UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
ENGENHARIA DE PRODUÇÃO ELÉTRICA

Djonathan Luiz de Oliveira Quadras

Otimização Adaptativa Baseada em Simulação para a programação da produção em sistemas *flow shop*: um estudo comparativo

Florianópolis
2021

Djonathan Luiz de Oliveira Quadras

Otimização Adaptativa Baseada em Simulação para a programação da produção em sistemas *flow shop*: um estudo comparativo

Trabalho Conclusão de Curso de Graduação em Engenharia de Produção Elétrica do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Elétrica com Habilitação em Engenharia de Produção
Orientador: Prof. Dr.-Ing Enzo Morosini Frazzon.
Coorientador: Doutorando Lúcio Galvão Mendes.

Florianópolis
2021

Ficha de identificação da obra

Quadras, Djonathan Luiz de Oliveira
Otimização Adaptativa Baseada em Simulação para a programação da produção em sistemas flow shop : um estudo comparativo / Djonathan Luiz de Oliveira Quadras ; orientador, Enzo Morosini Frazzon, coorientador, Lúcio Galvão Mendes, 2021.
62 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Produção Elétrica, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Produção Elétrica. 2. Planejamento e Controle da Produção. 3. Programação do Flow Shop. 4. Otimização Adaptativa Baseada em Simulação. 5. Algoritmo Genético. I. Frazzon, Enzo Morosini . II. Mendes, Lúcio Galvão. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Produção Elétrica. IV. Título.

Djonathan Luiz de Oliveira Quadras

Otimização Adaptativa Baseada em Simulação para a programação da produção em sistemas *flow shop*: um estudo comparativo

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia Elétrica com Habilitação em Engenharia de Produção e aprovado em sua forma final pelo Curso de Engenharia de Produção Elétrica

Florianópolis, 03 de setembro de 2021.

Documento assinado digitalmente
Monica Maria Mendes Luna
Data: 28/09/2021 09:45:49-0300
CPF: 314.104.993-91
Verifique as assinaturas em https://v.ufsc.br

Profª. Mônica Maria Mendes Luna, Drª.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Enzo Morosini Frazzon
Data: 27/09/2021 15:23:03-0300
CPF: 014.696.429-21
Verifique as assinaturas em https://v.ufsc.br

Prof. Enzo Morosini Frazzon, Dr.- Ing.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Carlos Manuel Taboada Rodriguez
Data: 27/09/2021 20:39:13-0300
CPF: 004.044.459-77
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Manuel Taboada Rodriguez, Dr.
Avaliador(a)
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Lucio Galvao Mendes
Data: 27/09/2021 16:49:45-0300
CPF: 036.968.243-21
Verifique as assinaturas em https://v.ufsc.br

Doutorando Lucio Galvão Mendes
Avaliador(a)
Universidade Federal de Santa Catarina

Este trabalho é dedicado em memória do meu bocó, Valerio Quadras.

AGRADECIMENTOS

Agradeço aos meus pais, Luiz e Eliana, e aos meus avós, Valerio (*in memoriam*) e Terezinha, por todo o apoio, por estarem sempre comigo me apoiando, e por sempre terem acreditado em mim. Agradeço à minha namorada, Camila Bertelli, por toda a ajuda (inclusive na elaboração deste trabalho), todo o companheirismo, toda a compreensão e, principalmente, por ter me feito feliz até nos períodos em que não achei que conseguiria. Agradeço à minha sogra, Roseli, e ao meu cunhado, Matheus, por me acolherem e serem minha segunda família.

Agradeço a todos os meus amigos que Florianópolis me proporcionou, e principalmente aos meus companheiros da República da Chatuba, André, Gabriel Diz e Renan, e ao meu primo que tive o prazer de reencontrar, Valerio. Vocês fizeram todos os dias valerem a pena, e me mostraram que mesmo a pior das situações devem ser vistas como uma piada em potencial. Agradeço a todos os companheiros que estiveram comigo em minha passagem pela Colômbia, em especial as minhas queridas amigas Giovanna, minha fiel escudeira, o Sancho da minha Pança, *y Micaela, mi compañera de viajes*.

Agradeço a todos os meus companheiros do ProLogIS, em especial à Marina, ao Ícaro, ao Matheus e ao Lúcio. Sem vocês eu não conseguiria estar aqui hoje.

Agradeço a todos os professores da UFSC, CUC e UDESC que fizeram parte de todo este trajeto. Em especial ao professor Enzo por todos estes anos de trabalho em conjunto, por toda a confiança e por todos os ensinamentos; e aos professores Carlos Taboada e Diego Fettermann por toda a amizade e por terem me proporcionado minha primeira experiência internacional (e uma das melhores experiências de minha vida). Definitivamente, “se cheguei até aqui, foi porque me apoiei sobre o ombro de gigantes”.

Por fim, agradeço a todos aqueles que defendem e acreditam na ciência e que combatem diariamente o obscurantismo científico.

“Pra entender, basta seiscentos anos de estudo... Ou seis segundos de atenção."

(GESSINGER, 1988)

RESUMO

O planejamento e controle da produção é uma atividade essencial para todos os sistemas produtivos, uma vez que seu desempenho afeta diretamente critérios importantes como custo, qualidade, tempo e disponibilidade, fatores fundamentais para a competitividade empresarial. O problema de programação do *flow shop* é um dos problemas mais comuns envolvendo o planejamento de sistemas produtivos, sendo altamente aplicado no mundo real. A solução para este problema não pode ser encontrada facilmente devido à sua complexidade, o que impossibilita a utilização de métodos que retornem resultados ótimos a um baixo tempo de execução computacional. Assim, a Otimização Adaptativa Baseada em Simulação é uma ferramenta com potencial aplicação pois considera a estocasticidade do problema, buscando soluções que estejam alinhadas com o cenário real por meio de uma simulação e otimizando as variáveis desejadas. O objetivo geral do presente trabalho é avaliar comparativamente diferentes modelos de otimização adaptativa baseada em simulação para sistemas de produção *flow shop*. Foram considerados dois métodos para comparação, ambos aplicando um algoritmo genético integrado com uma simulação de eventos discretos. O primeiro método busca encontrar a melhor regra de sequenciamento para cada máquina do sistema produtivo, enquanto o segundo busca encontrar diretamente o melhor sequenciamento de ordens de produção sem uma regra pré-determinada. Os resultados mostram que um sequenciamento direto de ordens de produção retorna resultados melhores que as regras de despacho, tanto o *makespan* quanto o tempo de execução.

Palavras-chave: Planejamento e Controle da Produção. Programação do *Flow Shop*. Otimização Adaptativa Baseada em Simulação. Otimização Baseada em Simulação. Algoritmo Genético.

ABSTRACT

Production planning and control is an essential activity for all production systems, since its performance directly affects important criteria such as cost, quality, time and availability, factors that are fundamental for business competitiveness. The flow shop scheduling problem is one of the most common problems involving the planning of production systems, being highly applied in the real world. The solution to this problem cannot be easily found due to its complexity, which makes it impossible to use methods that return optimal results at a low computational execution time. Thus, Adaptive Simulation-Based Optimization is a tool with potential application as it considers the stochasticity of the problem, seeking solutions that are aligned with the real scenario through a simulation and optimizing the desired variables. The general objective of the present work is to comparatively evaluate different adaptive simulation-based optimization models for flow shop production systems. Two methods for comparison were considered, which apply a genetic algorithm integrated with a simulation of discrete events. The first method seeks to find the best dispatching rule for each machine in the production system, while the second seeks to directly find the best sequencing of production orders without a pre-determined rule. The results show that the rule-free sequencing of production orders returns better results than dispatch rules, both makespan and execution time.

Keywords: Production Planning and Control. Flow Shop Scheduling. Adaptive Simulation-Based Optimization. Simulation-Based Optimization. Genetic Algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASBO	-	*Adaptive Simulation-Based Optimization*
EDD	-	*Earliest Due Date*
FASFS	-	*First Arrival in System, First Served*
FIFO	-	*First In, First Out*
FSF	-	*Flow Shop* Flexível
GA	-	*Genetic Algorithm*
LIFO	-	*Last In, First Out*
PCP	-	Planejamento e Controle da Produção
SBO	-	*Simulation-Based Optimization*
SFSF	-	Sequenciamento do *Flow Shop* Flexível
SPT	-	*Shortest Processing Time*

GLOSSÁRIO

Flow Shop - Sistema de produção que consiste em múltiplas máquinas cujo fluxo de trabalho é unidirecional.

Makespan - Tempo total decorrido desde o momento em que o primeiro produto começar a ser produzido na primeira máquina até o momento em que o último produto termina de ser produzido pela última máquina.

SUMÁRIO

1 INTRODUÇÃO

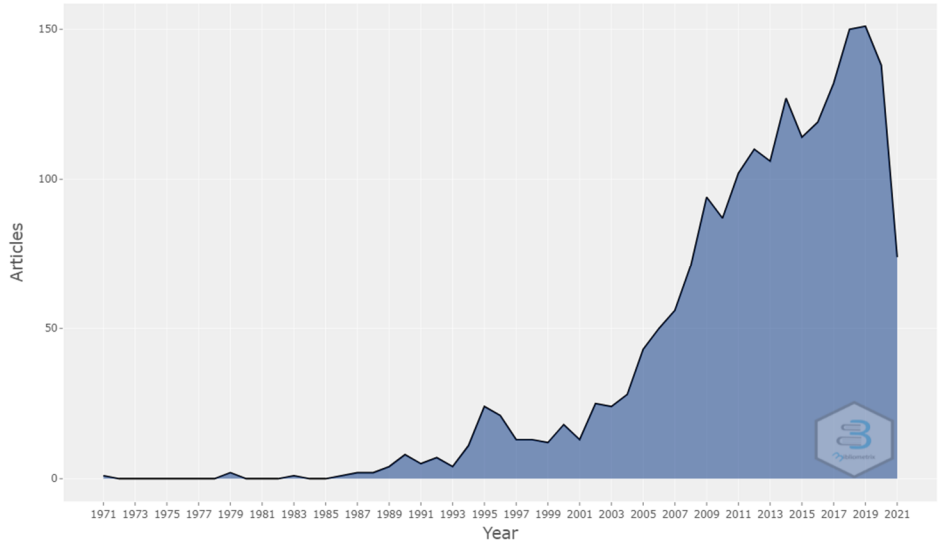
A programação da produção é uma atividade essencial para todos os sistemas produtivos, dado que seu desempenho afeta diretamente critérios importantes como custo, qualidade, tempo e disponibilidade. Tais fatores são fundamentais para a competitividade empresarial, motivação para que as empresas estejam sempre buscando suas melhorias por meio de soluções tecnológicas (AL-SHAYEA *et al.*, 2020; PAPROCKA; KRENCZYK; BURDUK, 2021).

Respostas rápidas para as mudanças no cenário industrial tem se mostrado um ponto chave para vantagem competitiva (BARBIERI *et al.*, 2021). Os sistemas industriais têm operado em sistemas gradativamente mais dinâmicos e voláteis, onde eventos imprevisíveis podem mudar todo o planejamento. Tal mudança faz com que uma programação anteriormente considerada ótima se torne obsoleta ou, até mesmo, inviável (BERGER; ZANELLA; FRAZZON, 2019).

Dentre os problemas mais comuns envolvendo o planejamento de sistemas produtivos, encontra-se o problema de programação do *flow shop,* considerado um dos tipos mais importantes de programação com um grande número de aplicações do mundo real (ABDEL-BASSET *et al.*, 2018). A literatura remete ao trabalho de Johnson (1954) como o primeiro trabalho notável a tentar buscar soluções para o problema, objetivando encontrar o sequenciamento ótimo para duas máquinas.

De fato, o tema "*flow shop*" tem recebido atenção da literatura nos últimos anos. A busca pelo tema em conjunto com "*optimization*" na base de dados *Scopus* revela estatísticas valorosas. Inicialmente, analisa-se o número de publicações ao longo dos anos, apresentado na Figura 1. Nota-se que o número de publicações acerca do tema tem crescido exponencialmente, apresentando o primeiro artigo (na plataforma) em 1971 até atingir o pico de publicações em 2019 com 150 artigos publicados. A partir da Figura 1, nota-se também que, apesar de apresentar uma tendência crescente, frequentemente existem declínios nas publicações. Tal situação o que sugere que, apesar do grande número de publicações na área, há muita possibilidade de pesquisas (CASTRO; FRAZZON, 2017).

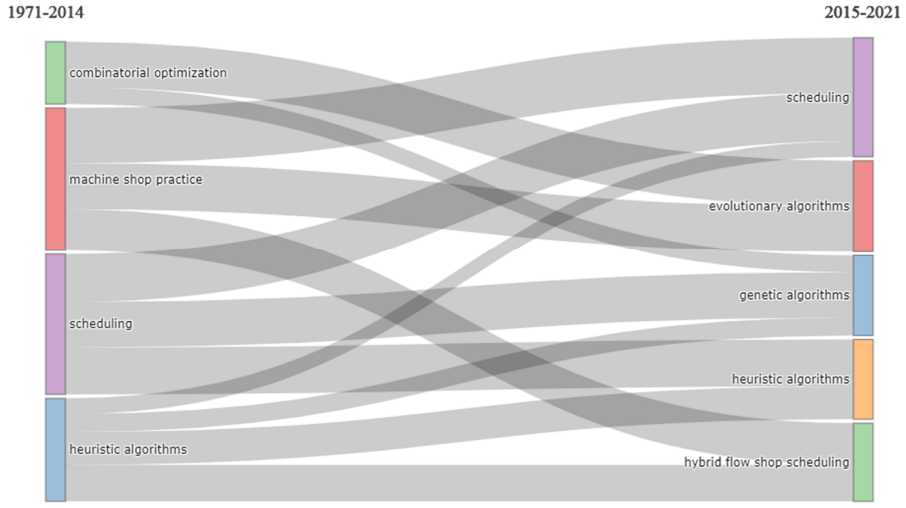
Figura 1 - Publicações ao Longo dos Anos



Fonte: base de dados SCOPUS

Outra discussão relevante acerca do tópico é a evolução temática, uma vez que torna possível visualizar o status, pontos de acesso, fronteiras e tendências de desenvolvimento de pesquisas (XIE *et al.*, 2020). De acordo com a Figura 2, de 1971 a 2014 os principais temas das pesquisas envolviam otimização combinatória, prática em chão de fábrica, sequenciamento e algoritmos heurísticos. O mapa ainda mostra que a partir de 2015 até 2021 se mantiveram os temas de sequenciamento e algoritmos heurísticos, entretanto, surgiram os algoritmos evolucionários e o estudo do sequenciamento do *flow shop* híbrido. Tal alteração sugere que a literatura compreende que as heurísticas evolucionárias são métodos fundamentais para o sequenciamento, além de incutir a aplicação de ferramentas computacionais.

