UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE DESENVOLVIMENTO SUSTENTÁVEL DO SEMIÁRIDO
UNIDADE ACADÊMICA DE ENGENHARIA DE PRODUÇÃO
CURSO DE ENGENHARIA DE PRODUÇÃO

JÚLIO CESAR VASCONCELOS CONSERVA

ABORDAGEM METAHEURÍSTICA HÍBRIDA PARA MINIMIZAÇÃO DE CUSTOS POR ANTECIPAÇÃO E ATRASOS NA PRODUÇÃO

SUMÉ - PB
2022

JÚLIO CESAR VASCONCELOS CONSERVA

ABORDAGEM METAHEURÍSTICA HÍBRIDA PARA MINIMIZAÇÃO DE CUSTOS POR ANTECIPAÇÃO E ATRASOS NA PRODUÇÃO

Monografia apresentada ao Curso Superior de Engenharia de Produção do Centro de Desenvolvimento Sustentável do Semiárido da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Engenharia de Produção.

Orientador: Professor Dr. Yuri Laio Texeira Veras Silva.

SUMÉ - PB
2022

C775a Conserva, Júlio Cesar Vasconcelos.
Abordagem metaheurística híbrida para minimização de custos por antecipação e atrasos na produção. / Júlio Cesar Vasconcelos Conserva. - 2022.

51 f.

Orientador: Professor Dr. Yuri Laio Teixeira Veras Silva.
Monografia - Universidade Federal de Campina Grande; Centro de Desenvolvimento Sustentável do Semiárido; Curso de Engenharia de Produção.

1. Job-Shop Scheduling. 2. Custos por antecipação e atraso. 3. Algoritmo genético. 4. Minimização de custos. 5. Sequenciamento da produção. 6. Benchmark. 7. Abordagem metaheurística híbrida – Engenharia de Produção. I. Silva, Yuri Laio Teixeira Veras. II. Título.

CDU: 658.5(043.1)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

JÚLIO CESAR VASCONCELOS CONSERVA

ABORDAGEM METAHEURÍSTICA HÍBRIDA PARA MINIMIZAÇÃO DE CUSTOS POR ANTECIPAÇÃO E ATRASOS NA PRODUÇÃO

Monografia apresentada ao Curso Superior de Engenharia de Produção do Centro de Desenvolvimento Sustentável do Semiárido da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Engenharia de Produção.

BANCA EXAMINADORA:

Professor Dr. Yuri Laio Texeira Veras Silva.
Orientador - UAEP/CDSA/UFCG

Professor Me. Josean da Silva Lima Júnior.
Examinador I - UAEP/CDSA/UFCG

Professora Me. Luana Marques Sousa Farias.
Examinadora II - UAEP/CDSA/UFCG

Trabalho Aprovado em: 20 de outubro de 2022.

SUMÉ - PB

AGRADECIMENTOS

A Deus, por ser tão bondoso me dando forças para vencer as dificuldades e permitindo que eu tivesse saúde. Por me iluminar diariamente com oportunidades de evoluir e ser uma pessoa melhor e se mantendo ao meu lado para superar as adversidades.

A minha mãe Jucilene, pelo amor e apoio incondicional aos meus sonhos mesmo diante das inúmeras dificuldades. Por ter sido essa guerreira, fazendo de tudo para proporcionar o melhor a nossa família. Por me dar diariamente exemplos de bondade estando sempre de coração aberto para ajudar o próximo. Essa conquista também é sua.

Aos meus familiares, principalmente meus irmãos Ana Clara, José Júnior e José Lucinaldo, por sempre incentivarem dando apoio durante minha trajetória. A minha sobrinha Rebeca por ter iluminado minha vida com bastante alegria em um momento complicado da jornada. Ao meu tio Naldo por ser um pai para mim, dando conselhos, ensinamentos e sempre lutando pela minha saúde sempre que necessitei. Aos meus avós maternos (*in memorian*) Sebastiana e Bonifácio, por toda dedicação a família, todos os ensinamentos que me deram e por também terem sonhado comigo.

A Hedivigem, por todo amor e carinho que foram tão importantes nessa etapa da minha vida. Por sempre me incentivar e dar apoio nos momentos difíceis, sendo sempre companheira e sendo um exemplo de resiliência.

Aos meus amigos Johan, Jelson, Benjamim e Felipe que mesmo com a distância sempre me incentivaram nessa jornada.

Aos amigos que o curso e Sumé me deram, que tornaram esta caminhada mais leve e cheia de boas lembranças (Jordana, Sandy, Marcielly, Gustavo, Matheus, Higor, Larissa e Keren). E aos colegas eu ganhei durante minha passagem na ProdUP Jr., sendo muito importantes no meu desenvolvimento.

Em especial aos irmãos que essa jornada me deu, João Pedro, Lucas Almeida, Marcelo Rodrigues e Yan Rink, que compartilharam comigo grandes momentos de aprendizado e parceria.

Ao meu orientador Yuri Laio, por ter aceitado me guiar neste desafio, compartilhando seus conhecimentos e dando sugestões para agregar a este trabalho, com muita dedicação e paciência. Obrigado pelos ensinamentos durante este trabalho, as aulas e projetos, com certeza você se tornou um exemplo de profissional para mim.

Ao professor Walton que me abriu as portas da iniciação cientifica, fazendo com que despertasse meu gosto pela área de pesquisa operacional.

Aos professores e funcionários do CDSA, que deram sua contribuição para o meu desenvolvimento ao compartilhar seus conhecimentos e experiências, sendo todos muito importantes na minha formação.

RESUMO

O *Job-Shop Scheduling* (JSS) é um problema de otimização que envolve processos decisórios a respeito da ordem em que pedidos de produção devem ser executados em um determinado conjunto de máquinas e etapas. Esse problema é bastante conhecido e vem recebendo grande atenção da literatura nos últimos anos. Diante disso, este trabalho tem como objetivos realizar um estudo acerca de uma variante do JSS, chamado de *Job-Shop Scheduling* com minimização de custos por antecipação e atraso (JSS-ETC) e propor algoritmos eficientes para sua resolução. No JSS-ETC é considerado os custos e adversidades operacionais causadas pela antecipação e atraso que ocorre quando existe a imprecisão da programação da produção, buscando-se a adoção da filosofia *Just-in-Time*. Nesta pesquisa, são apresentadas duas metaheurísticas baseadas em Algoritmo Genético e Busca Local. Os experimentos computacionais apresentados atestam a efetividade dos algoritmos propostos por meio da realização de *benchmark* com algoritmos presentes na literatura.

Palavras-chave: *Scheduling*; Custos por Antecipação e Atraso; Algoritmo Genético.

ABSTRACT

Job-Shop Scheduling (JSS) is an optimization problem that involves decision processes regarding the order in which production orders should be executed in a given set of machines and steps. This problem is well known and has received great attention in the literature in recent years. Therefore, this work aims to carry out a study about a variant of JSS, called Job-Shop Scheduling with minimization of costs by earliness and tardiness (JSS-ETC) and to propose efficient algorithms for its resolution. In JSS-ETC, costs and operational adversities caused by earliness and tardiness that occur when there is inaccuracy in the production schedule are considered, seeking to adopt the Just-in-Time philosophy. In this research, two metaheuristics based on Genetic Algorithm and Local Search are presented. The computational experiments presented attest to the effectiveness of the proposed algorithms by performing a benchmark with algorithms found in the literature.

Keywords: Scheduling; Earliness and Tardiness Costs; Genetic Algorithm.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Ao se analisar o cenário em que as instituições estão inseridas atualmente, é possível notar a busca constante pela obtenção de vantagem competitiva como forma de se destacar em um mercado cada vez mais acirrado. Em meio a esse desafio, Siqueira e Müller (2022) destacam que as organizações têm notado que a chave para criar um valor superior aos seus concorrentes está, principalmente, no melhor gerenciamento de seus recursos.

A gestão dos inúmeros recursos que as organizações precisam lidar no dia a dia é uma tarefa bastante complicada, que tende a ser ainda mais complexa a depender do porte e da quantidade de informações existentes. Dentre os vários problemas que abrangem a tomada de decisão envolvendo recursos em processos produtivos, pode-se citar: dimensionamento de lotes de produção, planejamento da produção, otimização da capacidade instalada e sequenciamento da produção.

Nesse contexto, problemas de sequenciamento e programação da produção (do inglês, *scheduling*) têm recebido uma grande atenção nas organizações nos últimos anos. Vallada e Ruiz (2012) destacam que tais problemas estão relacionados à alocação de recursos físicos (máquinas) para a realização/processamento de um determinado conjunto de tarefas (pedidos de produção) ao longo de um horizonte temporal. Isto é, referem-se aos processos decisórios a respeito da ordem em que os pedidos de produção devem ser executados em um determinado conjunto de máquinas e etapas.

Os problemas de *scheduling* possuem diversas variantes na literatura, algumas dessas variantes são os problemas de *Job-Shop Scheduling* (JSS) e *Flexible Job-Shop* (FJSS) que possuem diversas abordagens de aplicação. Os problemas clássicos de JSS e FJSP tratam de alguns objetivos comuns, como os de minimização de tempo máximo de execução, carga por máquina ou de número *jobs* em atraso. Contudo, no mundo real as organizações têm buscado cada vez mais adotar a filosofia *Just-in-Time* (JIT) como forma de alcançar maior eficiência, se fazendo assim necessário adotar critérios nos problemas de *scheduling,* que representem os custos e adversidades operacionais causadas pela antecipação-atraso (ETC, do inglês *Earliness and Tardiness Costs*) mostrando o real impacto que a imprecisão da programação da produção pode causar para as empresas. Então, se um *job* for concluído exatamente na data de vencimento, seu custo de antecipação-atraso será igual a zero. Caso contrário, a conclusão do *job* antes (depois) da data de vencimento incorre em um custo de antecipação (respectivamente, atraso) diferente de zero (AHMADIAN; SALEHIPOUR; KOVALYOV, 2020).

Essas particularidades podem ser vistas no setor de manufatura alimentício, que é caracterizado por processos produtivos com alto grau de complexidade, principalmente devido à natureza de seus produtos acabados. Por geralmente lidar com insumos e produtos acabados com prazos de validade curtos e alto controle de qualidade, as tomadas de decisão envolvendo o sequenciamento da produção em organizações dessa natureza possuem uma importância fundamental em seus resultados operacionais, bem como no abastecimento adequado dos atacadistas e varejistas que atendem as populações na ponta final da cadeia de suprimentos.

Os problemas de *scheduling*, em suas diversas naturezas, envolvem processos de tomada de decisão de alta complexidade matemática e computacional, podendo ser encontrados em organizações dos diversos setores industriais, tanto de manufatura, como de serviços (SILVA *et al*., 2018). Todavia, resolver alguns problemas de sequenciamento da produção é uma tarefa bastante complexa por vários deles pertencerem a classe *NP-Hard*, conforme destaca Peng *et al.* (2019), o que torna sua resolução por meio de métodos exatos pouco efetiva em instâncias de dimensões elevadas, dado não ser possível obter sua solução ótima em um tempo polinomial, especialmente em problemas de gestão da produção com múltiplos estágios, máquinas e operadores (GUPTA, 2006).

Com isso, os métodos heurísticos têm ganhado bastante destaque na otimização desses problemas por apresentarem soluções satisfatórias em um menor tempo quando comparado aos métodos exatos. De acordo com Amjad *et al.* (2018), o Algoritmo Genético (GA, do inglês *Genetic Algorithm*) tem se mostrado uma das técnicas mais eficazes para resolver os problemas de JSS e suas variantes, sendo uma das técnicas mais utilizadas atualmente.

O GA é uma técnica que tem sido muito utilizada em vários campos da ciência e em diversos tipos de problemas, demonstrando resultados sólidos não apenas em *scheduling*. Esse método vem sendo aplicado de forma híbrida com outras heurísticas, que possibilita, muitas vezes, alcançar uma melhora considerável em seu desempenho. Essa hibridização tem apresentado resultados interessantes em vários estudos, o que evidencia o potencial dessas técnicas, como no estudo de Yuce *et al.* (2017).

Diante desse cenário, essa pesquisa estuda o problema de JSS com minimização de custos por antecipação e atraso (JSS-ETC, do inglês *Job-Shop Scheduling with minimizing Earliness and Tardiness Costs*), e o aplica em uma empresa do ramo alimentício. Além disso, são propostos dois Algoritmos Genéticos hibridizados com uma heurística de Busca Local para o problema em questão e apresentada uma vasta revisão de literatura. Posteriormente, são expostas algumas análises dos resultados obtidos por meio dos experimentos computacionais,

com o intuito de investigar o desempenho do método proposto em diferentes instâncias e parâmetros.

