).

O *FJSP* foi escolhido por conta de sua flexibilidade e potencial de adaptação em meio a um chão de fábrica, diferente de sua versão anterior *JSP* onde só é possível

dedicar um *job* para uma máquina somente. Outro fator muito importante é que trabalhos sobre o *JSP* já foram produzidos por Andrade (2020), assim com que os processos adiantes sejam mais consistentes.

Multi Armed Bandits (MAB)

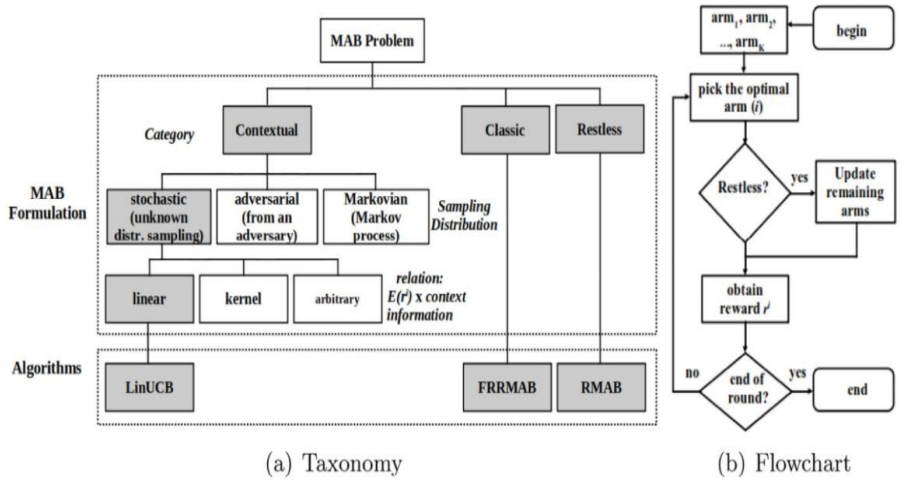
O problema de Bandidos de múltiplos braços é um problema de otimização de recompensas desconhecidas, onde existem *n* opções a serem consideradas, cada uma com uma recompensa desconhecida, onde deseja-se maximizar a recompensa obtida. Data-se o estudo deste tipo de problema a artigos publicados por Thompson (1933).

O problema *Multi-Armed Bandits* foi muito investigado, conforme relata a primeira edição do livro "*Multi-armed Bandit Allocation Índices*", publicado em 1989. Este foi criado para representar as máquinas de caça níqueis que se tornaram populares no final do século XIX, as quais eram conhecidas como bandidos de um braço (GLIMNE, 2015).

Desde a publicação por John Gittins em 1989, novas iterações do algoritmo foram propostas, com o principal foco em balancear a exploração (*exploration*) e extração (*exploitation*) (GITTINS; GLAZEBROOK; WEBER, 2011).

A Figura 3 representa um *workflow* dos tipos de *MAB*, sendo clássico o que segue a estrutura tradicional.

Figura 3 - Representação *workflow* dos estilos de *MAB*



O *FRRMAB* é uma categoria de *MAB* que utiliza a aptidão das heurísticas de baixo nível e a qualidade das soluções geradas por estas como principal fator de pontuação. Não sendo obrigatória a utilização de outras variáveis das heurísticas de baixo nível além do *fitness.* Quando o *FRRMAB* não está realizando testes preliminares, ou para assegurar a qualidade das heurísticas de baixo nível, o algoritmo seleciona a heurística com melhor *fitness* por padrão.

O *RMAB* alterna entre períodos de exploração e extração dentro do algoritmo, começando com um período de exploração, verificando todas as heurísticas de baixo nível, procurando a melhor opção para ser extraída. Gasta memória em operações não tão necessárias, porém explora ao máximo o algoritmo.

O *MAB UCB1* consiste no *MAB* onde seleciona os braços e o que ele toma como decisão é a fórmula de "Desconfiança" existente, que consiste em uma fórmula determinada que faz com que a cada iteração o valor de desconfiança mude e seja interferido com o contexto atual (ALMEIDA et al. 2020), O algoritmo contextual utiliza informações que o clássico não leva em consideração, como variáveis que pertencem ao próprio problema.