Figura 2 - Evolução Temática



Fonte: base de dados SCOPUS

A principal dificuldade em se encontrar uma boa solução para o problema do sequenciamento do *flow shop* está na complexidade do problema, uma vez que admite um número incalculável de possibilidades. Tal impasse torna inviável a determinação de uma solução única e ótima (LEBBAR *et al.*, 2018). Para contornar dada situação, faz-se necessário a utilização de ferramentas matemáticas e heurísticas assistidas por computador. Entretanto, uma preocupação na aplicação de métodos computacionais é o tempo de execução (LEBBAR *et al.*, 2018).

Nesse contexto, Frazzon, Albrecht e Hurtado (2016) defendem a utilização de sistemas híbridos, que sejam capazes de encontrar soluções satisfatórias para problemas complexos e com um baixo tempo de execução. Além do desempenho, faz-se necessário que o método de solução seja capaz de lidar com incertezas dos sistemas *flow shop*. Shakibayifar *et al.* (2019) afirmam que uma Otimização Baseada em Simulação (SBO) é a ferramenta ideal pois considera a estocasticidade do problema, buscando soluções que estejam alinhadas com o cenário real por meio de uma simulação e otimizando as variáveis desejadas. Kuck *et al.* (2017) defendem a utilização de uma variação do SBO que considere de forma definitiva as aleatoriedades do problema: a Otimização Adaptativa Baseada em Simulação (ASBO). Ermolieva (2005) corrobora, afirmando que o ASBO é uma ferramenta capaz de otimizar se ajustando às alterações do sistema em tempo real.

Apesar das vantagens da utilização do ASBO, poucos artigos buscam aplicá-lo, e há pouca conexão com os trabalhos que abordam o método puro SBO. Também não foi possível encontrar artigos focados na revisão da literatura acerca do tema, tampouco buscando esmiuçar os modelos aplicados e campo de aplicação. Dessa forma, faz-se necessário encontrar quais os modelos SBO e ASBO utilizados na programação de sistemas *flow shop* na literatura atualmente, bem como compreender as vantagens e desvantagens de cada abordagem.

1.1 OBJETIVO GERAL

Com base no apresentado, o objetivo geral do presente trabalho é de avaliar comparativamente modelos de otimização adaptativa baseada em simulação para sistemas de produção *flow shop*.

1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, são elencados os seguintes objetivos específicos:

- Definir modelos ASBO a serem avaliados;
- Desenvolver os modelos de simulação de um sistema *flow shop* baseado em um sistema real;
- Implementar os modelos de otimização de acordo com as propostas;
- Avaliar comparativamente o desempenho dos modelos.

1.3 DELIMITAÇÃO DA PESQUISA

Para melhor desenvolver o trabalho, buscando garantir maior foco no alcance dos objetivos da pesquisa, são definidas as seguintes delimitações do estudo:

- A pesquisa explora apenas sistemas de produção *flow shop*, desconsiderando outros sistema (e.g., *job shops*);
- Os únicos métodos de otimização da programação da produção investigados são variações do método de otimização baseada em simulação, assim, não são consideradas outras formas de sequenciamento;
- O estudo avalia comparativamente apenas as heurísticas de otimização definidas, justificadas a partir da revisão da literatura, não sendo consideradas demais heurísticas existentes;
- A comparação dos métodos é realizada com base no desempenho de otimização de um caso real, não sendo estudados múltiplos casos de aplicação.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho é organizado conforme apresentado a seguir. O *Capítulo 2* apresenta o referencial teórico referente aos temas estudados e é dividido em (i) programação da produção em *flow shops*; (ii) otimização baseada em simulação; e (iii) algoritmo genético. O *Capítulo 3* apresenta a metodologia aplicada no desenvolvimento da pesquisa. O *Capítulo 4* apresenta os resultados obtidos e as discussões acerca destes resultados, sendo divido em: (i) modelos SBO definidos; (ii) apresentação do cenário real simulado; (iii) modelos de simulação desenvolvidos; e (iv) resultados das simulações. Por fim, o *Capítulo 5* apresenta as conclusões. O trabalho ainda contém, no *Anexo A,* o código desenvolvido para o algoritmo genético em linguagem *java*.

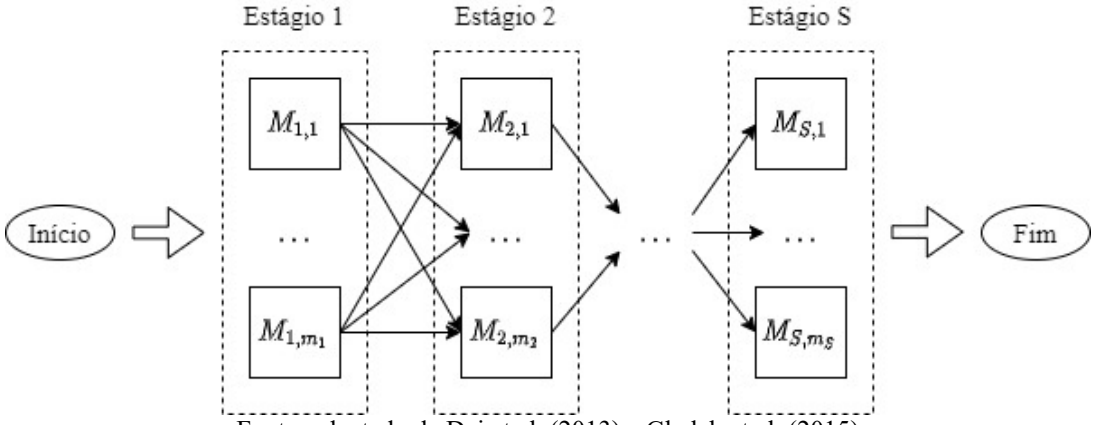
2 REFERENCIAL TEÓRICO

Nessa seção serão apresentados os conceitos utilizados para o desenvolvimento da pesquisa. Ela é dividida em três partes: (1) Programação de Produção em *Flow Shops*; (2) Otimização Baseada em Simulação; e (3) Algoritmo Genético.

2.1 PROGRAMAÇÃO DE PRODUÇÃO EM FLOW SHOPS

Os sistemas *flow shop* consistem em múltiplas máquinas cujo fluxo de trabalho é unidirecional (GUPTA, 1968). Gupta (1968) e Li et al. (2021) afirmam que o problema do sequenciamento em um *flow shop* (PSFS) é a determinação da sequência ótima da qual um conjunto de *jobs* (ordens de serviço) irá passar por um número específico de máquinas. Uma variação deste problema é o sequenciamento do *flow shop* flexível (SFSF), apresentado na Figura 3, que consiste em um ambiente com um número S de estágios, considerando que cada estágio i contém uma quantidade M_i de máquinas paralelas. No *flow shop* flexível (FSF), cada *job* segue o fluxo de produção de estágio em estágio, podendo ser produzido por qualquer máquina dos estágios (DAI *et al.*, 2013; GHALEB; SURYAHATMAJA; ALHARKAN, 2015).

Figura 3 - Representação de um *flow shop* flexível



Fonte: adaptado de Dai et al. (2013) e Ghaleb et al. (2015)

O principal objetivo dos artigos que abordam o PSFS é a minimização de *makespan*. O *makespan* é o tempo total da programação, i.e., o tempo total decorrido desde o momento em que o primeiro *job* começar a ser produzido na primeira máquina até o momento em que o último *job* termina de ser produzido pela última máquina (PALMER, 1965; RICHTER; WINKLER, 2017). A literatura remete a Palmer (1965) como o primeiro trabalho a considerar

a redução de *makespan* como objetivo principal. Entretanto, há diversos fatores a serem considerados no problema, que, de acordo com Rahman, Sarker e Essam (2017), tem como principais parâmetros que geram dificuldade em alcançar um bom planejamento são os tempos de *setup*, especificações do produto, tamanho do lote e capacidade de produção.

Além do *makespan*, há uma gama de artigos que visam a otimização do sequenciamento do *flow shop* para a minimização do custo de produção. Kumar e Lad (2016) propuseram uma abordagem integrada para programação de produção e planejamento de manutenção para um sistema *flow shop* híbrido considerando o efeito do custo de rejeição. Os autores aplicaram uma abordagem de otimização baseada em simulação para resolver o problema, integrando um algoritmo genético com uma simulação baseada em eventos discretos. Zhai *et al.* (2017) buscaram a otimização da programação da produção almejando aproveitar ao máximo a capacidade instalada de energia renovável para redução do custo da eletricidade. Os autores propuseram uma abordagem de programação dinâmica para minimizar o custo de eletricidade de um *flow shop* com uma turbina eólica integrada à rede. De forma análoga, Luo *et al.* (2013) aplicaram o conceito de manufatura verde a um sistema *flow shop* e propuseram uma nova meta-heurística de otimização de colônia de formigas considerando não apenas a eficiência de produção, mas também o custo de energia elétrica com a presença de preços de eletricidade por tempo de uso.

Há, na literatura, a presença de diversas variações do problema de sequenciamento de *flow* shop. Uma das variações que recebe bastante atenção é em como planejar a produção almejando a otimização do sequenciamento considerando a necessidade de manutenção. Paprocka, Krenczyk e Burduk (2021) afirmam que o tempo de inatividade da máquina relacionado à manutenção reduz a produtividade, mas que os custos incorridos devido a falhas não planejadas da máquina geralmente superam os custos associados à manutenção preditiva. Al-Shayea *et al.* (2020) corroboram com essa ideia, afirmando que um bom planejamento deve evitar uma manutenção excessiva, consequentemente evitando custos desnecessários.

O PSFS é, de acordo com Zhang e Chen (2016), classificado como um problema *np-hard* e, portanto, se faz necessário a utilização de heurísticas e/ou meta-heurísticas para a solução do problema. Uma das heurísticas mais utilizadas para o sequenciamento são as chamadas Regras de Despacho (RD) (KÜCK *et al.*, 2016). Kuck *et al.* (2017), Pires *et al.* (2020) e Richter e Winkler (2017) elencam como sendo as principais RDs aplicadas à produção:

- **FIFO:** "*First In, First Out*", a primeira ordem de produção a entrar na fila é a primeira a ser produzida;

- **LIFO:** "*Last In, First Out*", a última ordem de produção a entrar na fila é a primeira a ser produzida;
- **EDD:** "*Earliest Due Date*", a ordem de produção com a data de entrega mais próxima é a primeira a ser produzida;
- **FASFS:** "*First Arrival in System, First Served*", primeira ordem de produção a entrar no sistema é a primeira a ser produzida;
- **SPT:** "*Shortest Processing Time*", a ordem de produção com menor tempo de processamento na fila é a primeira a ser produzida.

As regras de despacho, apesar de muito utilizadas, se mostram bastante ineficientes se utilizadas sozinhas pois não correspondem a métodos matemáticos muito elaborados (KÜCK *et al.*, 2016). Diversas heurísticas e meta-heurísticas mais robustas são aplicadas para a solução dos problemas de *flow shop* e recebem bastante destaque por retornarem resultados melhores que as RD, com destaque para: *Ant Colony Optimization* (PAPROCKA; KRENCZYK; BURDUK, 2021), Algoritmos Evolucionários e/ou Algoritmo Genético (DELLA CROCE; TADEI; VOLTA, 1995; ONWUBOLU; DAVENDRA, 2006; RAHMAN; SARKER; ESSAM, 2017; ZUO *et al.*, 2020), *Whale Optimization Algorithm* (ABDEL-BASSET *et al.*, 2018; LI *et al.*, 2021), Pesquisa Tabu (AL-SHAYEA *et al.*, 2020), *Particle Swarm* (BEWOOR; PRAKASH; SAPKAL, 2018; FEI; MA, 2018; LIU; WANG; JIN, 2008; MUHARNI *et al.*, 2019) e Gêmeos Digitais (BARBIERI *et al.*, 2021).

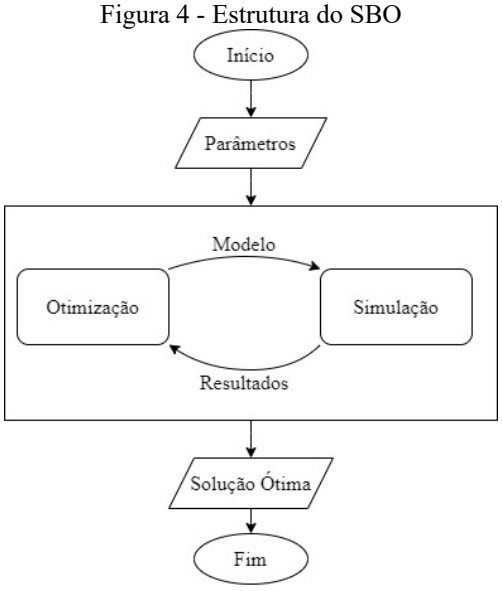
A complexidade em se encontrar um método ótimo para o problema do sequenciamento do *flow shop* motivou Dolgui *et al.* (2019) a desenvolverem uma revisão da literatura referente a programação da produção, cadeia de suprimentos e a Indústria 4.0. Os autores apresentam uma grande quantidade de aplicações, métodos e modelos possíveis a serem aplicados, além de mostrarem a complexidade e evidenciarem a quantidade de variáveis, parâmetros e cenários possíveis de serem explorados na solução dos problemas estudados. Dolgui *et al.* (2019) e Abdel-Basset *et al.* (2018) afirmam que o principal problema encontrado pelos algoritmos existentes na literatura são a descoberta de ótimos locais e o elevado custo computacional. Ademais, Zuo *et al.* (2020) defendem que diversos modelos de Algoritmos Evolucionários são utilizados na literatura, porém é uma tarefa complexa determinar qual modelo é o mais eficiente. Frente a estes desafios, Frazzon, Albrecht e Hurtado (2016), Kuck *et al.* (2017) e Pires *et al.* (2020) sugerem que a aplicação de uma Otimização Baseada em Simulação (SBO) é capaz de encontrar soluções eficientes com um baixo custo computacional por meio de uma metodologia híbrida.