O restante do trabalho está organizado da seguinte maneira. Na seção 2, apresenta-se os objetivos deste estudo, enquanto na 3 é exposto uma revisão de literatura com os conceitos que envolvem os problemas de *scheduling*, heurísticas evolucionárias onde é explicado cada operador que pode ser utilizado no algoritmo genético e pôr fim a seção 3 ainda expõe os trabalhos no campo de JSS e que usaram o GA como método de resolução. Na seção 4 é apresentada a metodologia de pesquisa que norteou o desenvolvimento deste trabalho. A seção 5 apresenta a definição formal do problema tratado por esta pesquisa, em seguida explica como foi a configuração dos experimentos e expõe os resultados obtidos nesses experimentos computacionais, posteriormente é exibido um *benchmark* com os resultados encontrados na literatura. A seção 6 conclui este trabalho e traz um direcionamento para pesquisas futuras.

2 OBJETIVOS

Diante do cenário apresentado, esse estudo tem como objetivos:

2.1 OBJETIVO GERAL

Desenvolver e implementar uma abordagem de resolução fundamentada em uma metaheurística de otimização evolutiva e híbrida que possibilite a resolução do problema de JSS com minimização de custos por antecipação e atraso (JSS-ETC, do inglês *Job-Shop Scheduling with minimizing Earliness and Tardiness Costs*), que esta presente em empresas do ramo alimentício.

2.1.1 Objetivos específicos

- Realizar uma vasta revisão crítica da literatura de *scheduling*, visando aprofundar os conhecimentos acerca de abordagens metaheurísticas de otimização voltadas para a resolução de problemas de *scheduling*;
- Desenvolver e Implementar algoritmos de otimização propostos, fundamentados em metaheurísticas evolucionárias e híbridas, mais especificamente Algoritmos Genéticos Híbridos;
- Analisar os resultados alcançados com a aplicação da abordagem desenvolvida nas instâncias de dados consolidadas na literatura, referentes ao problema estudado;
- Comparar os resultados alcançados pela aplicação dos métodos desenvolvidos com os principais métodos e resultados consolidados na literatura, de modo a verificar a eficiência e viabilidade da abordagem proposta;

3 REVISÃO DE LITERATURA

Esta seção é dedicada a apresentação do levantamento bibliográfico que foi realizado para fundamentar esta pesquisa. Aqui são abordados conceitos de sequenciamento da produção, *Job-Shop-Scheduling*, Algoritmos evolucionários e Algoritmos Genéticos. Por fim, são apresentados os trabalhos relacionados com o JSS e ainda com GA.

3.1 SEQUENCIAMENTO DA PRODUÇÃO

O trabalho de Jiang *et al.* (2021), ressalta que o sequenciamento da produção tem como objetivo entregar produtos ou serviços sob demanda aos consumidores em tempo hábil. Os autores ainda destacam que o sequenciamento da produção é um procedimento complexo de tomada de decisão que atribui recursos para diferentes tarefas em tempo regulado, visando otimizar um ou mais objetivos sob restrições relacionados a processos, recursos e outros fatores.

Já de acordo com Buettgen (2011), o sequenciamento da produção (*scheduling*) determina a sequência em que as necessidades de clientes serão atendidas pela empresa e define a prioridade da execução das tarefas de acordo com os critérios de sequenciamento definidos pela organização.

Diante disso, Tubino (2007) apresenta uma série de regras de sequenciamento da produção:

Quadro 1 - Regras de sequenciamento.

Sigla	Especificação	Definição
PEPS	Primeira que entra primeira que sai	Os lotes serão processados de acordo com sua chegada no recurso.
MTP	Menor tempo de processamento	Os lotes serão processados de acordo com os menores tempos de processamento no recurso.
MDE	Menor data de entrega	Os lotes serão processados de acordo de acordo com as menores datas de entrega.
IPI	Índice de prioridade	Os lotes serão processados de acordo com o valor da prioridade atribuída ao cliente ou ao produto.
ICR	Índice crítico	Os lotes serão processados de acordo com o menor valor de: $\frac{(data\ de\ entrega - data\ atual)}{Tempo\ de\ processamento}$

IFO	Índice de folga	Os lotes serão processados de acordo com o menor valor de: $\frac{(data\ de\ entrega - \sum tempo\ de\ processamento\ restante}{número\ de\ operações\ restantes}$
IFA	Índice de falta	Os lotes serão processados de acordo com o menor valor de: $\frac{quantidade\ em\ estoque}{taxa\ de\ demanda}$

Fonte: Tubino (2007).

3.1.1 Job-Shop-Scheduling

O problema de sequenciamento da produção (*scheduling*) possui diversas variantes que são estudadas na literatura, cada uma se destaca por explorar alguma característica especifica, seja o número de recursos disponíveis (máquinas) e número de ordens ou pedidos de produção (*jobs*). Dentre as diversas variantes, uma que recebe bastante atenção é o JSS e o FJSS.

De acordo com Yazdani *et al.* (2017), o tradicional problema de JSS, consiste em escalonar n *jobs* em m máquinas de modo a respeitar várias restrições. Cada *job* tem um número de operações que devem ser processadas em sequência e, uma vez iniciada uma operação, nenhuma preempção é permitida. Cada máquina pode realizar apenas uma operação por vez e um *job* não pode ser processado por duas máquinas simultaneamente. O objetivo do problema de escalonamento de *job-shop* é encontrar um escalonamento para processar todos os *jobs* de uma maneira que otimize determinados objetivos de desempenho. Enquanto no FJSS, cada operação pode selecionar uma máquina de um conjunto de máquinas. O FJSP inclui dois subproblemas: atribuição de máquina que é selecionar uma máquina de um conjunto de máquinas candidatas; sequenciamento de operações que está programando as operações em máquinas selecionadas para obter uma solução viável. Portanto, o FJSP é mais difícil do que o JSS clássico (GAO *et al.*, 2016).

A seguir se apresentado um modelo matemático proposto por Pinedo (2005) para o JSS. O modelo tem os seguintes parâmetros:

A → Conjunto de precedências

N → Conjunto de operações

n → Número de tarefas

m → Número de máquinas

p_{ij} → Tempo de processamento da atividade i na máquina j

O modelo considera ainda as seguintes variáveis de decisão:

y_{ij} → Tempo inicial de produção da atividade i na máquina j

C_{max} → *Makespan*

O modelo matemático é apresentado a seguir:

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

$$s.t. \quad y_{hj} - y_{ij} \geq p_{ij}, \qquad \forall\,(i,j) \rightarrow (h,j) \in A \tag{2}$$

$$C_{max} - y_{ij} \geq p_{ij}, \qquad \forall\,(i,j) \in \mathrm{N} \tag{3}$$

$$y_{ij} - y_{ik} \geq p_{ik} \;\; ou \;\; y_{ik} - y_{ij} \geq p_{ij}, \;\; \forall\,(i,j)\; e\; (i,j), i = 1, \ldots, m \tag{4}$$

$$y_{ij} \geq 0, \qquad \forall\,(i,j) \in N \tag{5}$$

3.2 ALGORITMOS EVOLUCIONARIOS

Na busca por métodos cada vez mais eficientes para resolução de problemas de otimização, passou-se a enxergar na natureza e em seus comportamentos de sobrevivência e reprodução uma resposta para avançar nesse objetivo. Nesse contexto, surgem os algoritmos evolucionários, que em geral são inspiradas nesses processos biológicos de evolução e foram inicialmente apresentados por Holland (1975).

De acordo com Pacheco (2017), as heurísticas evolucionárias, diferentemente das mais tradicionais, trabalham com uma população de soluções candidatas ao mesmo tempo, o que permite uma varredura mais ampla no espaço de solução, explorando assim diferentes regiões. O processo de busca, faz com que seja destinado um quantitativo de indivíduos para exploração em cada região, dando preferência as mais promissoras, mas sem deixar de lado as demais. Isso é possível devido a presença de mecanismos de memória e aprendizado entre os indivíduos de certa população e ainda a aleatoriedade presente no método, que contribui para que se fuja de soluções ótimas locais.

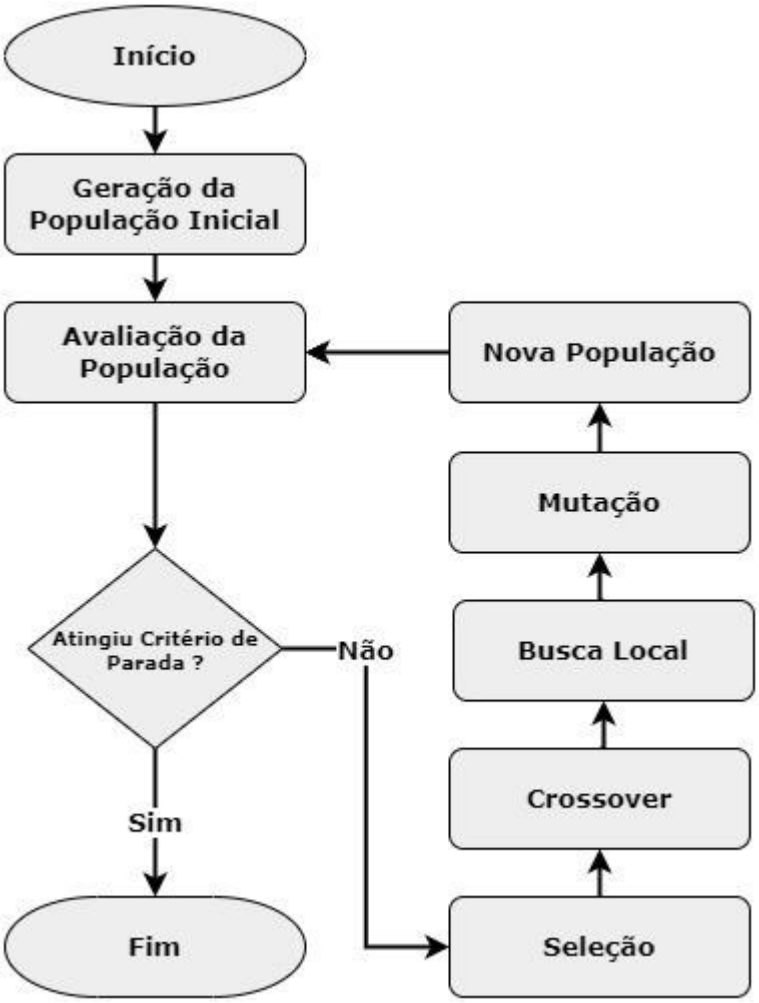
Freitas (2018) destaca que nos algoritmos evolucionários, alguns termos da biologia são utilizados para denotar elementos de otimização, de modo que gerações são iterações, indivíduos são soluções, população é um conjunto de soluções de cada iteração e função de aptidão é a função objetivo do algoritmo.

3.2.1 Algoritmo genético

Esta pesquisa fará o uso mais especificamente do GA, que é um método evolutivo que se inspira nos princípios da seleção natural, onde os indivíduos mais aptos, possuem uma maior probabilidade de reprodução e os indivíduos com mais descendentes têm mais chances de perpetuar seus códigos genéticos para as próximas gerações. Nas próximas seções serão apresentadas algumas características do GA e seus principais operadores.

Assim como outras heurísticas evolucionárias, o GA opera com um conjunto de soluções que é chamado de população, e cada membro desta população pode ser chamado de indivíduo ou cromossomo. Cada indivíduo é avaliado e atribuído um valor de aptidão (do inglês *Fitness*), onde as melhores soluções dos problemas estão correlacionadas com os melhores valores de aptidão. A partir da análise dessa aptidão de cada indivíduo, entra em ação um outro operador do algoritmo genético, este operador é conhecido como mecanismo de seleção, que é tendencioso para indivíduos mais aptos da população, ou seja, as melhores soluções têm mais chance de serem selecionadas. Este processo simula a “seleção natural” e “sobrevivência do mais apto” da já citada Teoria da Evolução (VALLADA e RUIZ, 2012). A Figura 1, ilustra o funcionamento do GA proposto neste trabalho e os operadores que foram considerados.

Figura 1 - Algoritmo Genético.