Genetic Algorithm (GA)

O algoritmo genético (*GA*) é um algoritmo evolucionário baseado na teoria da evolução (XHAFA; ABRAHAM, 2008), onde, a partir de uma população de soluções para um problema, são realizados cruzamentos e mutações visando a criação de novas soluções conforme as iterações do algoritmo.

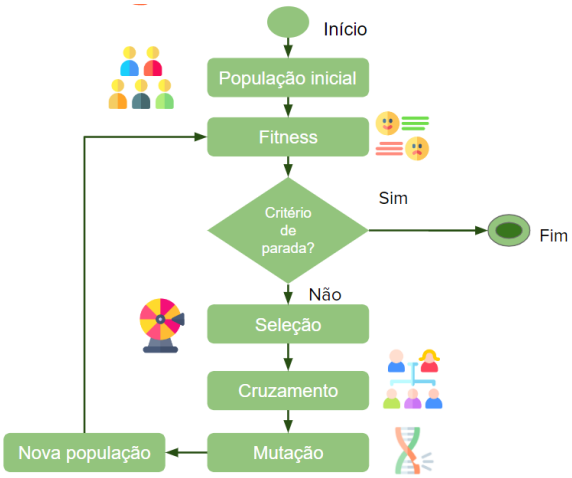
Para avaliar a qualidade das soluções é utilizada uma medida denominada aptidão (*fitness*). Esta medida é definida de acordo com o problema, tal algoritmo leva em conta soluções com melhor aptidão para serem escolhidas para cruzamento com maior frequência.

O algoritmo genético é um algoritmo baseado na teoria da evolução. Neste, os indivíduos, denominados cromossomos, de uma população representam as soluções para um problema, os quais realizam por cruzamento e mutação para gerar novas soluções (VIANA, 2019). O *HGA* é, então, uma versão modificada do *GA*, a qual faz

uso de outras heurísticas além do cruzamento e mutação, tais como o elitismo e as hiper-heurísticas, para melhorar sua performance.

A Figura 4 exemplifica um algoritmo genético, em respectiva ordem, o algoritmo seleciona uma população inicial, verifica a eficiência dos resultados (*fitness*) e caso seja a opção de continuar, o algoritmo seleciona uma população e realiza as operações de cruzamento e mutação, para assim gerar uma nova população.

Figura 4 - Algoritmo genético



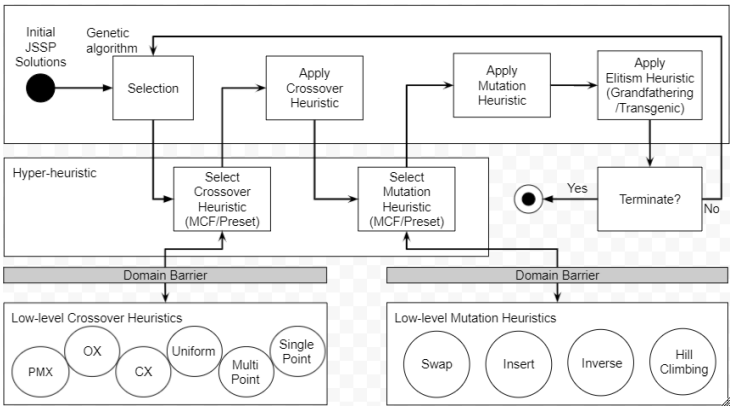
Para determinar quais cromossomos farão parte do processo de cruzamento, lhes é atribuída uma pontuação de aptidão (*fitness*) com base na qualidade da solução para o problema a ser resolvido. Quanto melhor o fitness, maior a probabilidade do cromossomo ser selecionado para o cruzamento. No caso do *JSSP*, o fator de fitness mais comumente utilizado é o *makespan*.

Hybrid Genetic Algorithm (HGA)

A Figura 5 exemplifica um algoritmo genético híbrido desenvolvido durante o período de pesquisa, a etapa inicial do mesmo é a seleção de um grupo de escalonamento com melhor *makespan*, após isto é realizado o *crossover* entre dois escalonamentos, agora chamados de cromossomos. Existem diversos tipos de fazer o crossover com dois escalonamentos, ao ser selecionado um tipo de heurística de *crossover* é aplicado entre os dois escalonamentos, logo depois ocorre a mutação para os escalonamentos gerados, esta mutação é escolhida por meio de uma

heurística de seleção de mutação, após aplicada é separado as gerações novas melhores e antigas melhores, fazendo uma implementação entre, deixando os 90% melhores novos e 10% dos melhores velhos. Assim realocando na piscina de escalonamentos. Esta seleção é chamada de elitismo.