2.2 OTIMIZAÇÃO BASEADA EM SIMULAÇÃO

A Otimização Baseada em Simulação (do inglês *Simulation-Based Optimization* - SBO) é uma boa ferramenta para dar suporte à tomada de decisão no Planejamento e Controle da Produção de sistemas industriais (HEINZL; KASTNER, 2020). Sua aplicação é altamente recomendada em situações onde há incertezas e dados estocásticos (AZIMI; SHOLEKAR, 2021; IRAWAN *et al.*, 2021). Ansari (2021) afirma que uma modelagem híbrida igual ao SBO para analisar efeitos mútuos de múltiplas fontes de incerteza pode gerar projetos mais eficientes e com maior grau de confiança.

A abordagem da otimização baseada em simulação consiste em um *loop* de *feedback* entre o simulador e o otimizador para obter planos mais robustos (CASTAÑÉ *et al.*, 2019). Azab, Karam e Eltawil (2019) corroboram com essa metodologia, afirmando que o modelo funciona da seguinte forma: O sistema é alimentado com todos os parâmetros necessários tanto para a otimização quanto para a simulação. A otimização então gera e fornece modelos e parâmetros que são enviados para a simulação. A simulação testa todos os modelos gerados pela otimização e retorna, para a otimização, os valores necessários (geralmente são os da função objetivo que estão sendo minimizados ou maximizados). A otimização recebe e analisa os valores simulados e, a partir deles, cria modelos e parâmetros que novamente serão enviados para a simulação. Essa iteração é repetida diversas vezes até que seja alcançado o ponto de parada definido para a otimização. No fim, é retornado do sistema a solução ótima. É apresentado na Figura 5 o fluxograma referente à aplicação do SBO.

Figura 4 - Estrutura do SBO



Fonte: adaptado de Castañé *et al.* (2019) e Mazzuco *et al.* (2018)

Como as variáveis estocásticas são frequentemente tratadas pela otimização, os trabalhos que aplicam o SBO geralmente utilizam uma simulação de eventos discretos (ANSARI, 2021; AZAB; KARAM; ELTAWIL, 2019; HEINZL; KASTNER, 2020; KÜHN; VÖLKER; SCHMIDT, 2020; MAZZUCO *et al.*, 2018). Diversas meta-heurísticas são aplicadas na otimização, sendo alguns exemplos: *Neighborhood Search* (ANSARI, 2021; HEINZL; KASTNER, 2020; IRAWAN *et al.*, 2021; TURAN *et al.*, 2020), *Simulated Annealing* (MAZZUCO *et al.*, 2018)*, Local Search Algorithm* (LI; KOOLE; JOUINI, 2019) e *Genetic Algorithm* (BERGER; ZANELLA; FRAZZON, 2019; KÜCK *et al.*, 2016; KÜHN; VÖLKER; SCHMIDT, 2020).

As abordagens que buscam estudar os processos de Planejamento e Controle da Produção devem compensar as influências estocásticas e atingir um alto desempenho no alcance de objetivos (KÜHN; VÖLKER; SCHMIDT, 2020). Kühn, Völker e Schmidt (2020) propuseram um SBO aplicado ao PCP com o objetivo de minimizar o *makespan* do projeto. Em contrapartida, Heinzl e Kastner (2020) objetivaram minimizar os custos do PCP avaliando os custos de energia em conjunto com as metas de produção (e.g. armazenamento), aplicando o método em um estudo de caso na programação de um *flow shop* de uma padaria industrial. Os resultados de Kühn, Völker e Schmidt (2020) demonstram que os resultados encontrados pelo SBO reduzem o tempo computacional ao mesmo tempo que aumenta a eficiência, o que é corroborado por Heinzl e Kastner (2020) que afirmam que a aplicação resultou em uma redução de custos de 6%.

Além das vantagens de se aplicar o SBO ao PCP, a otimização baseada em simulação apresenta grandes ganhos se aplicada em diversos outros cenários. Irawan *et al.* (2021) desenvolveram um modelo de otimização voltado para o Problema de Roteamento de Manutenção de turbinas de um parque eólico com o objetivo minimizar os custos. A aplicação do SBO foi aplicada devido a estocasticidade das variáveis relacionadas às rotas e consistiu em uma aplicação iterativa de uma simulação de Monte Carlo e os resultados oriundos da otimização. Turan *et al.* (2020) aplicaram o SBO para resolver o problema de otimização do planejamento de manutenção conjunta utilizando um modelo de simulação de evento discreto para modelar e analisar a rede de manutenção. O método de otimização consistiu em uma variação das variáveis de decisão, encontrando uma redução de custos de 81,42% em comparação com outros métodos.

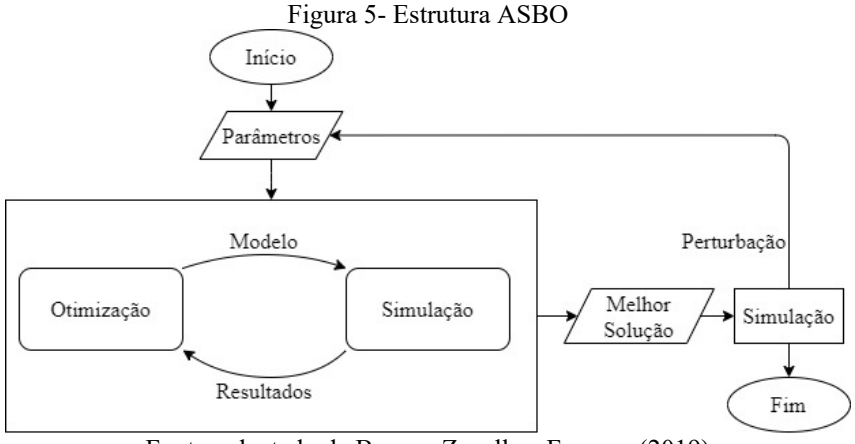
A otimização baseada em simulação também tem grande utilidade na área de logística e cadeia de suprimentos, conforme atestam (MAZZUCO *et al.*, 2018), que desenvolveram um

framework buscando uma solução mais adequada para o Problema do Roteamento de Veículos, encontrando resultados mais eficientes que os métodos tradicionais. Azab, Karam e Eltawil (2019) aplicaram a abordagem na programação em terminais de caminhões considerando as operações de pátio e portão, tempo de espera, emissões nocivas e produtividade do terminal, bem como suas naturezas estocásticas, atingindo uma redução de tempo de espera dos caminhões em 38%

Outro cenário interessante para a aplicação dessa ferramenta, além do cenário industrial, é a área de serviços. Li, Koole e Jouini (2019) aplicaram uma otimização baseada em simulação para solucionar o Problema de Pessoal e Programação de Turnos para um centro de contato de multicanais e múltiplas habilidades buscando maximizar a qualidade do serviço. Li, Koole e Xie (2020) aplicaram um SBO para o sequenciamento de pacientes de quimioterapia, buscando uma programação que fosse simples, flexível e justa com os pacientes. Considerando os fatores estocásticos no fornecimento de serviços de manutenção periódica, Castañé *et al.* (2019) aplicaram a otimização baseada em simulação para avaliar diferentes opções de design, relacionadas a decisões de pessoal, estratégias de escalonamento de técnicos e melhorias tecnológicas, otimizando para a geração dos planos diários.

A Indústria 4.0 tem exigido que as soluções para os problemas industriais estejam preparadas para enfrentar problemas em tempo real, assim sendo necessário um modelo que, além de resultar em uma programação ótima, também seja adaptativa para os diversos cenários (BERGER; ZANELLA; FRAZZON, 2019). Diante dessa necessidade, Kück *et al.* (2016) e Kuck *et al.* (2017) apresentam uma variação do SBO, chamada Otimização Adaptativa Baseada em Simulação (do inglês *adaptive simulation-based optimization* – ASBO). O fluxograma correspondente ao ASBO está apresentado na Figura 5.

Figura 5- Estrutura ASBO



Fonte: adaptado de Berger, Zanella e Frazzon (2019)

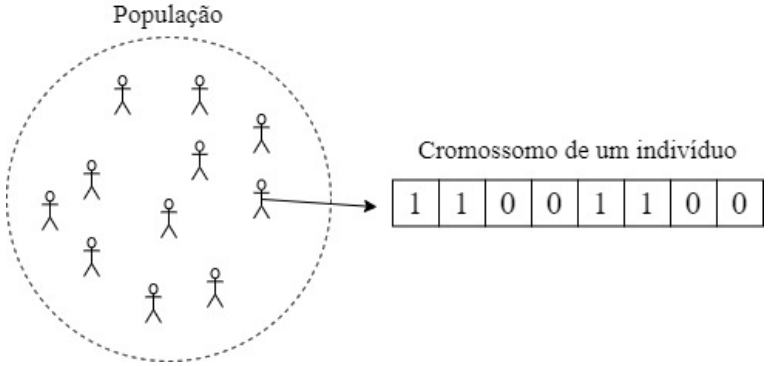
O ASBO consiste em um método mais robusto para a solução de problemas complexos. Sua modelagem consiste em: O problema é iniciado e uma solução ótima é encontrada pelo problema (aplicando um SBO). Essa solução alimenta um outro modelo de simulação, que irá utilizar a solução encontrada como parâmetros ótimos até que alguma perturbação ocorra no sistema. O sistema necessita encontrar uma nova solução que englobe os efeitos dessa perturbação (sendo, assim, adaptativo à estocasticidade do sistema). Dessa forma, é realizado um novo SBO que otimize os novos parâmetros e que alimentará novamente a simulação. Essa iteração ocorrerá sempre que houver uma perturbação (BERGER; ZANELLA; FRAZZON, 2019; KUCK *et al.*, 2017; KÜCK *et al.*, 2016).

2.3ALGORÍTMO GENÉTICO

O Algoritmo Genético (do inglês *Genetic Algorithm* – GA) é um método heurístico inicialmente proposto por Holland (1973) para a solução de problemas de otimização complexos que sejam caracterizados por: (1) uma incerteza inicial; (2) pela necessidade da aquisição de novas informações de forma rápida para a redução de incertezas; (3) pela necessidade de que a nova informação seja explorada da forma que foi adquirida para se obter um aumento de performance.

O conceito desse método surgiu da teoria de Charles Darwin em seus estudos acerca da evolução das espécies, conforme explicam Chen, Pan e Lin (2008). Esse método consiste em uma simulação de uma população composta por p indivíduos. Cada indivíduo possui uma carga genética chamado cromossomo. O cromossomo é a decodificação de uma solução para o problema estudado (CHEN; PAN; LIN, 2008; KESKIN; ENGIN, 2021). A Figura 6 mostra uma representação da população, indivíduo e cromossomo.

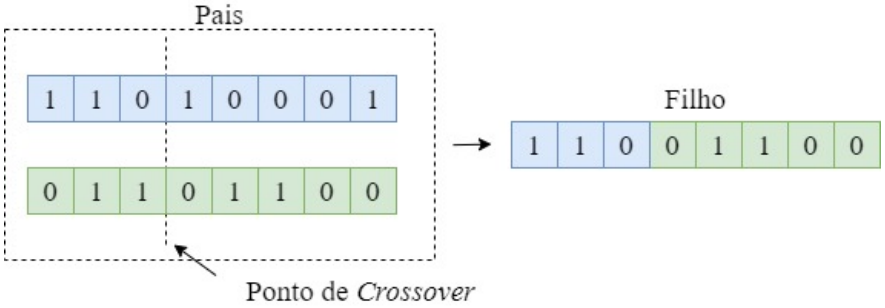
Figura 6 - Representação de População e Cromossomo



Fonte: O Autor (2021)

Assim como a teoria darwiniana, os indivíduos da população irão se reproduzir, gerando novas soluções. Esse processo é chamado de *crossover*, representado pela Figura 7. O *crossover* consiste em: duas soluções (chamadas pais) irão fornecer parte do seu material genético (i.e., uma parte da sua solução), o que resultará em uma solução filho. Após o fim da etapa de *crossover*, a população terá um tamanho $(p + n)$, sendo p o tamanho inicial da população e n a quantidade de novos indivíduos gerados. O c*rossover* ocorre devido a uma certa probabilidade especificada no problema (geralmente um valor alto, e.g., 80%), mas a escolha dos indivíduos a realizarem o crossover e o ponto onde ocorrerá a troca de material são aleatórios (ABDEL-BASSET *et al.*, 2021; IDZIKOWSKI; RUDY; GNATOWSKI, 2021).

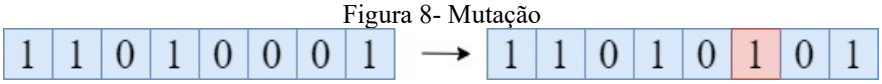
Figura 7 - Crossover



Fonte: adaptado de Abdel-Basset *et al.* (2021)

Ainda de acordo com a teoria da evolução, é possível que os novos descendentes nasçam com alguma mutação. Esse conceito é considerado no Algoritmo Genético e, de acordo com Engin, Ceran e Yilmaz (2011), é uma operação de extrema importância por garantir a diversidade da população e prevenir uma convergência prematura. A mutação, representada na Figura 8, consiste na alteração de um gene na geração de um novo indivíduo, ocorrendo com uma certa probabilidade relativamente baixa, na ordem de 10% (ABDEL-BASSET *et al.*, 2021; KESKIN; ENGIN, 2021; SARAÇOĞLU; SÜER; GANNON, 2021).