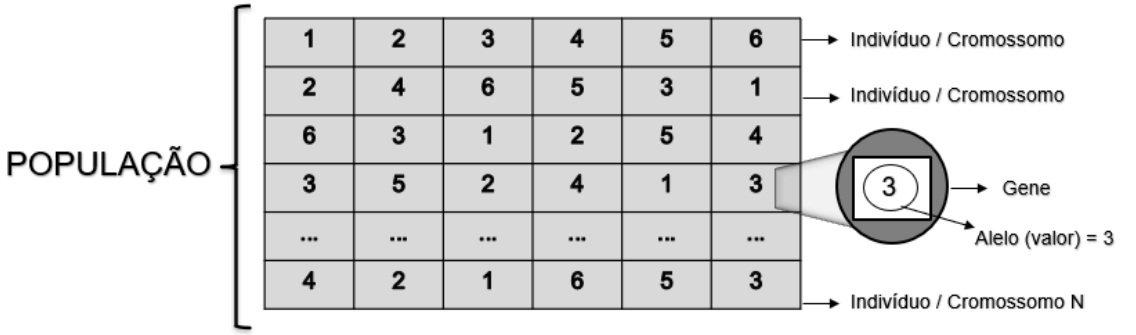


Fonte: Autoria própria (2022).

3.2.1.1 Representação da população

Pacheco (2017), explica que no GA cada indivíduo pode ser representado por uma cadeia de genes. Esses genes podem ser simbolizados em forma binária ou com números reais, a depender das características do problema que está sendo trabalhado. Nesta pesquisa, cada cromossomo deverá representar uma lista ordenada de *jobs* (pedidos de produção), que simboliza a sequência em que esses *jobs* deverão ser processados, ou seja, cada gene representa um pedido de produção em sua respectiva ordem. Na Figura 2, é ilustrada essa representação dos indivíduos e da população.

Figura 2 - Representação da População.



Fonte: Autoria própria (2022).

3.2.1.2 Função fitness

Asadzadeh (2015) explica em seu trabalho que no GA a aptidão dos indivíduos da população é calculada por meio da função *fitness*, ela calcula a qualidade de uma solução candidata para um determinado problema. Para essa finalidade, a função retorna um valor obtido através desta avaliação, valor esse que dependerá do objetivo do problema que está sendo analisado. No problema aqui estudado, a função *fitness* irá retornar a soma dos custos por antecipação e atraso das soluções candidatas, e por se tratar de um problema com o objetivo de minimização destes custos, as soluções que possuem uma melhor aptidão são as com menor valor *fitness*. Essa característica irá favorecer essas soluções com mais qualidade no decorrer de cada iteração do GA, fazendo com que elas sejam selecionadas para se perpetuar, gerando novas soluções. A cada nova geração da população, a função *fitness* é responsável por comparar se as novas soluções obtiveram melhora com relação à melhor solução global já encontrada, caso tenha melhor *fitness*, a solução candidata passa a ser a melhor solução global.

3.2.1.3 Operadores

Antes mesmo da implementação dos operadores do GA, o passo inicial a ser realizado é o procedimento de geração de uma população inicial para o problema, que pode ser construída de forma aleatória (gulosa) ou até mesmo com o uso de heurísticas mais tradicionais como as de busca local. Para isso, se faz necessário estabelecer o tamanho da população a ser gerada e o número de genes.

3.2.1.3.1 Seleção de reprodutores

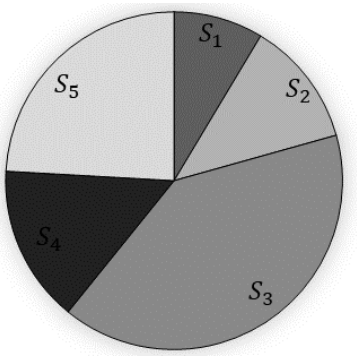
O processo de seleção é um princípio fundamental do GA, ele quem vai fazer com que, depois de diversas gerações, a população inicial de indivíduos venha a gerar indivíduos mais aptos. Esses métodos de seleção são desenvolvidos para priorizar indivíduos com melhores aptidões, mas não puramente estes, pois se faz necessário a manutenção da diversidade da população, o trabalho de Goldberg e Deb (1991) apresenta algumas variações desse mecanismo. Um mecanismo de eleição que é bastante utilizado é o da Roleta, onde os indivíduos são escolhidos por meio de um sorteio, em que cada cromossomo é representado na roleta de forma proporcional a sua aptidão. Assim, os com valores mais altos têm maior chance de serem sorteados. Contudo, pelo fato do problema aqui estudado se tratar de um problema que envolve custos, se faz necessário fazer uma representação de modo a proporcionar aos indivíduos com menor custo uma melhor aptidão, ou seja, uma probabilidade de escolha inversamente proporcional. Para isso é realizada uma adaptação na equação apresentada por Goldberg e Deb (1991) para o cálculo da probabilidade de seleção de um indivíduo pelo método de reprodução proporcional. Com isso, se f_i é o valor *fitness* do indivíduo i na população a probabilidade de ser selecionado é:

$$p_i = \frac{\frac{1}{f_i}}{\sum_{j=i}^{n} \frac{1}{f_i}} \quad (6)$$

Na Figura 3 é ilustrado o funcionamento do operador de seleção:

Figura 3 – Exemplo Operador de Seleção.

Indivíduo S_i	Fitness $f(S_i)$	Fitness Relativo
S_1	90	0,09
S_2	50	0,12
S_3	15	0,40
S_4	40	0,15
S_5	25	0,24



Fonte: Autoria própria (2022).

3.2.1.3.2 Operador de crossover

Outro operador do GA é o *crossover*, que tem o objetivo de gerar novos cromossomos por meio do cruzamento entre os indivíduos selecionados (pais) a partir do operador de seleção.

Esse cruzamento faz com que aconteça uma recombinação de características dos pais, proporcionando que as gerações futuras herdem essas características. Esse operador pode ser aplicado com o uso de uma taxa de probabilidade de *crossover*, e pode ser aplicado de diversas formas, como: *Crossover* de um ponto e *Crossover* de dois pontos.

A pesquisa de Vallada e Ruiz (2012), apresenta o *crossover* de um ponto, onde primeiro é selecionado aleatoriamente um ponto no intervalo de 0 a *n* (comprimento do cromossomo) e os genes no intervalo de 0 ao *n* selecionado são passados para o 1° filho a partir do 1° pai e para o 2° filho a partir do 2° pai. No 1° filho os genes que ainda se encontram vazios serão preenchidos percorrendo em ordem o material genético do 2° pai e atribuindo os valores genes que ainda não se encontram neste filho. Já no 2° filho acontece algo semelhante, porém se percorre o material genético do 1° pai para pegar os valores restantes.

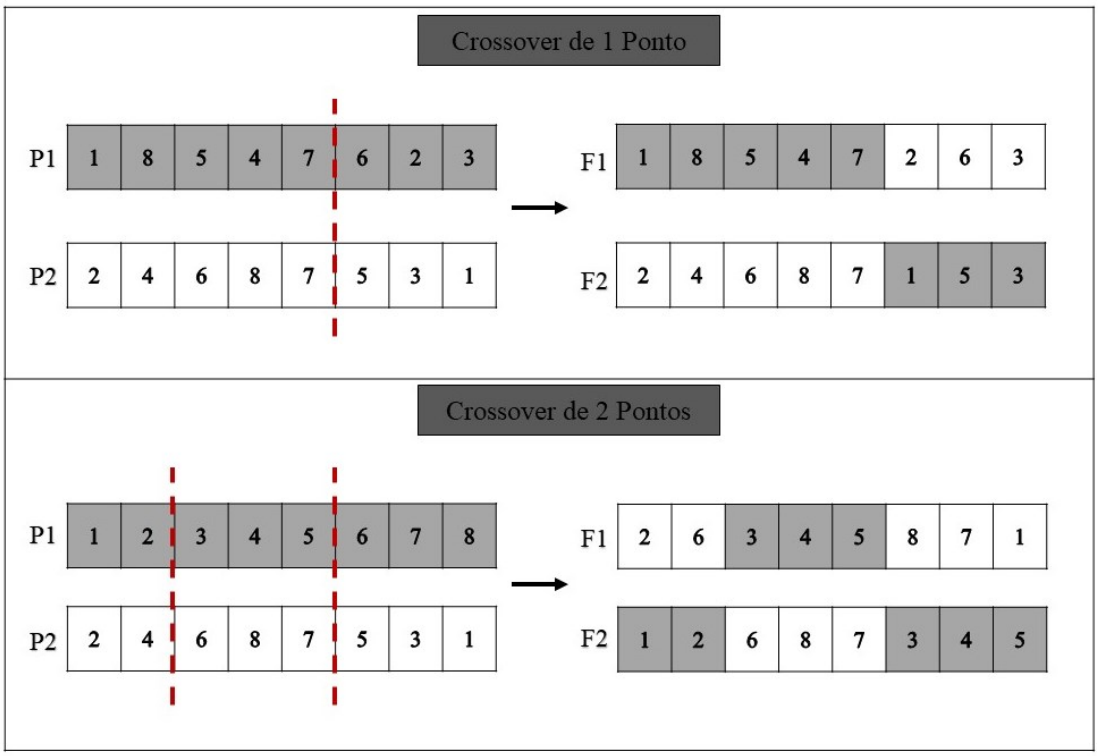
Já o *crossover* de dois pontos utilizado nesta pesquisa baseou-se em uma adaptação do *Order Crossover* como apresentado em Scofield (2002), onde os filhos são gerados escolhendo um pedaço do cromossomo e em seguida mantendo a sequência relativa dos *jobs* de um outro cromossomo. Primeiro são escolhidos 2 pontos de corte para os dois cromossomos pais. Após escolher os pontos de corte, os trechos são repassados para os filhos. Na geração dos filhos são usados os seguintes passos:

Para gerar o 1° filho, é preenchido os valores faltantes percorrendo o 2° pai para selecionar os valores em ordem, omitindo os valores que já estão no trecho do 1° filho.

Para gerar o 2° filho, é preenchido os valores faltantes percorrendo o 1° pai para selecionar os valores em ordem, omitindo os valores que já estão no segmento do 2° filho.

A Figura 4 ilustra como acontece ambos os processos de *crossover* descritos anteriormente:

Figura 4 - Operador de Crossover.



Fonte: Autoria própria (2022).

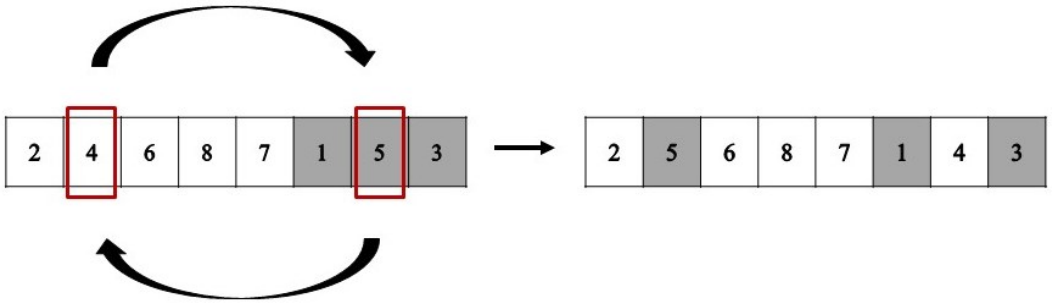
3.2.1.3.3 Operador de busca local

O operador de busca local tem como o objetivo melhorar a solução do problema por meio da alteração da solução atual, realizando uma ou mais trocas. Nesta pesquisa se utilizou mais precisamente o método Swap como apresentado em Subramanian, Batarra e Potts (2014), que consiste em realizar a troca de posição entre dois pedidos de produção levando-se em consideração uma probabilidade de ocorrer essa inversão.

Com isso, este estudo propõe uma hibridização de algoritmo genético com uma busca local, procedimento esse que pode ser encontrado com bastante frequência na literatura. Isso se explica pelo fato de tais métodos proporcionarem melhoras significativas na qualidade das soluções encontradas pelos algoritmos.

A Figura 5 exemplifica o funcionamento da heurística de busca local pelo método swap:

Figura 5 - Busca Local.



Fonte: Autoria própria (2022).

3.2.1.3.4 Operador de mutação

Tal operador é utilizado como estratégia importantíssima para que exista uma manutenção da diversidade da população, trocando entre si os valores de dois componentes (genes) de um certo indivíduo da população. Essa estratégia permite explorar diferentes regiões do espaço de busca, permitindo assim a fuga de mínimos ou máximos locais.