Figura 5 - Workflow do GA



Encoding de dois vetores máquina operação (Encoding MO)

Dado um escalonamento de um problema do job shop flexível foi necessário encontrar uma forma de efetuar o cruzamento e *crossover*, assim foi desenvolvido um *encoding* para tal processo com base no artigo de Yuan, Xu. (2013) e nomeado conforme o funcionamento do mesmo. O funcionamento é baseado em dois vetores sendo *V* o vetor sequencial de operações dos jobs e *U* o vetor correspondido como as máquinas denominadas para cada *job* em ordem decrescente de *makespan*.

O vetor *V* é composto pelo escalonamento da operação, como por exemplo, temos um escalonamento *V* = [6,1,7,3,4,2,5] dado os jobs da Figura 6.

Figura 6 - Jobs e suas operações

Job	Operation	M_1	M_2	M_3
J_1	$O_{1,1}$	2	–	3
	$O_{1,2}$	4	5	2
J_2	$O_{2,1}$	–	3	5
	$O_{2,2}$	3	2	3
	$O_{2,3}$	–	–	4
J_3	$O_{3,1}$	4	5	6
	$O_{3,2}$	3	–	2

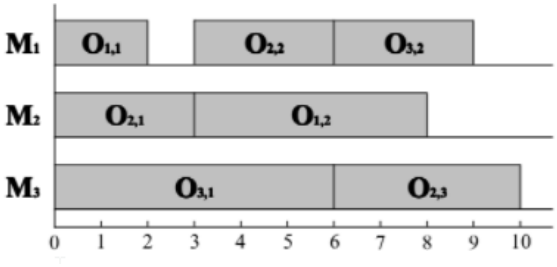
Consequentemente o escalonamento é denominado com número de 1 ao número de *jobs* disponíveis, considerando que cada job tem um número de operações tendo que ser respeitado pelo escalonamento, dado que, o *Job1* consiste de duas operações, no escalonamento determinado com o número “1”e “2” os dois tem que estar em tal sequência para a ordem de operações ser considerada.

Ou seja, conforme o escalonamento *V* a primeira sequência a ser inicializada é a 6, representada como o *Job 3* operação 1 (O3, 1) o próximo passo é determinar qual máquina o mesmo irá ser processado, isto será determinado pelo vetor *U*.

O vetor *U* é o vetor que determinará as máquinas a serem utilizadas em cada *job*, a ordem do vetor é conforme a ordem de job e suas operações e o número representado no vetor corresponde a colocação da máquina em ordem crescente do *makespan*. Ou seja, dado o vetor *U* = [1,3,1,2,1,3,2] é possível interpretar que o primeiro valor determina que o *Job 1* operação 1 (O1, 1) será processado pela primeira máquina de menor tempo.

Com estes dados é possível fazer a representação de operações conforme a Figura 7.

Figura 7 - Escalonamento das operações



Conforme o vetor de operações *V* a primeira operação do escalonamento foi (O3, 1) com e a máquina escolhida a terceira de menor tempo, ou seja, a máquina M3 com o *makespan* de 6 unidades de tempo, e por assim em diante.

3 CONCLUSÕES

O processo de escolha de qual hiper-heurística que selecionará as heurísticas

de baixo nível no algoritmo genético híbrido está em estudo, por um lado o *MAB* contextual traz um benefício para o algoritmo, porém aborda-se áreas para além da pesquisa atual, sendo necessário a maestria de conceitos ainda não vistos.

Dentre os estilos de *MAB* o que mais satisfaz para a utilização é o *FRRMAB*, utilizado por conta de sua facilidade de implementação e complexidade média. O *FRRMAB* aplica uma fórmula para seleção de *fitness* em uma dada solução, como, por exemplo, o *LinUCB*, criando uma certa "desconfiança" entre as respostas, fazendo o algoritmo explorar mais possibilidades e tender a outras opções se o fitness estiver estagnado.

O objetivo futuro da pesquisa é codificar a aplicação de hiper-heurísticas de seleção no algoritmo genético híbrido, com o objetivo de diminuir ao máximo o *makespan* da solução.