Figura 8- Mutação



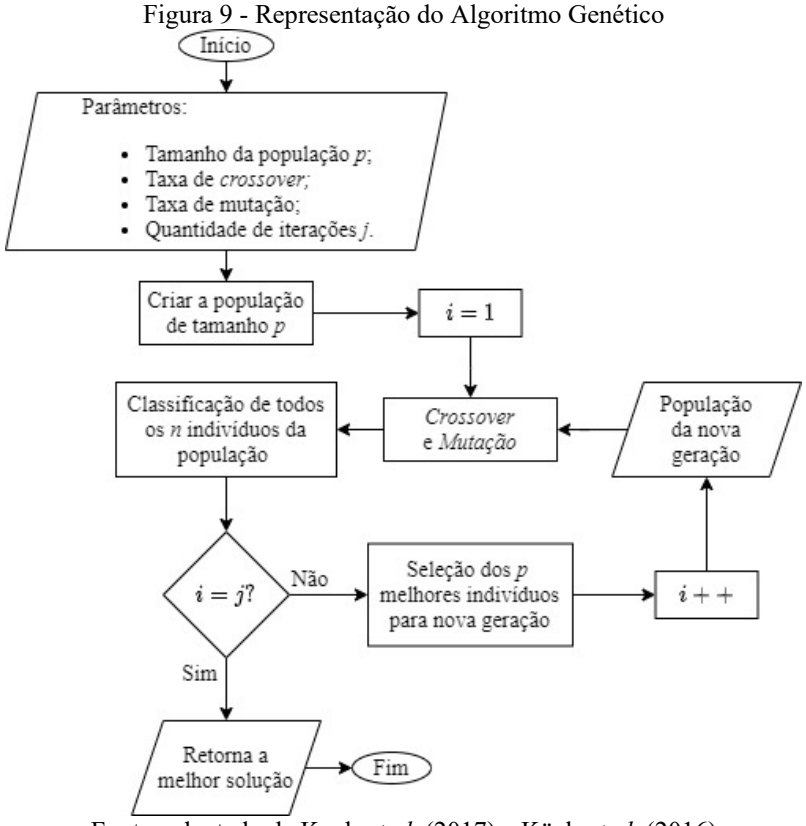
Fonte: adaptado de Abdel-Basset *et al.* (2021)

Após a realização das operações de *crossover* e mutação, todos os indivíduos (soluções) serão avaliados para o modelo proposto. Essa avaliação retorna o valor da variável objetivo de ser minimizada ou maximizada (e.g., *makespan*, custo, entregas tardias), chamada de *fitness* (SARAÇOĞLU; SÜER; GANNON, 2021). Para o cálculo do *fitness*, geralmente são utilizados modelos matemáticos de pesquisa operacional, e.g. Karacan *et al.*, (2021), entretanto,

Kuck *et al.* (2017) para se obter um modelo eficiente e que reduza o tempo computacional é indicado a utilização de uma simulação computacional.

Por fim, após o cálculo do *fitness*, ocorre o último estágio: a seleção. Nesse estágio, a população é classificada de acordo a variável estudada e os n indivíduos que obtiverem as piores soluções serão descartados, fazendo com que a população volte a ter o tamanho inicial p. Caso seja alcançado o critério de parada (número máximo de iterações), o algoritmo será encerrado e será retornada a solução ótima para o problema. Caso contrário, volta-se à etapa de *crossover* e continua-se o algoritmo até que o critério de parada seja alcançado (KESKIN; ENGIN, 2021; MURATA; ISHIBUCHI, 1994). A Figura 9 apresenta o Algoritmo Genético e todas as suas etapas.

Figura 9 - Representação do Algoritmo Genético



Fonte: adaptado de Kuck *et al.* (2017) e Kück *et al.* (2016)

O Algoritmo Genético é uma das meta-heurísticas mais utilizadas para a solução do problema de sequenciamento do *flow shop* (IDZIKOWSKI; RUDY; GNATOWSKI, 2021). Abdel-Basset *et al.* (2021) aplicaram o GA para a solução do problema do *flow shop* flexível e seus resultados apontam que o método é competitivo com outras metodologias encontradas na literatura. Engin, Ceran e Yilmaz (2011) aplicaram a metodologia para a solução do problema

do *flow shop* híbrido com tarefas multiprocessadoras com o intuito de reduzir o *makespan* e seus resultados apontam que o GA determinou o *lower bound* para vários cenários e retornou a melhor solução em 48,3% dos casos, indicando assim ser um algoritmo eficiente para a redução de *makespan*.

3 METODOLOGIA

Nessa seção será apresentada a metodologia de pesquisa, que será dividida em duas partes: (i) o enquadramento metodológico e (ii) os procedimentos metodológicos.

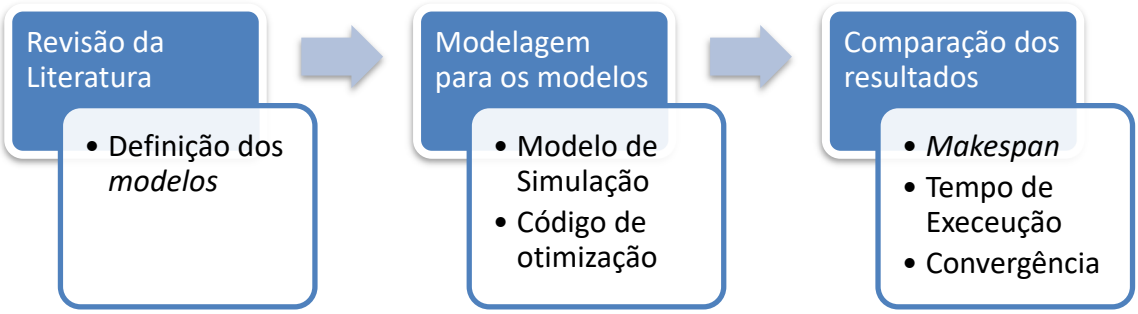
3.1 ENQUADRAMENTO METODOLÓGICO

O presente trabalho consiste em uma abordagem quantitativa por ser possível quantificar numericamente a relação entre eventos, além de apresentar modelos descritos em linguagem matemática e computacional, que utilizam técnicas analíticas, tanto matemáticas quanto estatísticas, ou experimentais, como o caso da simulação, para calcular valores numéricos das propriedades do sistema estudado (GIL, 2002; MIGUEL *et al.*, 2010). A fonte de informação é caracterizada como uma fonte de campo, por serem fornecidos por uma empresa real (GIL, 2002). Quanto à classificação com base nos objetivos, o presente trabalho é caracterizado por uma pesquisa exploratória, caracterizada por ter um objetivo principal voltado para o aprimoramento de ideias ou a descoberta de intuições, envolvendo a análise de exemplos que estimulem a compreensão (GIL, 2002). Em relação aos procedimentos técnicos, o presente trabalho se caracteriza pela aplicação de uma abordagem de Modelagem e Simulação, a qual imita as operações de um sistema real à medida que este evolui no tempo (MIGUEL *et al.*, 2010).

3.2 PROCEDIMENTOS METODOLÓGICOS

A pesquisa consistiu nas seguintes etapas, conforme apresentado pela Figura 10: (i) revisão da literatura para a definição dos modelos para comparação; (ii) modelagem dos dois modelos em *software* computacional; (iii) desenvolvimento do código da heurística de otimização e integração com a simulação; e, finalmente (iv) simulação e comparação dos resultados considerando o *makespan*, tempo de execução e convergência dos modelos.

Figura 10- Metodologia Aplicada



Fonte: O Autor (2021)

3.2.1 Procedimentos para a revisão da literatura

Para definição dos modelos foi realizada uma revisão da literatura com uma pesquisa na base de dados SCOPUS (TRISKA; FRAZZON; SILVA, 2020), utilizando as palavras-chave apresentadas na Tabela 1.

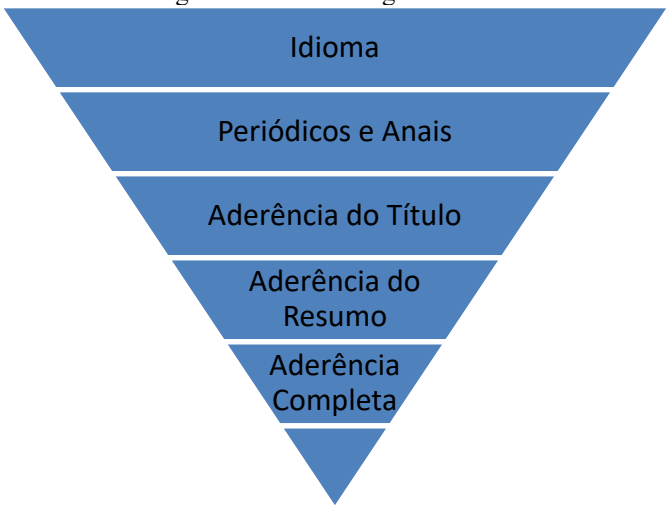
Tabela 1 - Palavras-Chave Utilizadas

Modelo	**Palavra-Chave**
SBO	(“*production scheduling” AND “Simulation-Based Optimization*”)
ASBO	“*Adaptive Simulation-Based Optimization*”

Fonte: O Autor (2021)

As palavras-chave foram escolhidas almejando encontrar tanto os modelos SBO utilizados na literatura para o sequenciamento da produção quanto os modelos ASBO existentes. Os artigos encontrados foram selecionados de acordo com a metodologia PRISMA (UHLMANN; FRAZZON, 2018) apresentada na Figura 11: (i) filtro por artigos nos idiomas inglês, espanhol, português e italiano; (ii) filtro por artigos publicados em periódicos e anais de congresso; (iii) filtro pela aderência do título com o tema; (iv) filtro da aderência do resumo; e, finalmente, (v) filtro da aderência artigos completos. Após a seleção dos artigos mais relevantes para a pesquisa, foi feita uma análise acerca do método aplicado. A análise comparou o otimizador utilizado, o método de simulação implementado e as variáveis a serem otimizadas (cromossomo da otimização). Com base nesses parâmetros foi definido o novo modelo SBO e o modelo a ser utilizado para comparação.

Figura 11 - Metodologia PRISMA

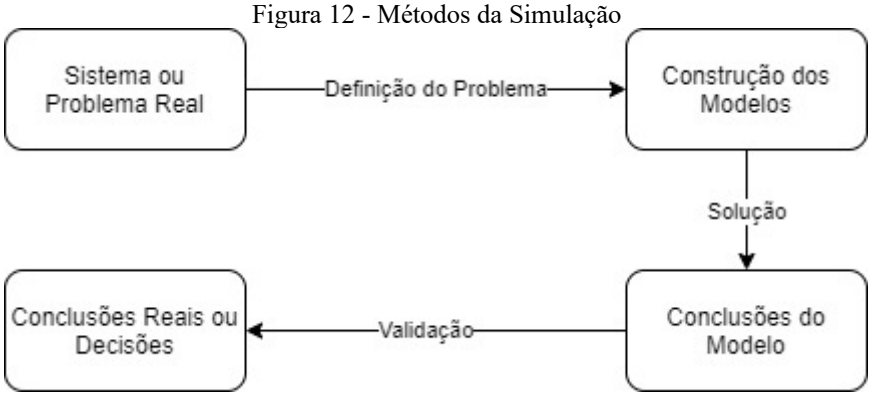


Fonte: adaptado de Uhlmann e Frazzon (2018)

3.2.2 Procedimentos para modelagem da simulação e otimização

O procedimento metodológico para o desenvolvimento da modelagem da simulação e otimização é baseado no proposto por Miguel *et al.* (2010) e está apresentado na Figura 12. O método consiste nas seguintes etapas: (1) Inicialmente foi definido o sistema a ser simulado, baseado em um cenário de uma empresa real; (2) Em seguida, foi estudado o cenário real, considerando suas características e peculiaridades. A partir do cenário real, foi definido o problema a ser estudado, considerando as máquinas, ordens de produção e os parâmetros inerentes ao processo produtivo. Nessa etapa também foi definido o modelo matemático que caracteriza o problema estudado. (3) Após a definição do problema, foi simulado em um *software* computacional o cenário real aplicando uma simulação de eventos discretos. De forma integrada à simulação, foi desenvolvida a otimização com base no modelo matemático desenvolvido. (4) Com os modelos de simulação e otimização desenvolvidos, foram simulados os cenários aplicando-se os métodos encontrados na revisão da literatura. (5) Por fim, com base nos parâmetros definidos, foram validados os resultados obtidos.

Figura 12 - Métodos da Simulação



Fonte: adaptado de Miguel *et al.* (2010)

As especificações técnicas são as seguintes: as simulações foram desenvolvidas no *software* AnyLogic enquanto a otimização foi inteiramente desenvolvida pelo autor em linguagem Java e adicionada como classe dentro do *software* (o código encontra-se disponível no Anexo A deste trabalho). O computador utilizado para a realização dos experimentos tem as seguintes características: sistema operacional Microsoft Windows 10 Home Single Language, versão 10.0.19042 x64, processador Intel® Core™ i&-8565U CPU @ 1.0GHz, 4 núcleos, 8 processadores lógicos, 16 GB de memória RAM.

4 RESULTADOS E DISCUSSÕES

Nessa seção serão apresentados os resultados da revisão da literatura, apresentado os modelos a serem comparados, a modelagem desenvolvida, os resultados da aplicação e as discussões acerca dos resultados obtidos.

4.1 MODELOS SBO

Para os modelos SBO, a busca na base de dados SCOPUS retornou com um total de 17 resultados. Após a aplicação dos filtros de idioma e aderência ao tema, foram selecionados oito artigos que possuíam relação com o problema de pesquisa. Os artigos estão apresentados na Tabela 2, em que expõe o artigo, o otimizador (i.e., método de otimização), o modelo de simulação e as variáveis a serem otimizadas. Vale ressaltar que foram encontrados outros artigos que aplicam o SBO mas que não se enquadraram no escopo desta pesquisa, como o caso de Feng, Rao e Raturi (2011) que utilizaram o SBO (otimizador de Aproximação Gama e simulação de eventos discretos) como apenas uma parte de uma metodologia a fim de calcular o estoque de segurança a cada iteração.

Tabela 2 - Artigos Selecionados SBO

Referência	Otimizador	Simulador	Variáveis
Völker e Gmilkowsky (2003)	Modelo Próprio	Eventos Discretos	Regras de Despacho
Ehrenberg e Zimmermann (2012)	Programação Inteira Mista	Eventos Discretos	Sequenciamento
Zhang e Rose (2013)	Algoritmo Genético	Eventos Discretos	Alocação de Máquinas
Kuck *et al.* (2017)	Algoritmo Genético	Eventos Discretos	Regras de Despacho
Kück *et al.* (2016)	Algoritmo Genético	Eventos Discretos	Regras de Despacho
Kumar e Lad (2016)	Algoritmo Genético	Monte Carlo	Sequenciamento
Purohit *et al.* (2017)	Algoritmo Genético	Monte Carlo	Sequenciamento
Heinzl e Kastner (2020)	*Neighborhood Search*	Eventos Discretos	Sequenciamento

Fonte: O Autor (2021)

A partir da Tabela 2 pode-se perceber determinados padrões de solução. O otimizador mais utilizado é o Algoritmo Genético, apesar de aparecerem outras heurísticas como a Programação Inteira Mista, *Neighborhood Search* e até mesmo heurísticas novas, como a proposta por Völker e Gmilkowsky (2003). Os modelos de simulação mais utilizados são as

simulações de eventos discretos assistidas por computador e as simulações de Monte Carlo. Quanto às variáveis a serem otimizadas, a literatura se mostra bastante dividida em otimizar a Regra de Despacho a ser implementada na linha de produção para sequenciar os *jobs* ou em otimizar o próprio sequenciamento de uma forma livre, não respeitando uma regra pré-estabelecida. Ainda quanto às variáveis, Zhang e Rose (2013) aplicaram um SBO a fim de otimizar o sequenciamento das máquinas no chão de fábrica para, assim, otimizar o *makespan*.