Pela característica do problema aqui tratado, esse operador de mutação pode ser colocado em pratica utilizando-se do mesmo princípio do swap usado na busca local e apresentado por Subramanian, Batarra e Potts (2014). Basta trocar dois genes (pedidos de produção) de posição que esta realizada a mutação. Esse procedimento foi ilustrado na Figura 5.

3.2.1.4 Atualização da população

Esse processo é responsável por atualizar a população corrente. Com isso, são inseridos na população os cromossomos que foram gerados nas etapas anteriores e os cromossomos com menor aptidão são removidos da população. Esse procedimento é conhecido como Elitismo e evita que a população piore com o passar das iterações, pois garante que sempre os melhores indivíduos sobrevivam para as próximas gerações, melhorando assim o mecanismo de memória do algoritmo. A partir daí, é realizada novamente a avaliação da população e o fluxo de execução do algoritmo segue até que seja atingida a condição estabelecida para parada.

3.2.1.5 Critério de parada

O critério de parada é um fator muito importante no GA, pois pode determinar a otimização e o tempo de execução do algoritmo. O algoritmo aqui proposto considerou um critério onde foi definido um número de gerações que o algoritmo deveria realizar, sendo este número igual a 5000 gerações.

3.3 TRABALHOS RELACIONADOS

Na literatura, podem ser encontrados diversos trabalhos relacionados com o JSS-ETC, como o FJSP, *Just-in-time Job-Shop Scheduling* (JIT-JSS) e Problema de *Job-Shop* com Janela de Tempo (JSS-TW, do inglês *Job Shop Scheduling with Time Windows*), dentre outros. Com isso, foi realizada uma vasta revisão de literatura, de modo a melhor compreender o estado da arte do problema, bem como as técnicas e abordagens de otimização emergentes na área. Inicialmente, esse levantamento teve seu foco na melhor compreensão dos métodos de otimização mais utilizados para resolução dos problemas relacionados com o JSS-ETC. Em seguida, foi realizado um mapeamento de trabalhos que usam o GA na sua resolução.

Para o problema de JSS biobjetivo que busca minimizar o tempo de fluxo médio ponderado e penalidades totais de antecipação e atraso (ET, do inglês *Earliness and Tardiness*), Tavakkoli-Moghaddam, Azarkish e Sadeghnejad-Barkousaraie (2011), apresentam um novo modelo matemático e um algoritmo multiobjetivo de otimização por enxame de partículas (PSO, do inglês *Particle Swarm Optimization*) combinado com busca de vizinhança variada (VNS, do inglês *Variable Neighborhood Search*), fazendo o uso de caractere de busca de dispersão (SS, *Scatter Search*) para selecionar novos enxames em cada iteração. O trabalho de Gao *et al.* (2016) trata de uma variante do JSS, o FJSP biobjetivo, onde existe uma combinação ponderada que almeja a minimização do *Makespan* e a média de ET. em sua obra, os autores propõem um algoritmo de busca de harmonia discreta (DHS, do inglês *Discrete Harmony Search*) e usam vários métodos de busca local para melhorar a capacidade de exploração local do algoritmo. Já Kianpour *et al.* (2021), propuseram para o JSS com tempo de processamento dinâmico e data de vencimento, uma aplicação de conceitos de valor agregado e custos total previsto na conclusão, integrando com o modelo de programação linear inteira mista (MIP, do inglês *Mixed Integer Linear Programming*) para projetar um sistema de programação automatizado adaptativo.

Com o intuito de resolver o JSS-TW, o trabalho de Huang e Yang (2008), utiliza um algoritmo de otimização por colônia de formigas (ACO, do inglês *Ant Colony Optimization*), que é baseado na experiência, exploração de novas soluções e melhoria das existentes, fazendo com que novas soluções ideais sejam geradas por autoaperfeiçoamento iterativo. Enquanto em Huang, Yang e Cheng (2013) é proposto um algoritmo ACO modificado, que é denomina otimização de colônia de formigas de dois feromônios (2PH-ACO), que adiciona um segundo grupo de feromônios para o sistema de formigas, a fim de resolver o problema de agendamento mais rapidamente.

Uma heurística híbrida que combina heurísticas de busca local e um algoritmo de busca evolutiva baseado em GA, é apresentada por M'Hallah (2007) para minimização do total de ET em um JSS de máquina única. As etapas iterativas do algoritmo evolutivo híbrido proposto, seguem um GA, onde a população inicial é gerada usando algoritmos gulosos construtivos, cada filho mutado usando escalada de colinas (HC, do inglês *Hill Climbing*) e os melhores indivíduos da geração melhorada usando recozimento simulado (SA, do inglês *Simulated Annealing*). Yazdani *et al.* (2017), introduzem uma formulação baseada em MIP e uma nova função objetivo, que é a soma dos critérios máximos de ET para o JSS. Enquanto Kramer e Subramanian (2019), desenvolveram em seu trabalho uma metaheurística de busca local eficiente de acordo com as características do problema ET, trazendo abordagens para problemas de máquinas paralelas que generalizam os existentes para máquina única. Além disso, foi fornecida uma biografia comentada sobre os problemas relacionados, apontando a abordagem utilizada em cada publicação e ainda as características dos problemas.

Kayvanfar, Manhdavi e Komaki (2013), apresentaram um modelo matemático, uma ILS e algoritmos baseados em regras de despacho de abordagem JIT, para o problema sequenciamento de máquina única para minimização de ET com tempo de processamento controlável. Schaller e Valente (2013) propuseram um GA para o minimizar o total de ET em uma permutação *flow-shop*, realizando em seguida testes para verificar se o método utilizado foi consistente para gerar boas soluções em comparação com outros já propostos. Enquanto em Schaller e Valente (2019), trabalharam com o problema de sequenciamento com permutação *flow-shop* para minimizar o total de ET com tempo ocioso não forçado permitido e realizaram testes com várias heurísticas de despacho simples já usadas em outros problemas semelhantes. Com isso, desenvolveram uma heurística híbrida a partir de três heurísticas de despacho: EDD, MDD e FV. Já Ahmadian, Salehipour e Cheng (2021), desenvolveram um algoritmo VNS para o JIT-JSS, a fim de solucionar um conjunto de 72 instâncias, que variam de 10 a 20 *jobs* e 20 a 200 operações,

Como visto nos trabalhos citados anteriormente, existem na literatura diversos problemas de JSS que lidam diretamente com a minimização de ET. Dentre tantas variantes, vamos abordar a seguir as pesquisas que tem como objetivo a minimização de custos/penalidades por ET, a fim de esmiuçar as técnicas utilizadas.

Nesse sentido, Feldmann e Biskup (2003) propuseram em seu trabalho três metaheurísticas para o problema de sequenciamento de máquina única (SMS, do inglês *Single Machine Scheduling*) com minimização de penalidades por antecipação e atraso (ETP, do inglês *Earliness and Tardiness Penalties*), uma de busca evolutiva, uma de SA e uma de aceitação de limiares (TA, do inglês *Threshold Accepting*). Os autores ainda desenvolveram uma nova variante de TA, realizando testes em 140 problemas com até 1000 *jobs*, e posteriormente, compararam a qualidade da solução. Para resolver o ETC, o trabalho de Beck e Refalo (2003), utiliza uma abordagem híbrida que se baseia em técnicas de programação de restrições (CP, do inglês *Constraint Programming*) e as técnicas de programação linear (LP, do inglês *Linear Programming*). Um algoritmo ILS foi proposto por Subramanian, Batarra e Potts (2014) para o problema de agendamento de atraso total ponderado de máquina única com tempos de *setup* dependentes da sequência (SM-WT, do inglês *Single Machine total Weighted Tardiness scheduling problem with sequence-dependent setup times*), sendo realizados ainda comparação de performance com outros algoritmos de metaheurísticas presentes na literatura.

Em Nogueira *et al.* (2014), foram propostas três heurísticas para resolver o problema de sequenciamento de máquinas paralelas não relacionadas com ETP, uma heurística GRASP (do inglês *Greedy Randomized Adaptive Search Procedure*) simples, uma heurística com procedimento de intensificação baseada em *Path Relinking* e heurística de busca local (ILS). Um algoritmo de *Branch-and-Bound* e testes com procedimentos heurísticos de pesquisa de vizinhança baseado em intercâmbio de pares adjacentes (API, do inglês *Adjacent Pairwise Interchanges*) é apresentado por Baker (2014) para o ETC. Enquanto Zambrano Rey *et al.*, propõem duas metaheurísticas para o JIT-JSS com ETC quadrático, sendo elas um GA e um PSO, e ainda duas abordagens para lidar com tempos de liberação de tarefas. Fazlollahtabar, Saidi-Mehrabad e Balakrishnan (2015) desenvolveram em sua pesquisa um modelo matemático para um problema de otimização de ETP aplicado a um múltiplo sistema automatizado de fabricação de veículos guiados. Em seguidas os autores ainda apresentaram uma busca heurística de duas etapas para encontrar soluções para o problema em questão. Já Ahmadian, Salehipour e Kovalyov (2020), estudaram o ETP para *open-shop* e desenvolveram uma heurística de relaxamento eficiente e posteriormente realizaram testes em 72 instâncias com até 200 operações, comprovando a eficiência do método proposto.

Na literatura é possível encontrar, ainda, inúmeras abordagens que foram utilizadas para outros problemas de *scheduling*. A seguir, são destacadas as principais pesquisas que fizeram o uso de abordagens baseadas em GA como método de otimização para esses problemas.

A pesquisa de Montoya-Torres, Gutierrez-Franco e Pirachican-Mayorga (2010) estudou o problema de sequenciamento de projetos com restrição de recursos (RCPS, do inglês *Resource-Constrained Project Scheduling*), apresentando um GA para o mesmo e propondo uma representação alternativa dos cromossomos, fazendo o uso de um modelo multi-array. Para o JSS, Asadzadeh e Zamanifar (2010) propõem uma abordagem em que a criação da população inicial e a paralelização do GA são realizadas de maneira baseada em agentes. Enquanto, Wang e Tang (2011) apresentaram um GA adaptativo melhorado (IAGA, do inglês *Improved Adaptive Genetic Algorithm*) para resolver o problema de *makespan* mínimo. Em seguida é mostrado os resultados computacionais que validam a eficácia do método proposto. Hosseinabad *et al.* (2017) investiga os efeitos da seleção de operadores de cruzamento e mutação em GA para resolver um problema de *scheduling* em ambiente *open-shop.*

É possível encontrar estudos de JSS com diversas aplicações, como no campo de eficiência energética. Nesse sentido, May *et al.* (2015) mostraram um novo GA capaz de considerar objetivos relacionados à produtividade e consumo de energia. Enquanto, Salido *et al.* (2016) desenvolveram um GA para o JSS em que máquinas podem consumir diferentes quantidades de energia para processar tarefas em diferentes taxas.

Existem ainda abordagens que buscam a hibridização do GA com outros métodos, a fim de melhorar a eficiência dos algoritmos. É isso que acontece na pesquisa de Wang *et al.* (2012) para o JSS, que traz um operador de busca local que pode melhorar muito a capacidade de busca local do GA. Os autores ainda apresentam alguns operadores genéticos para aumentar a diversidade da população: um operador de seleção mista baseado no valor de *fitness*, novos operadores de cruzamento baseado na máquina e operador de mutação baseado no caminho crítico. Asadzadeh (2015) também apresentou uma hibridização com técnicas de busca local para melhorar a eficácia do GA, aplicando em seguida em instâncias de *bechmark* para atestar o desempenho da abordagem. Outro trabalho que apresenta metodologias híbridas de GA é o de Kundakci e Kulak (2016), que objetiva minimizar o *makespan* num JSS dinâmico.

Os trabalhos que envolvem o FJSP também têm feito o uso de GA como estratégia de otimização. Nesse sentido, De Giovanni e Pezzella (2010) apresentaram um GA aprimorado para resolver o problema em que os trabalhos são processados por um sistema de vários sistemas de unidades de manufatura flexível (FMUs, do inglês *Flexible Manufacturing Units*). Na representação da solução, a codificação dos genes é estendida para incluir informações sobre

atribuição de *job-to-FMU*. Além dos tradicionais operadores de cruzamento e mutação, um novo operador de busca local é usado para melhorar as soluções disponíveis, refinando os indivíduos promissores de cada geração. Um algoritmo genético de integração e decomposição (DIGA, do inglês *Decomposition-Integration Genetic Algorithm*) eficiente foi desenvolvido por Lei (2010) para o FJSP com tempo de processamento difuso. Ele usa uma representação de duas *strings*, que é um método de decodificação eficaz para a população principal. Zhang, Gao e Shi (2011) propuseram um algoritmo com seleção global e local, gerando uma população inicial de alta qualidade. Os autores ainda usaram uma representação aprimorada dos cromossomos para uma melhor representação das soluções do problema.