REFERÊNCIAS

ANDRADE; João Vitor da Costa. **Desenvolvimento de Uma Hiper-Heurística Aplicada ao Escalonamento em Problemas de Job Shop**. 70 f. Trabalho de Conclusão de Curso (Graduação) - Centro Universitário Filadélfia. Londrina, 2020

BERRY DA, FRISTEDT B. Bandit problems: Sequential Allocation of Experiments. **Springer**, [S.L.], Outubro 1985. ISBN: 9401537135. https://doi.org/10.1007/978-94-015-3711-7

CHAURASIA, S. N.; SUNDAR, S.; JUNG, D.; LEE, H. M.; KIM, J. H. An Evolutionary Algorithm Based Hyper-heuristic for the Job-Shop Scheduling Problem with No-Wait Constraint. **Harmony Search and Nature Inspired Optimization Algorithms**. [S. l.]: Springer Singapore, 2018. p. 249–257. DOI 10.1007/978-981-13-0761-4_25. http://dx.doi.org/10.1007/978-981-13-0761-4_25.

FERREIRA, Alexandre Silvestre; GONCALVES, Richard Aderbal; POZO, Aurora Trinidad Ramirez. A Multi-armed Bandit Hyper-Heuristic. **2015 Brazilian Conference On Intelligent Systems (Bracis)**, Curitiba, v. 0, n. 0, p. 13-18, nov. 2015. IEEE. http://dx.doi.org/10.1109/bracis.2015.31.

XHAFA Gonçalves, Richard Aderbal; de Almeida, Carolina Paula; Venske, Sandra; Delgado, Myriam; Hyper-heuristics using multi-armed bandit models for multi-objective optimization. **Applied Soft Computing,** v. 95, Elsevier, Outubro 2020, https://doi.org/10.1016/j.asoc.2020.106520

Gittins, John; Glazebrook, Kevin; Weber,Richard; **Multi-armed Bandit Allocation Indices**. John Wiley & Sons, 2011. ISBN 1119990211..

GLIMNE D. Slot machine. **Encyclopedia Britannica**, 12 Nov. 2015, https://www.britannica.com/topic/slot-machine. Accessed 26 July 2021.

GOLDBARG M. **Otimização Combinatória e Programação Linear**. [S. l.: s. n.], 2005. p. 536(536). ISBN-10 8535215204

LIN, T.-L.; HORNG, S.-J.; KAO, T.-W.; CHEN, Y.-H.; RUN, R.-S.; CHEN, R.-J.; LAI, J.-L.; KUO, I.-H. An efficient job-shop scheduling algorithm based on particle swarm optimization. **Expert systems with applications**, v. 37, n. 3, p. 2629–2636, Mar. 2010. DOI 10.1016/j.eswa.2009.08.015.http://dx.doi.org/10.1016/j.eswa.2009.08.015.

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.

OBE O, EGWUCHE O. Genetic algorithm approach for fabric pattern generation in textile industries. **Annals. Computer Science Series**, Nigeria, v. XVI, fasc. 2, 2018. https://doi.org/10.1093/biomet/25.3-4.285

THOMPSON, William R.. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. **Biometrika**, [S.l.], v. 25, n. 3/4, p. 285, dez. 1933. JSTOR. http://dx.doi.org/10.2307/2332286.

VIANA, Monique Simplicio *et al*. A New Genetic Improvement Operator Based on Frequency Analysis for Genetic Algorithms Applied to Job Shop Scheduling Problem. **Artificial Intelligence And Soft Computing**, [S.l.], p. 434-450, 2021. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-87986-0_39.

VIANA, M. S**. Uma abordagem de otimização utilizando algoritmo genético com busca local e um novo operador de transgenia para minimização do makespan no problema de programação da produção job shop**. Qualificação de Doutorado - UFSC, São Carlos, 2019. https://repositorio.ufscar.br/handle/ufscar/9087.

XHAFA, F.; ABRAHAM, A. Metaheuristics for Scheduling in Industrial and Manufacturing Applications. 1st. ed. [S.l.]: **Springer Publishing Company**, Incorporated, 2008. ISBN 3540789847. 10, 13

Yuan Y, Xu H. A memetic algorithm for the multi-objective flexible job shop scheduling problem. **GECCO '13. Association for Computing Machinery**, New York, 2013. https://doi.org/10.1145/2463372.2463431