A pesquisa relacionada ao modelo ASBO retornou quatro artigos. Após os filtros, e selecionando os artigos disponíveis para leitura, restaram dois trabalhos, apresentados na Tabela 3. Kuck *et al.* (2017) propuseram a aplicação de um ASBO para determinar qual a melhor regra de despacho a ser aplicada no sequenciamento da produção, utilizando um Algoritmo Genético integrado com uma Simulação de Eventos Discretos. Pires *et al.* (2018) aplicaram o conceito de Kuck *et al.* (2017) para o sequenciamento da produção na otimização de toda a cadeia logística, composta por fornecedores, produção e clientes. Os autores aplicaram a heurística *Simulated Annealing* (Têmpera Simulada) integrado com uma simulação de eventos discretos.

Tabela 3 - Artigos Selecionados ASBO

Referência	Otimizador	Simulador	Variáveis
Pires *et al*. (2018)	*Simulated Annealing*	Eventos Discretos	Regras de Despacho
Kuck *et al.* (2017)	Algoritmo Genético	Eventos Discretos	Regras de Despacho

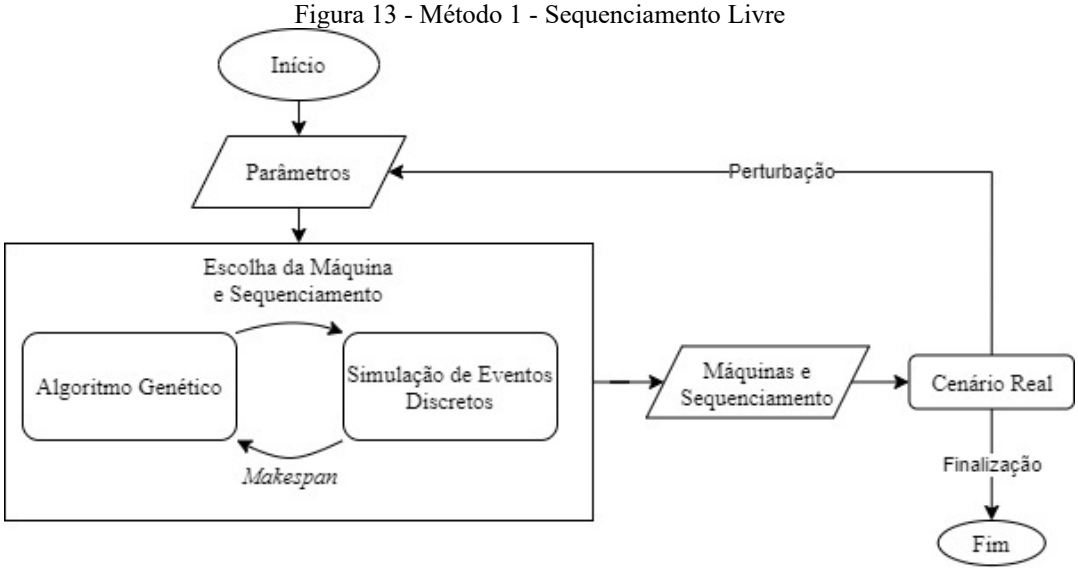
Fonte: O Autor (2021)

De acordo com a Tabela 3, os artigos que aplicam o método ASBO seguem um padrão, diferenciando unicamente no otimizador aplicado. Comparando os resultados encontrados das buscas de trabalhos que abordem o SBO e o ASBO, nota-se que a principal divergência entre os modelos são as variáveis a serem otimizadas. Ademais, os trabalhos abordando ASBO não exploram outras variáveis, aplicando unicamente a heurística a fim de determinar a regra de despacho ótima para o sequenciamento, enquanto os trabalhos voltados unicamente ao SBO consideram um sequenciamento livre, sem uma regra anteriormente estabelecida. Ainda, é notável que a heurística Algoritmo Genético é a mais utilizada por todos os trabalhos. Assim, é possível estabelecer dois métodos para comparação.

4.1.1 Método 1 – Sequenciamento Livre

O Método 1, apresentado na Figura 13, consiste em um ASBO composto por um algoritmo genético integrado com uma simulação de eventos discretos. A otimização será iniciada a cada chegada de pedidos de demandas.

Figura 13 - Método 1 - Sequenciamento Livre



Fonte: adaptado de Berger, Zanella e Frazzon (2019) e Heinzl e Kastner (2020)

O cromossomo característico do Método 1, conforme apresentado na Figura 14, considerará (i) a máquina m que irá produzir cada *job* j e (ii) a prioridade λ_j de cada *job* j para definir o sequenciamento livre.

Figura 14 - Cromossomo do Método 1

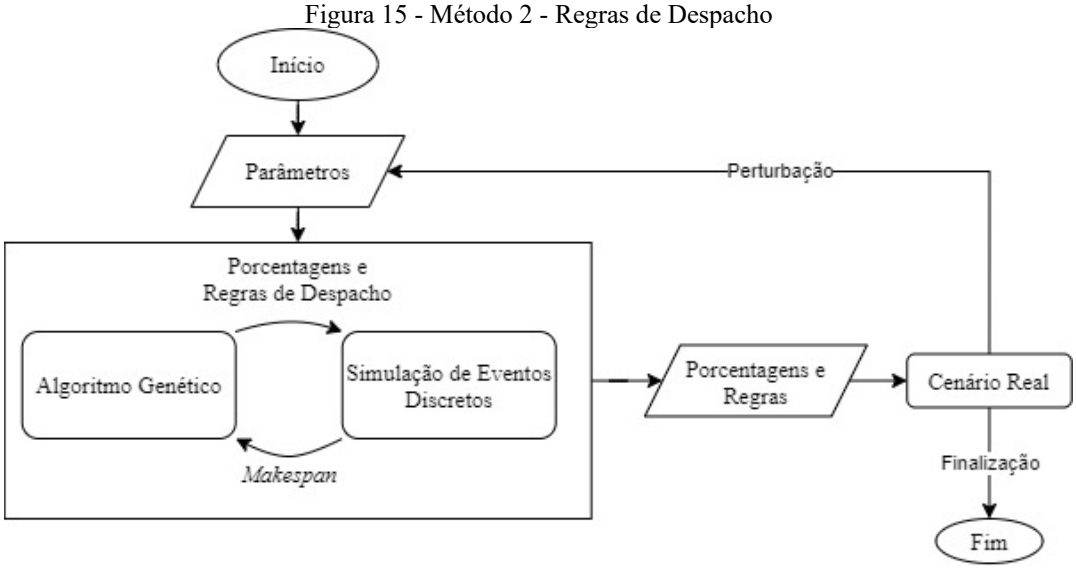
m_1	m_2	$\ldots$	m_j	λ_1	λ_2	$\ldots$	λ_j

Fonte: adaptado de Heinzl e Kastner (2020)

4.1.2 Método 2 – Regras de Despacho

O Método 2, apresentado na Figura 15, é baseado no modelo de Kuck *et al.* (2017). Semelhante ao Método 1, consiste em um ASBO composto por um algoritmo genético integrado com uma simulação de eventos discretos. A otimização será iniciada a cada chegada de pedidos de demandas.

Figura 15 - Método 2 - Regras de Despacho



Fonte: adaptado de Berger, Zanella e Frazzon (2019) e Kuck *et al.* (2017)

O cromossomo característico do Método 2, conforme apresentado na Figura 16, considerará (i) a porcentagem δ_m de *jobs* serão produzidos por cada máquina m e (ii) a regra de despacho R_m para definir o sequenciamento de cada máquina m. Serão consideradas as regras de despacho FIFO, LIFO, EDD, FASFS e SPT**.**

Figura 16 - Cromossomo do Método 2

δ_1	δ_2	...	δ_m	R_1	R_2	...	R_m

Fonte: adaptado de Kuck *et al.* (2017)

4.1.3 Modelo Matemático

O modelo matemático referente ao problema estudado, considerando função objetivo e premissas e limitações da empresa estão apresentados a seguir.

Tabela 4 - Parâmetros do Modelo Matemático

Parâmetro	Descrição
$\boldsymbol{J}$	Conjunto de *jobs*
$\boldsymbol{M}$	Conjunto de Máquinas
$\boldsymbol{\theta_{j,m}}$	Fator de alocação do job j na máquina m
$\boldsymbol{\lambda_{j,m}}$	Fator de Sequência do *job* j na máquina m

$\boldsymbol{\beta_j}$	Tamanho de lote do *job j*
$\boldsymbol{\pi}$	*Makespan* total
$\boldsymbol{\tau_m}$	Tempo no qual a máquina m completa o processamento de todos os *jobs*
$\boldsymbol{\rho_j}$	Código da natureza do *job j*
$\boldsymbol{\overline{\rho_m}}$	Código da natureza do último *job* produzido pela máquina m
$\boldsymbol{\sigma_{j,m}}$	Tempo de setup do *job j* na máquina m
$\boldsymbol{\varphi_m}$	Tempo de setup da máquina m
$\boldsymbol{\eta_{j,m}}$	Tempo de produção de cada componente do *job j* na máquina m
$\boldsymbol{\Theta_m}$	Quantidade de produtos sendo processados pela máquina m
$\boldsymbol{\psi_j}$	Quantidade de máquinas processando o *job j*
$\boldsymbol{\phi_{j,m}}$	Quantidade de vezes que é iniciada a produção do o *job j* pela máquina m

Fonte: O Autor (2021)

O objetivo geral é dado por

$$Min\ \pi = Min\ [Max(\tau_m)], \qquad \forall\ m \in M \tag{1}$$

Sujeito a

$$\tau_m = Max \sum_{j=1}^{J} \{\theta_{j,m} \times [\sigma_{j,m} + (\eta_{j,m} \times \beta_j)]\} \tag{2}$$

$$\theta_{j,m} = \begin{cases} 1, & se\ o\ job\ j\ está\ alocado\ na\ máquina\ m \\ 0, & caso\ contrário \end{cases} \tag{3}$$

$$\sigma_{j,m} = \begin{cases} \varphi_m, & se\ \overline{\rho_m} = \rho_j \\ 0, & caso\ contrário \end{cases} \tag{4}$$

$$\Theta_m \leq 1, \qquad \forall\ m \in M \tag{5}$$

$$\psi_j \leq 1, \qquad \forall\ j \in J \tag{6}$$

$$\phi_{j,m} \leq 1, \qquad \forall\ j \in J, \qquad \forall\ m \in M \tag{7}$$

A função objetivo (1) objetiva minimizar o *makespan* total; (2) apresenta a equação que define o tempo de processamento total da máquina como sendo as somas do tempo de *setup* e o tempo de processamento de todos os *jobs*; (3) apresenta a variável binária que define se o

job j será ou não produzido pela máquina m; (4) garante que será considerado o tempo de *setup* das máquinas para a produção de *jobs* de naturezas diferentes; (5) garante que cada máquina pode processar um único *job* por vez; (6) garante que cada *job* será processado por apenas uma única máquina por vez; (7) garante que não é permitido que seja interrompida a produção de um *job*. Ademais, todas as máquinas estarão continuamente disponíveis durante sua jornada de trabalho (i.e., não será considerada quebras de máquinas ou qualquer outro tipo de interrupção na disponibilidade), e o estoque de matéria-prima será considerado infinito.

O impacto dos métodos será considerado pelas variáveis $\theta_{j,m}$ e $\lambda_{j,m}$. O Método 1 retornará como resultado estes valores pré-estabelecidos. Assim, após a otimização, o sistema já terá conhecimento de qual máquina irá produzir quais *jobs* em qual sequência. Como o Método 2 retornará as porcentagens de *jobs* a serem produzidos por cada máquina e a regra de despacho que sequenciará as filas das máquinas, os valores de $\theta_{j,m}$ serão definidos durante a execução da simulação real e, apenas depois da definição das filas de cada máquina, será aplicada a regra de despacho para determinar a prioridade $\lambda_{j,m}$ de cada produto j em cada máquina m.

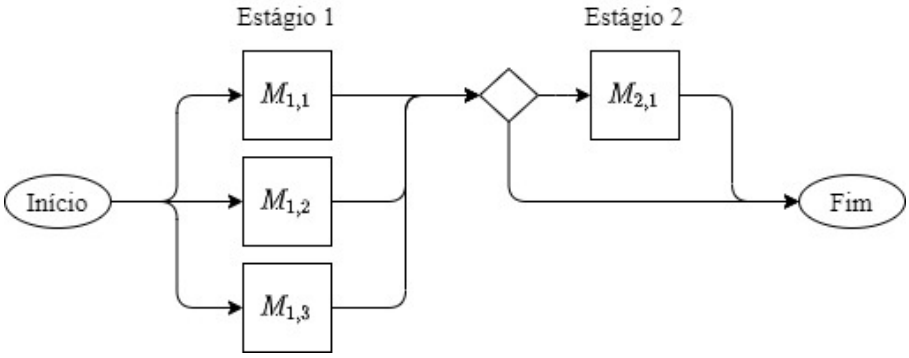
4.2 CASO REAL E CASOS SIMULADOS

Nessa seção serão apresentados o caso real a ser simulado e os modelos de simulação desenvolvidos.

4.2.1 Caso Real

O caso teste utilizado para a comparação dos dois modelos é baseado em uma célula de uma fábrica de motores elétricos situada em Jaraguá do Sul – SC, (empresa ABC). O cenário estudado é a célula responsável pela estampagem das lâminas de estator e rotor dos motores e é configurada conforme apresentado na Figura 17. Seu *layout* é composto por três máquinas iguais (máquinas de estamparia) no primeiro estágio e uma máquina (forno) no segundo estágio. Todos *jobs* produzidos precisam necessariamente ser processados por uma, e somente uma, máquina do estágio 1. Entretanto, apenas alguns *jobs* necessitam ser processados no segundo estágio (dependendo do tipo de aço utilizado para a produção da lâmina).