Em Demir e Isleyen (2014) é apresentado, inicialmente, um novo modelo matemático para o FJSP com sobreposição nas operações, mas por se tratar de um problema NP-difícil, em seguida é proposto um GA para resolver o mesmo com maior eficiência computacional. A pesquisa de Liangxiao e Zhongjun (2015), apresenta um GA aprimorado para minimizar o *makespan*. O algoritmo adota um novo método de inicialização para melhorar a qualidade da população inicial e acelerar a velocidade de convergência do algoritmo. Um algoritmo de hibridiza o GA e busca tabu (TB, do inglês *Tabu Search*) é proposto por Li e Gao (2016) para resolver o FJSP, esse algoritmo proposto combina busca global e local, o que fez com que obtivesse melhorias significativas para a resolução do problema. Já Fang (2022) propõe para o FJSP, uma hibridização de GA com DHS, que permite melhorar a qualidade dos indivíduos pelo operador de elitismo.

É possível encontrar pesquisas que fornecem revisões robustas da literatura da área, que podem auxiliar no entendimento de determinados problemas e seus respectivos métodos de otimização adotados. É o que acontece com o FJSP, que possui diversos trabalhos reportados neste sentido. Amjad *et al.* (2018), por exemplo, apresentam uma revisão abrangente da literatura do FJSP que são resolvidos com o uso do GA, se ampliando a ilustrar as técnicas de GA híbridas usadas para solução do problema. Outra contribuição nesse sentido pode ser vista em Gao *et al.* (2019), que fornece uma revisão abrangente de inteligência de enxame e algoritmos evolutivos para o mesmo problema. Em seguida são apresentadas estratégias de codificação e decodificação usadas. E apresentando ainda um GA híbrido clássico e um algoritmo competitivo imperialista (ICA, do inglês *Imperialist Competitive Algorithm*) com VNS.

Podemos encontrar ainda trabalhos que fazem o uso de GA e hibridização para resolver os problemas de ET e suas variantes. Como o é visto em Yuce *et al.* (2017), que desenvolvem um algoritmo híbrido utilizando o GA e algoritmo de abelhas (BA, do inglês *Bees Algorithm*),

onde é usado operador genético na busca global do BA. Khanh Van e Van Hop (2021) realizaram uma atualização de capacidade para o MIP e propuseram uma combinação do GA com a heurística de sequência inicial baseada no critério ET de máquina paralela (ISETP, do inglês *Initial Sequence based on Earliness-Tardiness criterion on Parallel machine*).

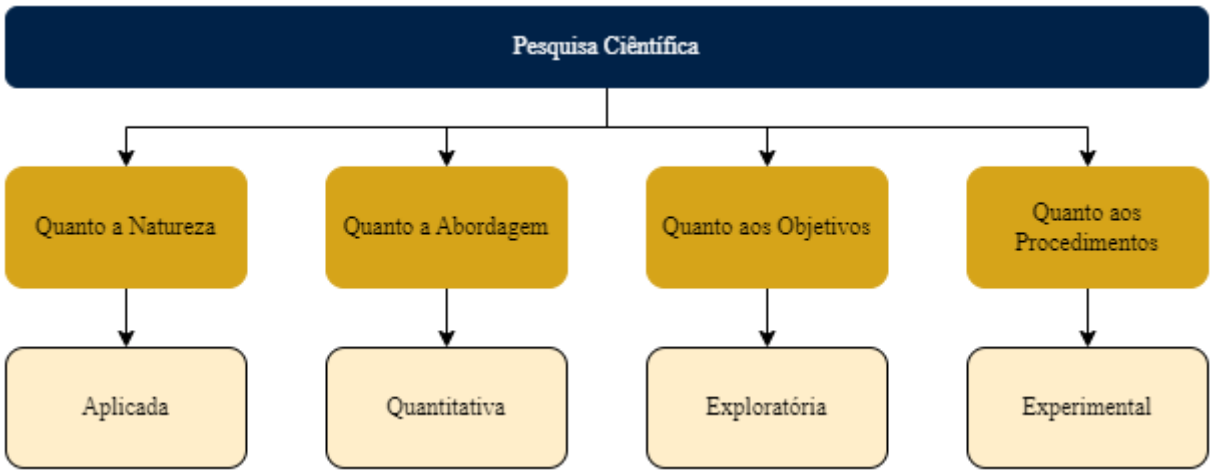
4 METODOLOGIA

Esta seção dedica-se a exposição dos métodos que foram empregados para o desenvolvimento deste trabalho. Inicialmente apresenta-se a caracterização da pesquisa e em seguida é apresentado como se deu a organização das atividades de desenvolvimento por meio de etapas.

4.1 CARACTERIZAÇÃO DA PESQUISA

De acordo com Libório e Terra (2015), o método cientifico pode ser definido como um conjunto de regras básicas empregadas em uma investigação cientifica, com o objetivo de obter resultados, de forma imparcial e confiável. Nesse sentido, para que se pudesse atingir os objetivos traçados, o presente estudo foi classificado com base em quatro critérios: quanto a sua natureza, quanto a seus objetivos, quanto a sua abordagem e quanto a seus procedimentos. A Figura 6, mostra de forma estruturada a caracterização deste estudo.

Figura 6 - Caracterização da Pesquisa Científica.



Fonte: Autoria própria (2022).

Esse estudo se caracteriza quanto a sua natureza como aplicada. A pesquisa aplicada interessa-se pela aplicação, utilização e consequências práticas dos conhecimentos. Destina-se a aplicar os conhecimentos científicos para solução dos mais variados problemas individuais ou coletivos. Concretiza-se por meio das “ciências aplicadas” e “tecnológicas” (ASSIS, 2009). Já de acordo com Gil (2019), a pesquisa aplicada abrange estudos elaborados com a finalidade de resolver problemas identificados no âmbito das sociedades em que os pesquisadores vivem.

Quando se analisa a pesquisa quanto a sua abordagem, para Porto (2011), a pesquisa quantitativa caracteriza-se pela investigação apoiada predominantemente em dados estatísticos e utiliza pesquisas anteriores e conhecimento teórico para escolha das variáveis. Nascimento e Cavalcante (2018) destacam que as pesquisas quantitativas possibilitam testar hipóteses, analisar a realidade de forma objetiva e generalizar os resultados pesquisados por meio de procedimentos estatísticos, avaliando os dados obtidos no processo da investigação, bem como utilizar recursos tecnológicos (computadores, softwares, planilhas eletrônicas) para auxiliar o pesquisador na descrição, analise interpretação e apresentação dos resultados da pesquisa.

Quanto aos objetivos este trabalho se caracteriza como exploratório. De acordo com Marconi e Lakatos (2017), pesquisas exploratórias são investigações de pesquisa empírica cujo objetivo é a formulação de questões ou de um problema, com tripla finalidade: descrever hipóteses; aumentar a familiaridade do pesquisador com ambiente, fato ou fenômeno para a realização de uma pesquisa futura mais precisa ou modificar e clarificar conceitos. os principais métodos empregados em estudos exploratórios, dentro da ótica objetivista, são: levantamentos em fontes secundárias, levantamentos de experiências (consultas a especialistas), observação e estudos de caso. (RÉVILLION, 2003).

O presente estudo se classifica quanto ao procedimento como experimental, que segundo Nascimento (2016) permite que o pesquisador estabeleça um objeto de estudo, selecione as variáveis que podem influenciá-lo, defina mecanismos e formas de controle e de observação dos efeitos causados pelas variáveis selecionadas sobre o objeto pesquisado.

4.2 ETAPAS DA PESQUISA

O desenvolvimento desta pesquisa foi organizado em duas fases e 5 etapas, a Figura 7 mostra como ficaram dispostas cada uma das etapas ao decorrer das fases:

Figura 7 - Organização do desenvolvimento da pesquisa.

Fonte: Autoria própria (2022).

A fase 1 do desenvolvimento do estudo englobou períodos de estudo e implementação e teve seu início com a etapa 1, onde foi realizado um estudo para aprofundar os conhecimentos sobre a linguagem de programação utilizada, que foi a Linguagem Julia. Em seguida, a fase 1 continuou por meio da etapa 2, que se designou a ampliação dos estudos, só que agora voltados para proporcionar uma melhor compreensão do problema aqui tratado e ainda sobre os métodos de otimização heurísticos e metaheurísticos, mais especificamente um aprofundamento sobre algoritmos evolucionários e o algoritmo genético que foi um dos métodos escolhidos para compor os algoritmos desenvolvidos neste estudo. E por último a realização da etapa 3 finalizou a fase 1, realizando-se a implementação das duas abordagens metaheurísticas propostas em código na linguagem de programação Julia. O código desenvolvido se dividiu em 3 partes, a primeira é responsável pela leitura e coleta das informações presentes nas instâncias que foram utilizadas para os testes, a segunda realiza a aplicação do método desenvolvido a partir dos dados coletados anteriormente, e a última parte fica responsável pela apresentação e registro dos resultados para realização de analises posteriormente.

Já a fase 2, englobou a aplicação do método e a análise dos resultados e teve seu ponto de partida na etapa 3, em que houve a aplicação, avaliação e ajustes necessários no método, com o intuito de maximizar o desempenho dos resultados dos experimentos. A fase 2 do desenvolvimento teve seu encerramento por meio da etapa 5, em que ocorreu a consolidação dos resultados obtidos nos testes, realização das devidas analises desses resultados e posteriormente a produção cientifica de todo o estudo realizado.

5 RESULTADOS

Esta seção é dedicada à apresentação dos testes computacionais desenvolvidos neste trabalho. Inicialmente, são descritos os problemas aqui aplicados e como são compostas as instâncias consideradas para os experimentos. Em seguida, são apresentados os parâmetros necessários para os algoritmos e quais os valores utilizados nos testes. Finalmente, os resultados computacionais são apresentados. Primeiro foram comparados o desempenho dos dois Algoritmos Genéticos hibridizados propostos. Posteriormente, houve a comparação com outros procedimentos existentes na literatura.

5.1 SEQUENCIAMENTO DA PRODUÇÃO COM CUSTOS DE ANTECIPAÇÃO E ATRASO

Com a ascensão dos princípios da produção enxuta, principalmente os conceitos do JIT, os problemas de sequenciamento envolvendo custos por antecipação ou atraso vem recebendo bastante atenção. Isso pode ser explicado pela importância do mesmo para que as empresas possam atingir os objetivos de acordo com essa filosofia. De acordo com o JIT, a antecipação e o atraso são considerados prejudiciais à lucratividade e, por isso, devem ser minimizados: o atraso causa perda da boa vontade do cliente e prejudica a reputação, bem como o atraso de pagamentos, enquanto a antecipação causa custos de manutenção de estoque e possível perda de qualidade do produto.

Como visto nas seções anteriores, o problema de sequenciamento da produção em máquina única vem sendo amplamente estudado. Neste problema, existem *n jobs* disponíveis no tempo zero para serem processados em uma única máquina e entregues em uma data de vencimento d_i, para cada *job i*. Cada *job i* requer exatamente uma operação e seu tempo de processamento p_i é conhecido. Se um *job i* é concluído antes da data de vencimento, sua antecipação é dada por $E_i = d_i - C_i$, onde C_i é o tempo de conclusão do *job i*. Por outro lado, se um trabalho *i* é concluído após a data desejada, seu atraso é dado por $T_i = C_i - d_i$. Cada *job i* tem sua própria multa unitária por antecipação α_i e multa por atraso unitário β_i. O objetivo do problema é obter um cronograma ótimo que minimize a soma das penalidades de antecipação e atraso (RONCONI; KAWAMURA, 2010).

A seguir será exposto para o problema em questão, uma formulação matemática que foi adaptada a partir do modelo apresentado por Ronconi e Kawamura (2010). Os parâmetros apresentados foram organizados a seguir:

d_i → Data de vencimento do pedido *i*;

α_i → Penalidade por unidade de tempo de antecipação para o pedido *i*;

β_i → Penalidade por unidade de tempo de atraso para o pedido *i*;

p_i → Tempo de processamento do pedido *i*;

R → Número muito grande;

Considere ainda as seguintes variáveis de decisão:

x_{ik} → Variável binaria igual a 1 se o pedido *i* é sequenciado antes do pedido *k* e 0 caso contrário;

C_i → Tempo de conclusão do pedido *i*;

E_i → Antecipação do pedido *i*;

T_i → Atraso do pedido *i*;

Desse modo, o modelo matemático que representa o problema tratado neste trabalho é apresentado a seguir:

$$Min \sum_{i=1}^{n} \alpha_i E_i + \sum_{i=1}^{n} \beta_i T_i \tag{7}$$

$$s.t. \quad T_i - E_i = C_i - d_i, \qquad i = 1, 2, \dots, n, \tag{8}$$

$$C_i \leq C_k - p_k + R(1 - x_{ik}), \qquad i = 1, 2, \dots, n-1, \qquad k = i+1, \dots, n, \tag{9}$$

$$C_k \leq C_i - p_i + Rx_{ik}, \qquad i = 1, 2, \dots, n-1, \qquad k = i+1, \dots, n, \tag{10}$$

$$C_i - p_i \geq 0, \qquad i = 1, 2, \dots, n, \tag{11}$$

$$T_i \geq 0, \qquad i = 1, 2, \dots, n, \tag{12}$$

$$E_i \geq 0, \qquad i = 1, 2, \dots, n, \tag{13}$$

$$x_{ik} \in \{0,1\}, \qquad i = 1,2,\dots,n-1, \qquad k = i+1,\dots,n. \tag{14}$$

Na formulação acima, (7) representa a função objetivo que consiste em minimizar a soma de penalidades por atraso e antecipação. O atraso e a antecipação podem ser calculados por meio de (8). As restrições (9) e (10) indicam o tempo de conclusão de cada pedido. A restrição (11) assegura que o tempo inicial de cada pedido i não negativo. O conjunto de restrições (12) e (13) define a não negatividade das variáveis T_i e E_i, enquanto a restrição (14) define a variável x_{ik} como binária.

5.2 CONFIGURAÇÃO EXPERIMENTAL

Os algoritmos propostos nesta pesquisa, foram testados em dois diferentes problemas de sequenciamento da produção. No primeiro experimento, as instâncias foram geradas de acordo com as instâncias presentes em OR-Library, para problema de sequenciamento com data de vencimento comum (CDD, do inglês *Common Due Date scheduling*). Já no segundo experimento foram usadas instâncias de *bechmark* também presentes em OR-Library, a partir de Tanaka, Fujikuma e Araki (2009), para o problema de sequenciamento com data de vencimento diferente (DDD, do inglês *Different Due Date scheduling)* para cada *job*.

Para o CDD, são considerados instâncias com seis diferentes números de *jobs* $n \in \{40, 20, 50, 100, 200, 500\}$ e quatro fatores restritivos $h \in \{0.2, 0.4, 0.6, 0.8\}$. O fator h representa o quanto emperrada a linha de produção fica no início do cronograma, sendo utilizado para encontrar a data de vencimento comum, de acordo com a expressão (15): $d = h\sum_{i=1}^{n} p_i$, onde p_i representa o tempo de processamento do *job* i. As instâncias usadas neste experimento consideram como tempo de processamento, inteiros com distribuição uniforme entre [1, 20], penalidades por antecipação entre [1,10] e penalidades por atraso entre [1, 15]. Cada arquivo disponibilizado em OR-Library, contém um quantitativo de 10 instâncias, que quando são avaliadas considerando os diferentes n de *jobs* e diferentes h, geram um total de 240 instâncias experimentadas para esse problema.

Já para o DDD, são consideradas instâncias com três diferentes números de *jobs* $n \in \{40, 50, 100\}$, e tempos de processamento entre [1, 100]. Essas instâncias ainda consideram as penalidades por antecipação e atraso no intervalo de [1, 10] para ambos os casos. Para cada grupo com diferente n de *job*, existem 125 instâncias, o que gera um total de 375 instâncias experimentadas para este problema.

Antes da realização dos testes dos algoritmos propostos no conjunto de instâncias de ambos os problemas citados, foram definidos os parâmetros necessários para conduzir esses experimentos. No Quadro 2, são mostrados os parâmetros: número de gerações $(Generations)$, que é usado como critério de parada do algoritmo, tamanho da população (P_{Size}), probabilidade de cruzamento (P_C), probabilidade de mutação (P_M), probabilidade de mutação por número de iterações sem encontrar solução melhor $(P_{M/I})$, probabilidade de busca local (P_{LS}), número de iteração sem melhora $(Iteration)$, para então aplicar mutação $(P_{M/I})$ e ainda número de indivíduos no Elitismo $(Elitism)$.

Quadro 2 - Valores dos parâmetros.

Parâmetro	Valor testado
$(Generations)$ Critério de parada	5000
(P_{Size}) Tamanho da população	20
(P_C) Probabilidade de cruzamento	0.9
(P_M) Probabilidade de mutação	0.04
$(P_{M/I})$ Probabilidade de mutação por falta de melhoria	0.5
(P_{LS}) Probabilidade de busca local	0.2
$(Iteration)$ Iterações sem melhora para aplicar $(P_{M/I})$	$0.2(Generations)$
$(Elitism)$ Indivíduos no Elitismo	0.2 (P_{Size})

Fonte: Autoria própria (2022).

5.3 RESULTADOS PRELIMINARES

Os algoritmos de otimização desenvolvidos foram implementados utilizando-se Julia versão 1.7.3, que é uma linguagem de programação dinâmica de alto nível que vem sendo muito utilizada ultimamente por possuir um ótimo desempenho numérico e científico. Todos os experimentos foram realizados em um computador de 64 bits, com processador Intel(R) Core (TM) i5-7200U de 2.5 GHz e com 8 GB de memória RAM.

Primeiro, analisou-se o desempenho das duas heurísticas propostas, em um conjunto de instâncias para o *Common Due Date scheduling*. Os resultados preliminares dos experimentos encontram-se no Quadro 3, onde estão representados na coluna 1 o número de *Jobs* da instância estudada, na 2 o valor do parâmetro h, nas colunas 3, 4 e 5 encontram-se, sucessivamente, os valores mínimo, médio e máximo encontrados pelo algoritmo HGA_1, enquanto na colona 6 pode-se visualizar o tempo computacional de execução em segundos. As colunas seguintes apresentam os resultados dos mesmos itens analisados para HGA_1, só que agora para o algoritmo HGA_2.

Quadro 3 - Desempenho dos algoritmos no CDD.

		HGA_1				HGA_2			
		Fitness			CPU Time (s)	Fitness			CPU Time (s)
n	h	Min	Average	Max	Average	Min	Average	Max	Average
10	0,2	1001	1651	2139	0,94	1001	1651	2139	1,96
	0,4	619	961	1374	0,91	619	961	1374	1,94
	0,6	550	779	1083	0,93	550	779	1083	1,97
	0,8	560	969	1432	0,94	560	969	1432	1,94
20	0,2	3412	6161	10349	1,35	3408	6170	10352	5,26
	0,4	2090	3621	6191	1,41	2091	3636	6209	5,16
	0,6	1710	2904	4137	1,39	1729	2924	4160	5,18
	0,8	2125	3718	5567	1,37	2127	3715	5557	5,19
50	0,2	28671	36192	44648	2,73	28381	36354	44756	27,34
	0,4	17655	21251	26080	2,74	17819	21461	26527	27,53
	0,6	14788	16827	23290	2,74	14686	17032	23353	27,86
	0,8	17084	21938	31391	2,76	17401	22160	31486	27,73
100	0,2	122055	137946	166280	5,32	122979	138023	165599	110,83
	0,4	77065	84047	100846	5,18	75236	84056	101756	111,23
	0,6	62968	72459	90174	5,29	61702	72204	90471	110,37
	0,8	73778	95287	127103	5,44	76394	95244	127463	111,78
200	0,2	487522	543150	611538	10,86	493688	546044	612755	516,84
	0,4	310218	336437	378761	11,05	307947	337790	381489	519,20
	0,6	258892	290443	321970	11,08	267954	295086	324137	521,15
	0,8	344174	388476	438290	11,40	342647	390263	436429	523,98
500	0,2	3079966	3434003	3680198	28,99	3045332	3395916	3665630	4760,95
	0,4	1889502	2126900	2265862	29,82	1845837	2103469	2259273	4718,87
	0,6	1712169	1898729	2036687	31,01	1673721	1864544	2026031	4741,85
	0,8	2241324	2519253	2681528	31,04	2280477	2510578	2686596	4757,90

Fonte: Autoria própria (2022).

Observa-se que no geral o algoritmo HGA_1 tem um tempo de execução mais rápido em comparação com o HGA_2, independentemente do grupo de instâncias analisado. Isso pode ser explicado ao se analisar as características dos operadores genéticos presentes em cada um dos algoritmos utilizados. Enquanto o HGA_1 utiliza do *crossover* de 1 ponto, o HGA_2 faz do uso do *crossover* de 2 pontos, que acaba demandando bem mais iterações para preencher os genes que ainda não receberam material genético do segundo pai, sem que ocorra a repetição dos genes que já foram herdados do primeiro pai.

Posteriormente a realização dos testes com os algoritmos no problema CDD, foram feitos novos experimentos com os mesmos algoritmos no problema *Diferent Due Date scheduling*. Os resultados preliminares dos experimentos do DDD encontram-se no Quadro 4, onde estão representados seguindo o mesmo padrão de organização visto no Quadro 3, exceto a presença do fator restritivo h, que não foi utilizado nos testes do DDD. Com isso, foram apresentados os valores da função objetivo (*fitness*) e tempo computacional para os algoritmos HGA_1 e HGA_2, em cada um dos grupos de instâncias do problema em questão.

Quadro 4 - Desempenho dos algoritmos no DDD.

	HGA_1				HGA_2			
	Fitness			CPU Time (s)	Fitness			CPU Time (s)
n	Min	Average	Max	Average	Min	Average	Max	Average
40	16433	58409	160070	3,70	16423	58170	160389	24,91
50	26279	86131	241681	3,85	26354	85485	243065	38,10
100	129372	370533	924691	6,51	126912	374157	925917	157,04

Fonte: Autoria própria (2022).

É possível notar que o desempenho melhor do HGA_1 com relação ao tempo computacional de execução se manteve, quando comparado ao HGA_2. Porém, quando se analisa o *fitness* médio obtido por cada grupo de instância, observa-se que o HGA_2 obteve melhor desempenho nas instâncias dos grupos com 40 e 50 *jobs*, enquanto o HGA_1 foi melhor nas instâncias com 100 *jobs*.

5.4 BENCHMARK COM A LITERATURA DO CDD

Nesta seção, são apresentadas as comparações realizadas entre os procedimentos metaheurísticos aqui propostos com os trabalhos presentes na literatura, sendo confrontados os valores obtidos para a função objetivo em cada grupo de instância do *Common Due Date Scheduling*. Os valores da função objetivo (*fitness*) que serviram para o *benchmark*, foram obtidos a partir do trabalho de Biskup e Feldmann (2001), que usam procedimentos heurísticos. O Quadro 5 fornece os valores de Gap mínimo, médio e máximo obtidos por cada um dos algoritmos propostos (F_{AP}) em relação ao valor de *Benchmark* (F_{BF}) Biskup e Feldmann (2001). Esses valores foram calculados da seguinte forma:

$$GAP = \frac{(F_{AP})-(F_{BF})}{(F_{BF})} \quad (15)$$

O Quadro 5 mostra o Gap para cada algoritmo proposto, sendo mostrado o valor mínimo, médio e máximo para cada grupo de instância analisado.