Figura 17- *Flow Shop* da célula da Fábrica ABC



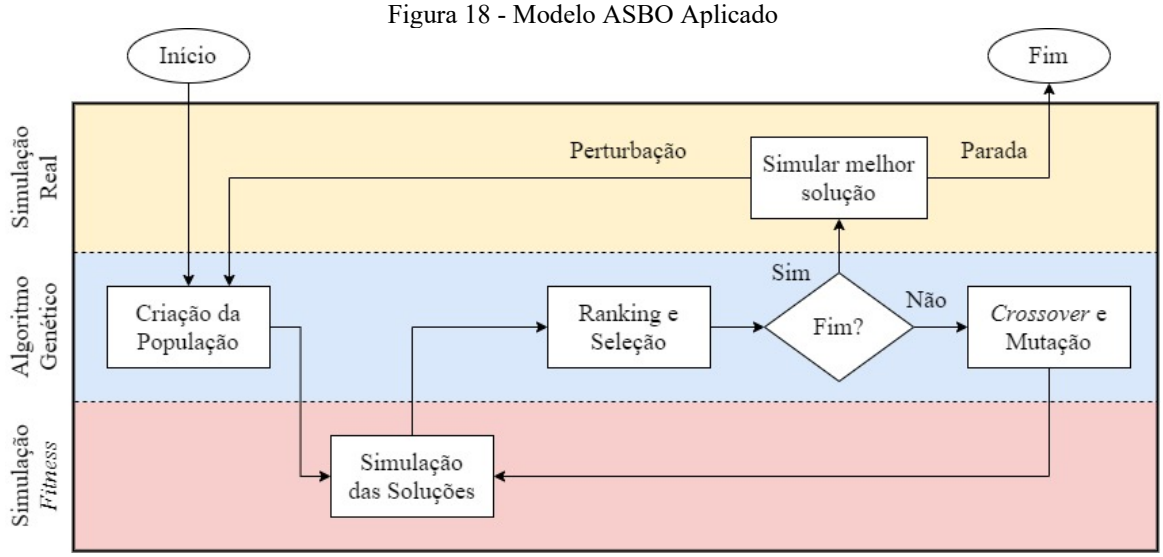
Fonte: Fornecido pela Empresa

Ademais, outros dados fornecidos para a realização desta pesquisa foram: (i) demandas de ordens de produção para os 702 tipos de lâminas diferentes produzidos por essa célula; (ii) tempos de produção; (iii) tempos de setup; (iv) quantidades de máquinas e estágios e (v) disponibilidade das máquinas.

4.2.2 Modelos de Simulação Desenvolvidos

Tanto o modelo do Método 1 quanto o do Método 2 serão implementados no cenário da empresa ABC em um formato ASBO conforme apresentado na Figura 18. No início da simulação, o sistema irá receber as primeiras demandas, iniciando um processo SBO. A otimização retornará, para a simulação do caso real, as variáveis de acordo com a melhor solução encontrada. Assim que receber a solução, a simulação do caso real irá simular a produção para essa solução até que ocorra a chegada de novas demandas (chamadas perturbações no sistema). Cada vez que ocorrer uma perturbação, a antiga programação será desconsiderada e será iniciado um novo SBO a fim determinar o novo melhor sequenciamento dos *jobs*. É importante ressaltar que as perturbações não influenciarão nos *jobs* já processados ou em processamento, i.e., os *jobs* a serem considerados para o novo sequenciamento serão aqueles que estão na fila esperando serem processados.

Figura 18 - Modelo ASBO Aplicado

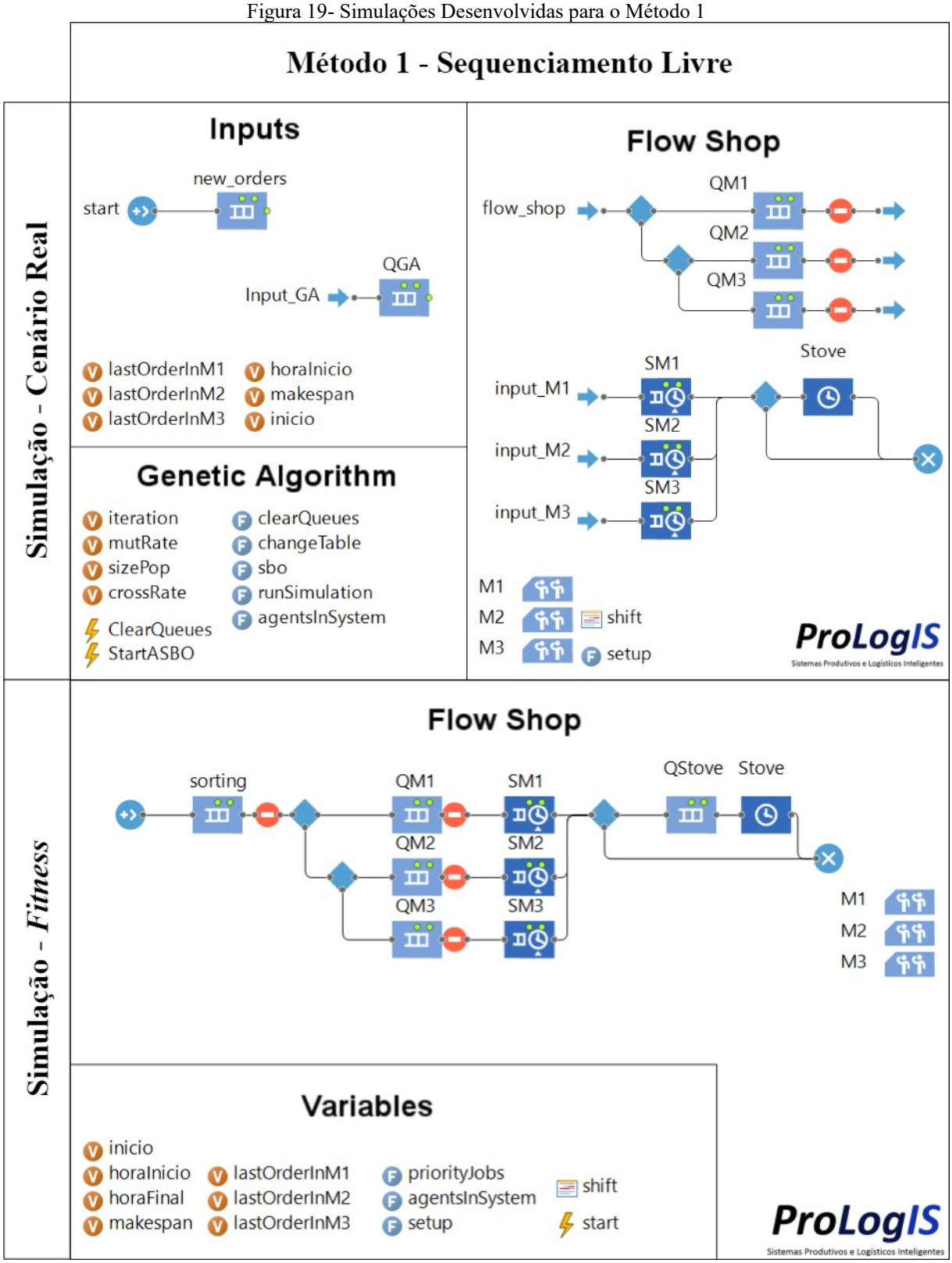


Fonte: baseado em Berger, Zanella e Frazzon (2019)

A Figura 19 apresenta os modelos de simulação desenvolvidos no *software* AnyLogic. Como os dois ASBO's são aplicados no mesmo cenário, os modelos de simulação devem ser semelhantes. As diferenças entre eles consistem em como serão consideradas as variáveis e na forma que as decisões serão tomadas dentro do sistema. Para ambos os modelos foram desenvolvidos dois cenários: o cenário real e o cenário de *fitness*. O primeiro cenário será utilizado para simular a fábrica durante todo o período de 2020, enquanto o *fitness* será utilizado apenas durante a execução do Algoritmo Genético como forma de calcular o *makespan* das soluções. Os códigos desenvolvidos para a aplicação do Algoritmo Genético constam no Anexo A, sendo que os parâmetros utilizados foram:

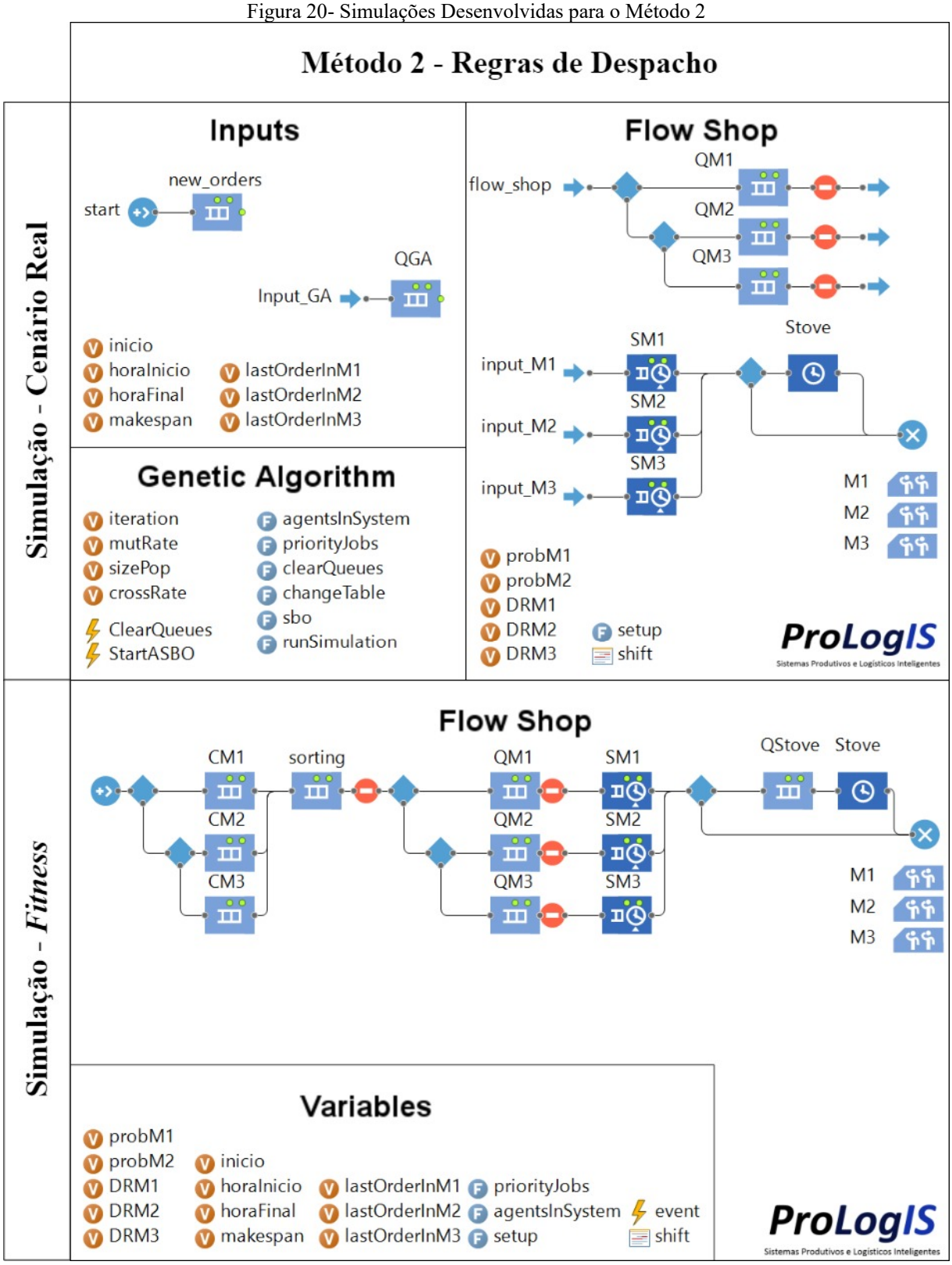
- Tamanho da população: 50;
- Probabilidade de *Crossover*: 80%;
- Probabilidade de Mutação: 10%;
- Número máximo de iterações: 100.

Figura 19- Simulações Desenvolvidas para o Método 1



Fonte: O Autor (2021)

Figura 20- Simulações Desenvolvidas para o Método 2



Fonte: O Autor (2021)

4.3 RESULTADOS DAS SIMULAÇÕES

Serão realizadas quatro análises dos resultados obtidos: o *makespan* mensal para cada um dos dois métodos, o *makespan* anual, o tempo de processamento e a convergência dos modelos. A Tabela 5 apresenta os valores mensais para as duas heurísticas. Para todos os meses o modelo do Método 1 obteve um valor melhor que o encontrado na literatura, alcançando um *makespan* até 29,69% menor.

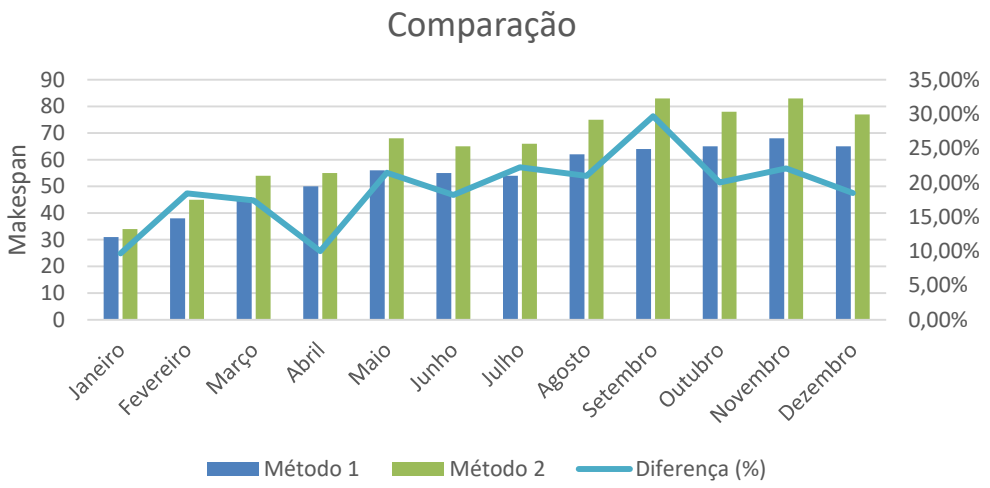
Tabela 5 - *Makespan* mês a mês

Mês	Método 1	Método 2	Diferença (dias)	Diferença (%)
Janeiro	31	34	3	9,68%
Fevereiro	38	45	8	18,42%
Março	46	54	8	17,39%
Abril	50	55	5	10,00%
Maio	56	68	12	21,43%
Junho	55	65	10	18,18%
Julho	54	66	12	22,22%
Agosto	62	75	13	20,97%
Setembro	64	83	19	29,69%
Outubro	65	78	13	20,00%
Novembro	68	83	15	22,06%
Dezembro	65	77	12	18,46%

Fonte: O Autor (2021)

Os resultados apresentam a vantagens do Método 1 frente ao Método 2. A análise mensal é capaz de gerar conclusões importantes para o cenário industrial. As programações para o mês de janeiro resultaram em uma vantagem de três dias para o Método 1 enquanto as programações para o mês de fevereiro resultaram em uma vantagem de 8 dias. É perceptível que a vantagem aumenta com o passar dos meses, chegando ao máximo de 19 dias de produção a mais para o Método 2, conforme apresentado na Figura 21.

Figura 21- Comparação dos Métodos



Fonte: O Autor (2021)

Tal comparação dos meses mostra um aumento de *makespan* ao longo dos meses, o que é justificado pela ineficiência da solução encontrada pelo modelo de comparação, a qual gera um sequenciamento fraco e, consequentemente, gera atrasos na finalização das ordens de produção. Tal atraso faz com que alguns *jobs* não sejam finalizados ao final do mês, sendo considerados para a produção do mês subsequente. Possivelmente a diferença entre os dois métodos aumentaria caso fosse considerado no modelo períodos posteriores devido ao acúmulo de ordens não produzidas.

A comparação do *makespan* anual para ambos os modelos simulados e o tempo de execução total da simulação estão apresentados na Tabela 6. Pode-se perceber que o Método 1 apresenta uma redução de 15 dias no *makespan* anual, além de uma redução de 19 minutos na execução.

Tabela 6- Resultados

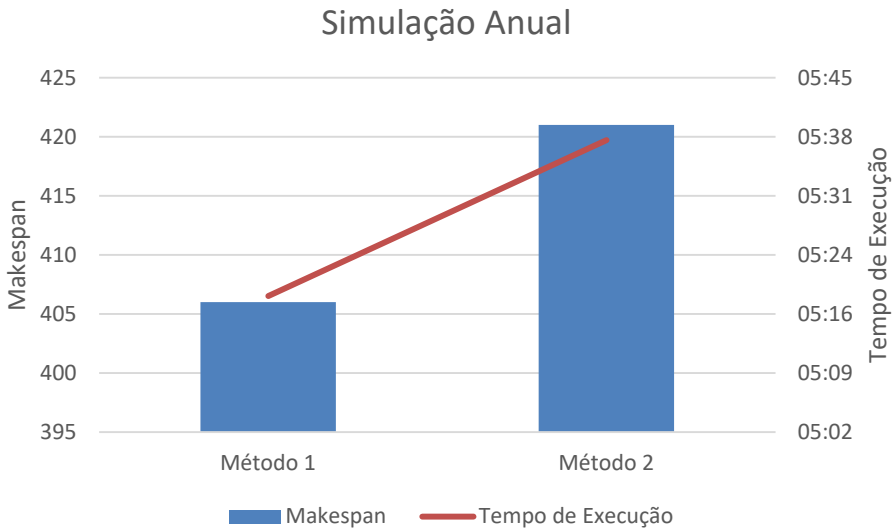
Parâmetro	**Método 1**	**Método 2**	**Diferença**
Makespan	406 dias	421 dias	15 dias
Tempo de Execução	05:19	05:38	00:19

Fonte: O Autor (2021)

O *makespan* anual apresentou uma diferença de quinze dias entre as soluções, o que corresponde a um *makespan* anual 3,69% maior para o Método 2. A primeira consideração a ser feita é em relação ao resultado e a disponibilidade da célula. Como não foram consideradas

as outras linhas de produção que utilizam as mesmas máquinas, estabeleceu-se uma jornada de cinco horas para as máquinas, que acarretou em um *makespan* de 406 dias de produção, i.e., precisaria de um mês e dez dias a mais no ano de 2020 para poder suprir toda a demanda. Todavia, como na prática a disponibilidade é variável, seria necessário realizar um novo SBO a cada mudança de tempos (tanto de ganho quanto de perda de tempo nos turnos das máquinas). Apesar dessa limitação, o modelo se apresentou mais eficiente que o Método 2 e possivelmente retornaria um *makespan* ainda melhor em caso de maior disponibilidade. O tempo de execução do Método 1 também se apresenta como mais uma vantagem, sendo dezenove minutos mais rápido que o modelo de comparação no Método 2, conforme apresenta a Figura 22.

Figura 22 - Resultado da Simulação Anual



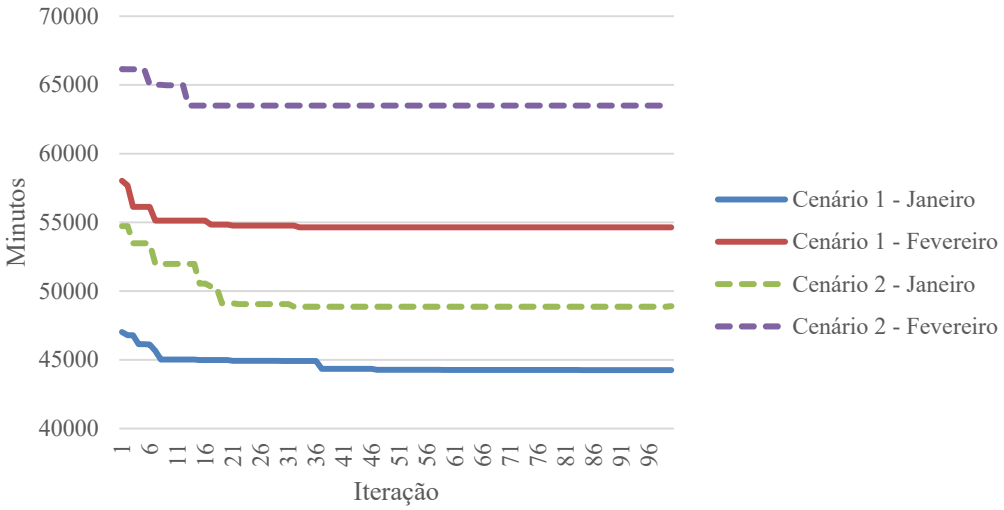
Fonte: O Autor (2021)

O resultado apresentado na Figura 22 demonstra que um método de sequenciamento livre sem regras pré-estabelecidas pode gerar um resultado muito melhor e em um tempo de execução mais curto. No cenário industrial isso se mostra como uma grande vantagem competitiva, pois é possível de se reduzir significativamente o tempo total da produção com um baixo tempo de resposta. Ademais, uma redução no tempo de execução é crucial em ambientes com muita incerteza e volatilidade.

A análise da convergência do algoritmo é importante para compreender a velocidade com que a heurística encontra a melhor solução, além de compreender a otimização da variável ao longo de todas as iterações. Como todos os dados fornecidos ao sistema (tempos de processamento nas máquinas e tempos de setup) foram dados em minutos, se faz necessária a

análise da convergência através do *makespan* em minutos. A Figura 23 apresenta a convergência da otimização em minutos para os dois primeiros meses de 2020. Foram considerados apenas os dois primeiros meses para que o gráfico ficasse mais claro e menos poluído, uma vez que para todos os meses os resultados apresentaram o mesmo padrão.

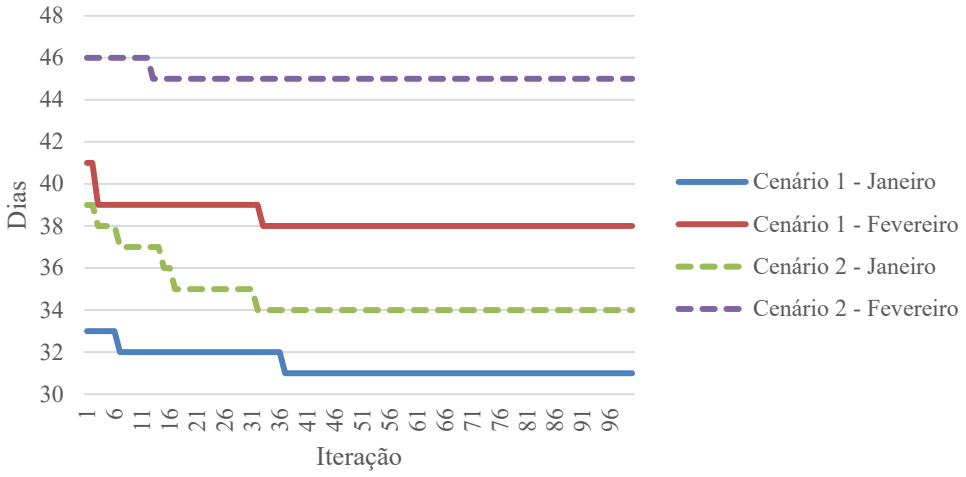
Figura 23 - Convergência em Minutos



Fonte: O Autor (2021)

Outrossim, na Figura 24 apresenta a mesma convergência convertida em dias para os dois primeiros meses de 2020.

Figura 24- Convergência em Dias



Fonte: O Autor (2021)

As curvas de convergência mostram que o valor permanece estagnado por aproximadamente trinta iterações, o que indica que poderia ser reduzido o número máximo de iterações sem prejudicar a eficiência do método e, ainda, reduzindo o tempo de execução. Pode-se destacar também que as cinco horas de execução (para ambos os modelos) foi para a simulação de todo o período de 2020, otimizando o sistema por doze vezes (otimização inicial e onze perturbações com a chegada das demandas mensais), obtendo uma média de 26,58 e 28,17 minutos de para cada otimização mensal do Método 1 e Método 2, respectivamente.

Por fim, pode-se considerar a curva de convergência para os meses de janeiro e fevereiro para ambos os modelos mostra que o Método 1 parte de uma solução melhor desde a primeira iteração. Inclusive, para ambos os cenários, a pior solução encontrada pelo modelo do Método 1 foi melhor que a melhor solução escolhida pelo modelo do Método 2. Isso comprova que o método livre de sequenciamento já inicia encontrando soluções mais eficientes do que a escolha de qualquer regra de despacho. Os resultados de Kuck *et al.* (2017) apresentam que a livre escolha de uma regra de despacho para cada máquina ao invés de uma única regra para todo o *flow shop* reduz significativamente o tempo de ciclo da produção. De fato, o método de escolha de diferentes regras de despacho está englobado pelo método do sequenciamento livre. Por fim, pode-se concluir que os resultados aqui obtidos estão alinhados com os resultados de Kuck *et al.* (2017) e, além, se aprofunda na discussão quanto a uma otimização livre, sem regras limitantes.

5 CONCLUSÃO

O presente trabalho teve como objetivo avaliar comparativamente modelos de otimização adaptativa baseada em simulação para sistemas de produção *flow shop*. Para tanto, uma série de procedimentos foram realizados: inicialmente foi realizada uma revisão da literatura a fim de definir modelos ASBO a serem avaliados; em seguida, foram desenvolvidos os modelos de simulação baseados em um sistema real; com os modelos desenvolvidos, implementou-se os modelos de otimização; por fim, foram executadas as simulações e avaliados comparativamente os desempenhos dos modelos.

Os resultados apresentados se mostram valorosos tanto no cenário acadêmico quanto no industrial. Para a academia, a comparação destaca um método capaz de encontrar valores satisfatórios, com um baixo tempo computacional e que explore uma grande gama de possibilidades. A escolha de se determinar um sequenciamento livre em detrimento de escolher a melhor regra de despacho não se apresenta melhor apenas em uma simulação, mas se mostra superior se comparada com o impacto do acúmulo de *jobs* não processados ao longo prazo. Assim, o trabalho indica um modelo a ser utilizado em futuras pesquisas.

Outrossim, o sequenciamento livre se mostra importante para a indústria. A minimização do *makespan* tem diversas vantagens na prática: menor tempo de resposta, melhor satisfação do cliente, maior disponibilidade das máquinas e funcionários e menores custos com setup. Portanto, o modelo ASBO aplicado, além de gerar vantagem competitiva, garante para a empresa a possibilidade de se programar de forma rápida, em tempo real e considerando todos os parâmetros importantes.

Quanto ao atual cenário da literatura, o presente trabalho apresenta um estudo inovador, buscando de forma inédita comparar modelos de simulação adaptativa baseada em simulação a fim de contrastar resultados e tempo de execução. Estudos dessa natureza se fazem necessário por aprofundar o conhecimento em uma área relativamente recente. Com isso, torna-se possível determinar métodos base de solução para, então, avançar em abordagens mais complexas e aprofundadas no tema.

O presente trabalho apresentou algumas limitações: foi considerado um sistema real relativamente pequeno, com poucos estágios e máquinas, além de ter considerado apenas uma única heurística e um único modelo de simulação. Portanto, para trabalhos futuros, recomenda-se: (i) a aplicação de diferentes meta-heurísticas para o problema proposto; (ii) a utilização de diferentes softwares e/ou linguagens de programação para a comparação dos tempos de

execução; e (iii) a aplicação da metodologia em diferentes cenários para a comparação dos resultados.