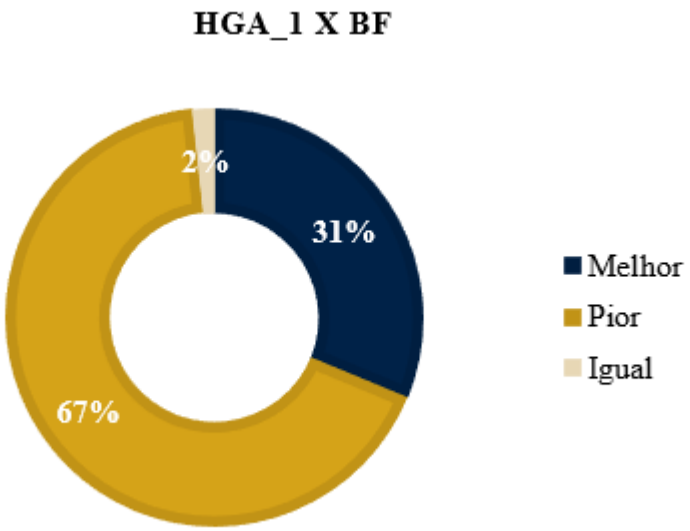
Quadro 5 - Gap entre valores obtidos e os de Biskup e Feldmann (2001).

n	h	HGA_1			HGA_2		
		GAP_{min} %	GAP_{avg} %	GAP_{max} %	GAP_{min} %	GAP_{avg} %	GAP_{max} %
10	0,2	-12,17%	-7,71%	-7,71%	-12,17%	-7,71%	-3,36%
	0,4	-9,15%	-3,22%	-3,22%	-9,15%	-3,22%	2,11%
	0,6	-1,72%	6,76%	6,76%	-1,72%	6,76%	42,60%
	0,8	3,70%	36,79%	36,79%	3,70%	36,79%	132,85%
20	0,2	-9,85%	-4,16%	-4,16%	-9,50%	-4,00%	-0,45%
	0,4	-62,15%	-7,94%	-7,94%	-62,04%	-7,57%	1,24%
	0,6	-1,04%	2,66%	2,66%	-0,36%	3,39%	9,11%
	0,8	12,36%	36,21%	36,21%	12,46%	36,10%	61,90%
50	0,2	-7,09%	-3,76%	-3,76%	-6,52%	-3,38%	1,71%
	0,4	-6,01%	-0,83%	-0,83%	-5,31%	0,14%	4,43%
	0,6	2,78%	5,47%	5,47%	3,20%	6,78%	11,23%
	0,8	22,45%	37,74%	37,74%	24,72%	39,23%	51,45%
100	0,2	-5,54%	-2,27%	-2,27%	-4,90%	-2,19%	0,43%
	0,4	-4,61%	2,40%	2,40%	-3,96%	2,34%	6,71%
	0,6	8,90%	11,48%	11,48%	8,50%	11,02%	13,46%
	0,8	33,47%	46,16%	46,16%	33,06%	46,08%	57,67%
200	0,2	-3,01%	-0,07%	-0,07%	-1,76%	0,49%	2,93%
	0,4	2,56%	6,83%	6,83%	3,60%	7,23%	10,26%
	0,6	8,31%	13,31%	13,31%	9,04%	15,14%	18,79%
	0,8	38,72%	51,58%	51,58%	40,20%	52,27%	60,42%
500	0,2	0,00%	2,57%	2,57%	-1,13%	1,42%	3,09%
	0,4	7,85%	11,00%	11,00%	6,27%	9,75%	11,85%
	0,6	17,96%	19,84%	19,84%	16,15%	17,69%	18,84%
	0,8	52,64%	58,95%	58,95%	55,23%	58,45%	63,32%

Fonte: Autoria própria (2022).

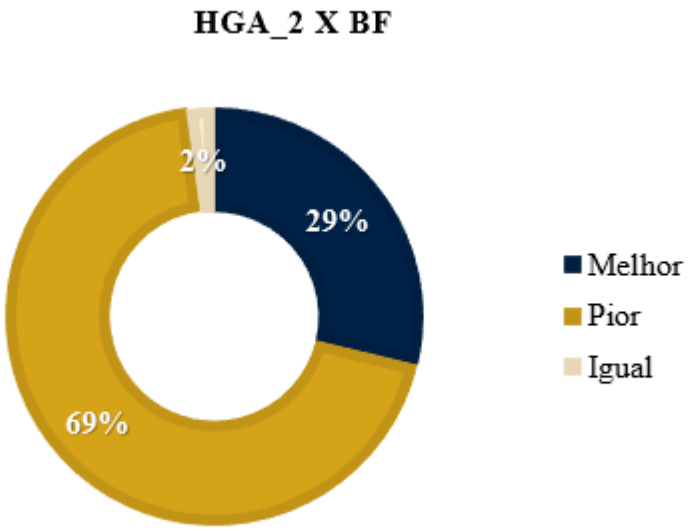
Ao se analisar os resultados apresentados no Quadro 5, pode-se notar que os algoritmos HGA_1 e HGA_2, obtiveram resultados satisfatórios em diversas instâncias, em alguns casos até superando os valores obtidos por Biskup e Feldmann (2001). Os Gráficos 1 e 2 apresentam graficamente o resultado do comparativo realizado, evidenciando em termos percentuais, em quantas instâncias os algoritmos propostos obtiveram um desempenho melhor, igual ou inferior aos presentes no trabalho usado na comparação.

Gráfico 1 - Desempenho HGA_1 comparado a Biskup e Feldmann (2001).



Fonte: Autoria própria (2022).

Gráfico 2 - Desempenho HGA_2 comparado a Biskup e Feldmann (2001).

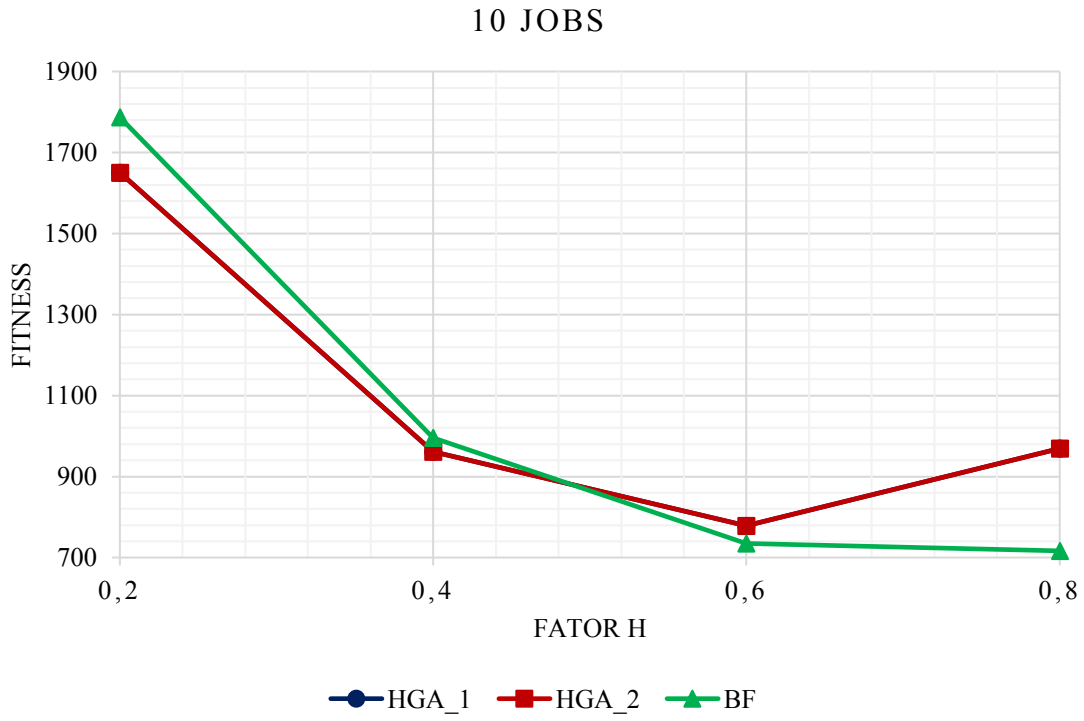


Fonte: Autoria própria (2022).

Os resultados apresentados no Gráfico 1, mostram que de todas as instâncias usadas no presente estudo, o HGA_1 obteve resultados melhores em comparação a Biskup e Feldmann (2001) em 31% delas, resultados iguais em 2% e piores em 67%. Enquanto o Gráfico 2, mostra que o HGA_2 alcançou resultados melhores em 29% das instâncias, iguais em 2% e piores em 69%, tendo assim uma leve queda no desempenho em comparação com os obtidos pelo HGA_1.

Com o intuito de realizar uma análise mais detalhada dos resultados, compreendendo melhor em quais grupos de instância os algoritmos obtiveram melhor desempenho, foram desenvolvidos mais alguns gráficos que permitem uma melhor interpretação dos resultados. Esses Gráficos possibilitam que se visualize o comportamento de cada um dos algoritmos por grupo de instância, sendo capaz de enxergar como o *Fitness* (função objetivo) obtido por determinado algoritmo é impactado pelo aumento do fator restritivo *h*, que impacta diretamente no aumento da data de entrega dos pedidos (*jobs*). Os Gráficos 3, 4, 5, 6, 7 e 8 que apresentam o *benchmark* entre os algoritmos desenvolvidos e o de Biskup e Feldmann (2001) são apresentados a seguir.

Gráfico 3 - Benchmark em instâncias de 10 *jobs*.



Fonte: Autoria própria (2022).

Gráfico 4 - Benchmark em instâncias de 20 *jobs*.

Fonte: Autoria própria (2022).

Gráfico 5 - Benchmark em instâncias de 50 *jobs*.

Fonte: Autoria própria (2022).

Gráfico 6 - Benchmark em instâncias de 100 *jobs*.

Fonte: Autoria própria (2022).

Gráfico 7 - Benchmark em instâncias de 200 *jobs*.

Fonte: Autoria própria (2022).

Gráfico 8 - Benchmark em instâncias de 500 *jobs*.

Fonte: Autoria própria (2022).

Ao observar os gráficos acima expostos é possível constatar que as metaheurísticas propostas conseguem ser mais eficientes principalmente nas instâncias de até 100 *jobs* com contenham o fator restritivo *h* igual a 0.2 e 0.4. Nas instâncias maiores, ou seja, de 200 e 500 *job*, os algoritmos HGA_1 e HGA_2 perdem desempenho em comparação com Biskup e Feldmann (2001). Quando se analisa as instâncias com fator *h* igual a 0.6 e 0.8, independente do número de *jobs*, o algoritmo de Biskup e Feldmann (2001) obtém resultados melhores.

É possível notar ainda que à medida que o fator *h* aumenta e o consequentemente o prazo de entrega dos pedidos aumenta, os resultados obtidos pelo trabalho de Biskup e Feldmann (2001) melhoram, apresentando o tendencia de decréscimo na função objetivo. Já os algoritmos HGA_1 e HGA_2, mostraram resultados piores quando o fator *h* passou de 0.6 para 0.8. Esse fenômeno pode ser facilmente explicado pelo fato dos trabalhos que são precursores no CDD, considerarem soluções em que o primeiro pedido (*job*) a ser produzido, não necessita obrigatoriamente iniciar no instante zero, enquanto o HGA_1 e HGA_2 obrigam que isso aconteça. Essa restrição acaba fazendo com que à medida que se tenha um prazo maior para entrega, os pedidos iniciais gerem mais multa por antecipação, impactando negativamente na função objetivo.

6 CONSIDERAÇÕES FINAIS

Este trabalho abordou e apresentou duas metaheurísticas capazes de resolver problemas de *scheduling* que são relacionados com a filosofia *Just-in-Time*, mais especificamente problemas que envolvem custos por antecipação e atrasos na produção. Para tal, foi desenvolvido uma abordagem híbrida entre algoritmos genéticos e mecanismos de busca local sendo implementado utilizando-se da linguagem Julia.

Os algoritmos propostos contribuíram de forma eficaz no alcance de bons resultados, visto que ambos apresentaram desempenhos de qualidade considerável na obtenção do sequenciamento da produção, considerando os objetivos de minimização de custos por antecipação e atraso nos dois problemas em que foram realizados testes. Além disso, a abordagem desenvolvida conseguiu alcançar soluções eficientes em um tempo computacional satisfatório.

Adicionalmente, em termos de comparação com outros métodos propostos na literatura, os experimentos computacionais evidenciaram o bom desempenho das metaheurísticas propostas, sendo apresentado o *benchmark* dos resultados obtidos com os dos principais métodos existentes para a resolução da problemática estudada. Foi possível constatar que o desempenho das abordagens propostas apresentou uma eficiência significativa nas instâncias de porte menor, principalmente na problemática que considera datas de entrega em comum.

Tais testes computacionais demonstraram, ainda, que ambos os algoritmos obtiveram resultados superiores aos reportados na literatura de *benchmark* em alguns dos parâmetros, especialmente quando consideradas instâncias de até 100 pedidos de produção, o que não acontece em instâncias com quantidade de pedidos superior. Com isso, os resultados permitem ponderar que ainda há margem para melhorias com relação às estratégias utilizadas nos algoritmos em pesquisas futuras, para que estas instâncias com número maior de *jobs,* possam apresentar resultados mais satisfatórios.

Para um melhor direcionamento de pesquisas futuras, propõem-se que sejam considerados métodos para resolução de problemas de *scheduling* envolvendo antecipação e atraso (ETC) que considerem o cronograma de manutenção preventiva das máquinas, temática essa proposta recentemente na literatura.

REFERÊNCIAS

AHMADIAN, Mohammad Mahdi.; SALEHIPOUR, Amir.; CHENG, T. C. E. A meta-heuristic to solve the just-in-time job-shop scheduling problem. **European Journal of Operational Research**, v. 288, n. 1, p. 14-29, 2021.

AHMADIAN, Mohammad Mahdi.; SALEHIPOUR, Amir.; KOVALYOV, Mikhail. An efficient relax-and-solve heuristic for open-shop scheduling problem to minimize total weighted earliness-tardiness. **Available at SSRN** 3601396, 2020.

AMJAD, Muhammad Kamal *et al.* Recent research trends in genetic algorithm based flexible job shop scheduling problems. **Mathematical Problems in Engineering**, v. 2018, 2018.

ASADZADEH, Leila. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. **Computers & Industrial Engineering**, v. 85, p. 376-383, 2015.

ASADZADEH, Leila; ZAMANIFAR, Kamran. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. **Mathematical and Computer Modelling**, v. 52, n. 11-12, p. 1957-1965, 2010.

ASSIS, Maria Cristina de. **Metodologia do Trabalho Científico.** Faculdade do Sertão (UESSBA) – Pedagogia. 2013. Disponível em: https://www.docsity.com/pt/por-maria-cristina-de-assis-metodologia-do-trabalho-cientifico/4863932/.

BAKER, Kenneth R. Minimizing earliness and tardiness costs in stochastic scheduling. **European Journal of Operational Research**, v. 236, n. 2, p. 445-452, 2014.

BECK, J. Christopher.; REFALO, Philippe. A hybrid approach to scheduling with earliness and tardiness costs. **Annals of Operations Research**, v. 118, n. 1, p. 49-71, 2003.

BISKUP, Dirk.; FELDMANN, Martin. Benchmarks para agendamento em uma única máquina em relação a datas de vencimento comuns restritivas e irrestritas. **Computers & Operations Research** , v. 28, n. 8, pág. 787-801, 2001.

BUETTGEN, John Jackson. **Planejamento e Controle da Produção**. Indaial: Uniasselvi, 2011.

DE CM NOGUEIRA, João Paulo *et al*. Hybrid GRASP heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. **Electronic Notes in Theoretical Computer Science**, v. 302, p. 53-72, 2014.

DE GIOVANNI, Luigi.; PEZZELLA, Ferdinando. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. **European journal of operational research**, v. 200, n. 2, p. 395-408, 2010.

DEMIR, Yunus.; İŞLEYEN, Selçuk Kürşat. An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations. **International Journal of Production Research**, v. 52, n. 13, p. 3905-3921, 2014.

DO NASCIMENTO, Leandra Fernandes.; CAVALCANTE, Maria Marina Dias. Abordagem quantitativa na pesquisa em educação: investigações no cotidiano escolar. **Revista Tempos e Espaços em Educação**, v. 11, n. 25, p. 9, 2018.

FANG, Junfeng. An Effective Hybrid Multiobjective Flexible Job Shop Scheduling Problem Based on Improved Genetic Algorithm. **Scientific Programming**, v. 2022, 2022.

FAZLOLLAHTABAR, Hamed.; SAIDI-MEHRABAD, Mohammad.; BALAKRISHNAN, Jaydeep. Mathematical optimization for earliness/tardiness minimization in a multiple automated guided vehicle manufacturing system via integrated heuristic algorithms. **Robotics and Autonomous Systems**, v. 72, p. 131-138, 2015.

FELDMANN, Martin.; BISKUP, Dirk. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. **Computers & Industrial Engineering**, v. 44, n. 2, p. 307-323, 2003.

FREITAS, Diogo Machado de. **Geração Evolucionária de Heurísticas para Localização de Defeitos de Software**. 2018. 90 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás, Goiânia, 2018. Disponível em: https://repositorio.bc.ufg.br/tede/bitstream/tede/9010/5/Disserta%C3%A7%C3%A3o%20-%20Diogo%20Machado%20de%20Freitas%20-%202018.pdf. Acesso em: 06 out. 2022.

GAO, Kaizhou. *et al*. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. **IEEE/CAA Journal of Automatica Sinica**, v. 6, n. 4, p. 904-916, 2019.

GAO, Kai-Zhou. *et al.* Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. **Journal of Intelligent Manufacturing**, v. 27, n. 2, p. 363-374, 2016.

GIL. Antonio Carlos. **Como Elaborar Projetos de Pesquisas.** 6. Ed. São Paulo: Atlas, 2019.

GOLDBERG, David E.; DEB, Kalyanmoy. Uma análise comparativa de esquemas de seleção usados em algoritmos genéticos. In: **Fundamentos de Algoritmos Genéticos**. Elsevier, 1991. pág. 69-93.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems:** An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor: University of Michigan Press, 1975.

HUANG, Rong-Hwa.; YANG, Chang-Lin. Ant colony system for job shop scheduling with time windows. **The International Journal of Advanced Manufacturing Technology**, v. 39, n. 1, p. 151-157, 2008.

HUANG, Rong-Hwa; YANG, Chang-Lin; CHENG, Wei-Che. Flexible job shop scheduling with due window—a two-pheromone ant colony approach. **International Journal of Production Economics**, v. 141, n. 2, p. 685-697, 2013.

JIANG, Zengqiang. *et al*. A evolução da programação da produção da Indústria 3.0 até a Indústria 4.0. **International Journal of Production Research**, v. 60, n. 11, pág. 3534-3554, 2022.

KAYVANFAR, Vahid.; MAHDAVI, Iraj.; KOMAKI, GH M. Single machine scheduling with controllable processing times to minimize total tardiness and earliness. **Computers & Industrial Engineering**, v. 65, n. 1, p. 166-175, 2013.

KHANH VAN, Bui; VAN HOP, Nguyen. Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness-tardiness. **Journal of Industrial and Production Engineering**, v. 38, n. 1, p. 18-28, 2021.

KIANPOUR, Parsa. et al. Automated job shop scheduling with dynamic processing times and due dates using project management and industry 4.0. **Journal of Industrial and Production Engineering**, v. 38, n. 7, p. 485-498, 2021.

KRAMER, Arthur; SUBRAMANIAN, Anand. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems. **Journal of Scheduling**, v. 22, n. 1, p. 21-57, 2019.

KUNDAKCI, Nilsen.; KULAK, Osman. Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. **Computers & Industrial Engineering**, v. 96, p. 31-51, 2016.

LEI, Deming. A genetic algorithm for flexible job shop scheduling with fuzzy processing time. International **Journal of Production Research**, v. 48, n. 10, p. 2995-3013, 2010.

LI, Xinyu; GAO, Liang. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. **International Journal of Production Economics**, v. 174, p. 93-110, 2016.

LIANGXIAO, Jiang.; ZHONGJUN, Du. An improved genetic algorithm for flexible job shop scheduling problem. *In:* 2015 2nd **International Conference on Information Science and Control Engineering**. IEEE, 2015. p. 127-131.

LIBÓRIO, Daisy.; TERRA, Lucimara. **Metodologia científica**. São Paulo: Laureate, 2015.

M'HALLAH, Rym. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. **Computers & Operations Research**, v. 34, n. 10, p. 3126-3142, 2007.

MARCONI, Marina de Andrade.; LAKATOS, Eva Maria. **Técnicas de Pesquisa.** 8. Ed. São Paulo: Atlas, 2017.

MAY, Gökan. *et al.* Multi-objective genetic algorithm for energy-efficient job shop scheduling. International **Journal of Production Research**, v. 53, n. 23, p. 7071-7089, 2015.

MONTOYA-TORRES, Jairo R.; GUTIERREZ-FRANCO, Edgar; PIRACHICÁN-MAYORGA, Carolina. Project scheduling with limited resources using a genetic algorithm. **International Journal of Project Management**, v. 28, n. 6, p. 619-628, 2010.

NASCIMENTO, Francisco Paulo do. **Classificação da Pesquisa. Natureza, método ou abordagem metodológica, objetivos e procedimentos.** Metodologia da Pesquisa Científica: teoria e prática–como elaborar TCC. Brasília: Thesaurus, 2016.

PINEDO, M. L. **Planning and Scheduling in Manufacturing and Services**. New York: Springer, 2005.

PORTO, Geciane Silveira. **Pesquisa quantitativa.** São Paulo: USP, 2011.

RAHMANI HOSSEINABADI, Ali Asghar et al. Extended genetic algorithm for solving open-shop scheduling problem. **Soft computing**, v. 23, n. 13, p. 5099-5116, 2019.

RÉVILLION, Anya Sartori Piatnicki. A utilização de pesquisas exploratórias na área de marketing. **Revista Interdisciplinar de Marketing**, v. 2, n. 2, p. 21-37, 2003.

RONCONI, Débora P.; KAWAMURA, Márcio S. O problema de escalonamento de antecipação e atraso de uma única máquina: limites inferiores e algoritmo branch-and-bound. **Matemática computacional e aplicada**, v. 29, p. 107-124, 2010.

SALIDO, Miguel A. *et al.* A genetic algorithm for energy-efficiency in job-shop scheduling. The International **Journal of Advanced Manufacturing Technology**, v. 85, n. 5, p. 1303-1314, 2016.

SCHALLER, Jeffrey.; VALENTE, Jorge MS. A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness**. International Journal of Production Research**, v. 51, n. 3, p. 772-779, 2013.

SCHALLER, Jeffrey.; VALENTE, Jorge MS. Heuristics for scheduling jobs in a permutation flow shop to minimize total earliness and tardiness with unforced idle time allowed. **Expert Systems with Applications**, v. 119, p. 376-386, 2019.

SCOFIELD, Wendrer Carlos Luz. **Aplicação de Algoritmos Genéticos ao Problema Job-Shop.** 2002. 131 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Computação do Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Ouro Preto, 2002.

SILVA, Y. L. T.; SUBRAMANIAN, A.; & PESSOA, A. A. Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. **Computers & operations research**, 90, 142-160, 2018.

SIQUEIRA, Fernanda Rodrigues de.; MÜLLER, Carlos André da Silva. Integração entre Teoria dos Stakeholders e Visão Baseada em Recursos: trajetória percorrida pela literatura de Administração. **Revista Ibero-Americana de Estratégia**, 2022.

SUBRAMANIAN, Anand.; BATTARRA, Maria.; POTTS, Chris N. An iterated local search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. **International Journal of Production Research**, v. 52, n. 9, p. 2729-2742, 2014.

TANAKA, Shunji.; FUJIKUMA, Shuji.; ARAKI, Mituhiko. Um algoritmo exato para agendamento de máquina única sem tempo ocioso da máquina. **Journal of Scheduling** , v. 12, n. 6, pág. 575-593, 2009.

TAVAKKOLI-MOGHADDAM, Reza.; AZARKISH, Mojgan.; SADEGHNEJAD-BARKOUSARAIE, Azar. A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. **Expert Systems with Applications**, v. 38, n. 9, p. 10812-10821, 2011.

TUBINO, D. F. **Planejamento e controle da produção:** teoria e prática. São Paulo: Atlas, 2007.

VALLADA, Eva.; RUIZ, Rubén. Agendamento de máquinas paralelas não relacionadas com tempos de configuração dependentes da sequência e minimização ponderada de antecipação-atraso. In: **Sistemas Just-in-Time**. Springer, Nova York, NY, 2012. p. 67-90.

WANG, Lei; TANG, Dun-bing. An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem. **Expert Systems with Applications**, 2011.

WANG, Yuping. *et al.* A new hybrid genetic algorithm for job shop scheduling problem. **Computers & Operations Research**, v. 39, n. 10, p. 2291-2299, 2012.

YAZDANI, Maziar. **et al.** Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. **Computers & Industrial Engineering**, v. 107, p. 12-24, 2017.

YUCE, Baris et al. Hybrid Genetic Bees Algorithm applied to single machine scheduling with earliness and tardiness penalties. **Computers & Industrial Engineering**, v. 113, p. 842-858, 2017.

ZAMBRANO REY, Gabriel. *et al.* Solving the flexible job-shop just-in-time scheduling problem with quadratic earliness and tardiness costs. **The International Journal of Advanced Manufacturing Technology**, v. 81, n. 9, p. 1871-1891, 2015.

ZHANG, Guohui.; GAO, Liang.; SHI, Yang. An effective genetic algorithm for the flexible job-shop scheduling problem. **Expert Systems with Applications**, v. 38, n. 4, p. 3563-3573, 2011.