REFERÊNCIAS

ABDEL-BASSET, M.; MANOGARAN, G.; EL-SHAHAT, D.; MIRJALILI, S. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. **Future Generation Computer Systems**, v. 85, n. 1, p. 129–145, 2018. Disponível em: https://doi.org/10.1016/j.future.2018.03.020

ABDEL-BASSET, M.; MOHAMED, R.; ABOUHAWWASH, M.; CHAKRABORTTY, R. K.; RYAN, M. J. A simple and effective approach for tackling the permutation flow shop scheduling problem. **Mathematics**, v. 9, n. 3, p. 1–23, 2021. Disponível em: https://doi.org/10.3390/math9030270

AL-SHAYEA, A.; FARARAH, E.; NASR, E. A.; MAHMOUD, H. A. Model for Integrating Production Scheduling and Maintenance Planning of Flow Shop Production System. **IEEE Access**, v. 8, n. 1, p. 208826–208835, 2020. Disponível em: https://doi.org/10.1109/ACCESS.2020.3038719

ANSARI, R. Dynamic Optimization for Analyzing Effects of Multiple Resource Failures on Project Schedule Robustness. **KSCE Journal of Civil Engineering**, v. 25, n. 5, p. 1515–1532, 2021. Disponível em: https://doi.org/10.1007/s12205-021-0564-1

AZAB, A.; KARAM, A.; ELTAWIL, A. A simulation-based optimization approach for external trucks appointment scheduling in container terminals. **International Journal of Modelling and Simulation**, v. 40, n. 5, p. 321–338, 2019. Disponível em: https://doi.org/10.1080/02286203.2019.1615261

AZIMI, P.; SHOLEKAR, S. A simulation optimization approach for the multi-objective multi-mode resource constraint project scheduling problem. **International Journal of Industrial Engineering and Production Research**, v. 32, n. 1, p. 37–45, 2021. Disponível em: https://doi.org/10.22068/ijiepr.32.1.37

BARBIERI, G.; BERTUZZI, A.; CAPRIOTTI, A.; RAGAZZINI, L.; GUTIERREZ, D.; NEGRI, E.; FUMAGALLI, L. A virtual commissioning based methodology to integrate digital twins into manufacturing systems. **Production Engineering**, v. 15, n. 3–4, p. 397–412, 2021. Disponível em: https://doi.org/10.1007/s11740-021-01037-3

BERGER, S. L. T.; ZANELLA, R. M.; FRAZZON, E. M. Towards a data-driven predictive-reactive production scheduling approach based on inventory availability. **IFAC-PapersOnLine**, v. 52, n. 13, p. 1343–1348, 2019. Disponível em: https://doi.org/10.1016/j.ifacol.2019.11.385

BEWOOR, L. A.; PRAKASH, V. C.; SAPKAL, S. U. Production scheduling optimization in foundry using hybrid Particle Swarm Optimization algorithm. **Procedia Manufacturing**, v. 22, p. 57–64, 2018. Disponível em: https://doi.org/10.1016/j.promfg.2018.03.010

CASTAÑÉ, G. G.; SIMONIS, H.; BROWN, K. N.; LIN, Y.; OZTURK, C.; GARRAFFA, M.; ANTUNES, M. Simulation-Based Optimization Toll for Field Service Planning. *In*: 2019, **Proceedings of the 2019 Winter Simulation Conference**. *[S. l.: s. n.]* p. 1684–1695.

CASTRO, V. F. de; FRAZZON, E. M. Benchmarking of best practices: an overview of the academic literature. **Benchmarking**, v. 24, n. 3, p. 750–774, 2017. Disponível em: https://doi.org/10.1108/BIJ-03-2016-0031

CHEN, J.-S.; PAN, J. C.-H.; LIN, C.-M. A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. **Expert Systems with Applications**, v. 34, n. 1, p. 570–577, 2008. Disponível em: https://doi.org/10.1016/j.eswa.2006.09.021

DAI, M.; TANG, D.; GIRET, A.; SALIDO, M. A.; LI, W. D. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. **Robotics and Computer-Integrated Manufacturing**, v. 29, n. 5, p. 418–429, 2013. Disponível em: https://doi.org/10.1016/j.rcim.2013.04.001

DELLA CROCE, F.; TADEI, R.; VOLTA, G. A genetic algorithm for the job shop problem. **Computers and Operations Research**, v. 22, n. 1, p. 15–24, 1995. Disponível em: https://doi.org/10.1016/0305-0548(93)E0015-L

DOLGUI, A.; IVANOV, D.; SETHI, S. P.; SOKOLOV, B. Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. **International Journal of Production Research**, v. 57, n. 2, p. 411–432, 2019. Disponível em: https://doi.org/10.1080/00207543.2018.1442948

EHRENBERG, C.; ZIMMERMANN, J. Simulation-Based Optimization in Make-To-Order Production: Scheduling for a Special-Purpose Glass Manufacturer. *In*: 2012, Berlin - Germany. **Winter Simulation Conference**. Berlin - Germany: *[s. n.]*, 2012. p. 1–12.

ENGIN, O.; CERAN, G.; YILMAZ, M. K. An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. **Applied Soft Computing**, v. 11, n. 3, p. 3056–3065, 2011. Disponível em: https://doi.org/10.1016/j.asoc.2010.12.006

ERMOLIEVA, T. Simulation-based optimization of social security systems under uncertainty. **European Journal of Operational Research**, v. 166, n. 3, p. 782–793, 2005. Disponível em: https://doi.org/10.1016/j.ejor.2004.03.044

FEI, Y.; MA, H. Multi-objective joint optimization of batch-discrete hybrid flow shop scheduling integrated with machine maintenance. **2018 5th International Conference on Industrial Engineering and Applications, ICIEA 2018**, p. 247–253, 2018. Disponível em: https://doi.org/10.1109/IEA.2018.8387105

FENG, K.; RAO, U. S.; RATURI, A. Setting planned orders in master production scheduling under demand uncertainty. **International Journal of Production Research**, v. 49, n. 13, p. 4007–4025, 2011. Disponível em: https://doi.org/10.1080/00207543.2010.495955

FRAZZON, E. M.; ALBRECHT, A.; HURTADO, P. A. Simulation-based optimization for the integrated scheduling of production and logistic systems. **IFAC-PapersOnLine**, v. 49, n. 12, p. 1050–1055, 2016. Disponível em: https://doi.org/10.1016/j.ifacol.2016.07.581

GHALEB, M. A.; SURYAHATMAJA, U. S.; ALHARKAN, I. M. Metaheuristics for two-stage no-wait flexible flow shop scheduling problem. *In*: 2015, Dubai. **IEOM 2015 - 5th**

International Conference on Industrial Engineering and Operations Management, Proceeding. Dubai: *[s. n.]*, 2015. p. 1–9. Disponível em: https://doi.org/10.1109/IEOM.2015.7093943

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Editora Atlas S.A., 2002. v. 38

GUPTA, J. N. D. A general algorithm for the n×M flowshop scheduling problem. **International Journal of Production Research**, v. 7, n. 3, p. 241–247, 1968. Disponível em: https://doi.org/10.1080/00207546808929814

HEINZL, B.; KASTNER, W. A general variable neighborhood search for simulation-based energy-aware flow shop scheduling. *In*: 2020, **Simulation Series**. *[S. l.: s. n.]* p. 115–126.

HOLLAND, J. H. Genetic Algorithms and The Optimal Allocation of Trials. **Society for Industrial and Applied Mathematics**, v. 2, n. 2, p. 88–105, 1973. Disponível em: https://doi.org/10.7551/mitpress/1090.003.0008

IDZIKOWSKI, R.; RUDY, J.; GNATOWSKI, A. Solving non-permutation flow shop scheduling problem with time couplings. **Applied Sciences**, v. 11, n. 10, p. 1–18, 2021. Disponível em: https://doi.org/10.3390/app11104425

IRAWAN, C. A.; ESKANDARPOUR, M.; OUELHADJ, D.; JONES, D. Simulation-based optimisation for stochastic maintenance routing in an offshore wind farm. **European Journal of Operational Research**, v. 289, n. 3, p. 912–926, 2021. Disponível em: https://doi.org/10.1016/j.ejor.2019.08.032

JOHNSON, S. M. Optimal Two- and Three-Stage Production Schedules With Setup Times Included. **Naval Research Logistics Quarterly**, v. 1, p. 61–68, 1954.

KARACAN, I.; KARACAN, I.; SENVAR, O.; BULKAN, S. An integrated solution approach for flow shop scheduling. **Technical Gazette**, v. 28, n. 3, p. 786–795, 2021. Disponível em: https://doi.org/10.17559/TV-20200208192653

KESKIN, K.; ENGIN, O. A hybrid genetic local and global search algorithm for solving no-wait flow shop problem with bi criteria. **SN Applied Sciences**, v. 3, n. 6, 2021. Disponível em: https://doi.org/10.1007/s42452-021-04615-3

KÜCK, M.; EHM, J.; HILDEBRANDT, T.; FREITAG, M.; ENZO M. FRAZZON. Potential of Data-Driven Simulation-Based Optimization for Adaptative Scheduling and Control of Dynamic Manufacturing Systems. *In*: 2016, USA. **Proceedings of the 2016 Winter Simulation Conference**. USA: *[s. n.]*, 2016. p. 2820–2831.

KUCK, M.; EIKE, B.; FREITAG, M.; HILDEBRANDT, T.; FRAZZON, E. M. Towards Adaptive Simulation - Based Optimization To Select Individual. *In*: 2017, Las Vegas. **Winter simulation conference**. Las Vegas: *[s. n.]*, 2017. p. 3852–3863.

KÜHN, M.; VÖLKER, M.; SCHMIDT, T. An algorithm for efficient generation of customized priority rules for production control in project manufacturing with stochastic job processing times. **Algorithms**, v. 13, n. 12, p. 1–20, 2020. Disponível em:

https://doi.org/10.3390/a13120337

KUMAR, S.; LAD, B. K. Integrated production and maintenance planning for parallel machine system considering cost of rejection. **Journal of the Operational Research Society**, v. 68, n. 7, p. 1–13, 2016. Disponível em: https://doi.org/10.1057/jors.2016.46

LEBBAR, G.; EL ABBASSI, I.; JABRI, A.; EL BARKANY, A.; DARCHERIF, M. Multi-criteria blocking flow shop scheduling problems: Formulation and performance analysis. **Advances in Production Engineering And Management**, v. 13, n. 2, p. 136–146, 2018. Disponível em: https://doi.org/10.14743/apem2018.2.279

LI, J.; GUO, L.; LI, Y.; LIU, C.; WANG, L.; HU, H. Enhancing whale optimization algorithm with chaotic theory for permutation flow shop scheduling problem. **International Journal of Computational Intelligence Systems**, v. 14, n. 1, p. 651–675, 2021. Disponível em: https://doi.org/10.2991/ijcis.d.210112.002

LI, S.; KOOLE, G.; JOUINI, O. A Simple Solution for Optimizing Weekly Agent Scheduling in a Multi-Skill Multi-Channel Contact Center. **Proceedings - Winter Simulation Conference**, v. 2019- Decem, p. 3657–3668, 2019. Disponível em: https://doi.org/10.1109/WSC40007.2019.9004714

LI, S.; KOOLE, G.; XIE, X. An adaptive priority policy for radiotherapy scheduling. **Flexible Services and Manufacturing Journal**, v. 32, n. 1, p. 154–180, 2020. Disponível em: https://doi.org/10.1007/s10696-019-09373-4

LIU, B.; WANG, L.; JIN, Y. H. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. **Computers and Operations Research**, v. 35, n. 9, p. 2791–2806, 2008. Disponível em: https://doi.org/10.1016/j.cor.2006.12.013

LUO, H.; DU, B.; HUANG, G. Q.; CHEN, H.; LI, X. Hybrid flow shop scheduling considering machine electricity consumption cost. **International Journal of Production Economics**, v. 146, n. 2, p. 423–439, 2013. Disponível em: https://doi.org/10.1016/j.ijpe.2013.01.028

MAZZUCO, D. E.; CARREIRÃO DANIELLI, A. M.; OLIVEIRA, D. L.; SANTOS, P. P. P.; PEREIRA, M. M.; COELHO, L. C.; FRAZZON, E. M. A concept for simulation-based optimization in Vehicle Routing Problems. **IFAC-PapersOnLine**, v. 51, n. 11, p. 1720–1725, 2018. Disponível em: https://doi.org/10.1016/j.ifacol.2018.08.208

MIGUEL, P. A. C.; FLEURY, A.; MELLO, C. H. P.; NAKANO, D. N.; TURRIONI, J. B.; HO, L. L.; MORABITO, R.; MARTINS, R. A.; PUREZA, V. **Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações**. 1. ed. Rio de Janeiro: Elsevier, 2010.

MUHARNI, Y.; FEBIANTI, E.; HANIFA; ARLIANUR. Production scheduling of bar mill using the combination of particle swarm optimization and Nawaz enscore ham for minimizing makespan in steel company. **AIP Conference Proceedings**, v. 2114, n. 1, p. 5, 2019. Disponível em: https://doi.org/10.1063/1.5112410

MURATA, T.; ISHIBUCHI, H. Performance Evaluation of Genetic Algorithms for Flowshop Scheduling Problems. *In*: 1994, Orlando, USA. **Proceedings of the First IEEE**

Conference on Evolutionary Computation. Orlando, USA: *[s. n.]*, 1994. p. 812–817. Disponível em: https://doi.org/10.1016/0360-8352(96)00053-8

ONWUBOLU, G.; DAVENDRA, D. Scheduling flow shops using differential evolution algorithm. **European Journal of Operational Research**, v. 171, n. 2, p. 674–692, 2006. Disponível em: https://doi.org/10.1016/j.ejor.2004.08.043

PALMER, D. S. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining a Near Optimum. **Operational Research Quarterly**, v. 16, n. 1, p. 101–107, 1965. Disponível em: https://doi.org/https://doi.org/10.1057/jors.1965.8

PAPROCKA, I.; KRENCZYK, D.; BURDUK, A. The method of production scheduling with uncertainties using the ants colony optimisation. **Applied Sciences**, v. 11, n. 171, p. 1–14, 2021. Disponível em: https://doi.org/10.3390/app11010171

PIRES, M. C.; FRAZZON, E. M.; CARREIRÃO DANIELLI, A. M.; KÜCK, M.; FREITAG, M. Towards a simulation-based optimization approach to integrate supply chain planning and control. **Procedia CIRP**, v. 72, p. 520–525, 2018. Disponível em: https://doi.org/10.1016/j.procir.2018.03.288

PIRES, M. C.; FRAZZON, E. M.; QUADRAS, D.; BRODA, E.; FREITAG, M. Simulation-based optimization for the integrated control of production and logistics: A performance comparison. **IFAC-PapersOnLine**, v. 53, n. 2, p. 10639–10644, 2020. Disponível em: https://doi.org/10.1016/j.ifacol.2020.12.2824

PUROHIT, B. S.; KUMAR, S.; LAD, B. K.; MANJREKAR, V.; SINGH, V. Optimization of multi-item operation sequences and batch size for non-parallel capacitated machines: A case study. **International Journal of Performability Engineering**, v. 13, n. 5, p. 557–568, 2017. Disponível em: https://doi.org/10.23940/ijpe.17.05.p1.557568

RAHMAN, H. F.; SARKER, R.; ESSAM, D. A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM**, v. 31, n. 1, p. 87–103, 2017. Disponível em: https://doi.org/10.1017/S0890060416000196

RICHTER, M.; WINKLER, H. Flow shop scheduling optimization in the chipboard industry: A simulation-based analysis using priority dispatching rules. *In*: 2017, Lancaster, Pennsylvania. **24th International Conference on Production Research, ICPR 2017**. Lancaster, Pennsylvania: *[s. n.]*, 2017. p. 296–301. Disponível em: https://doi.org/10.12783/dtetr/icpr2017/17625

SARAÇOĞLU, İ.; SÜER, G. A.; GANNON, P. Minimizing makespan and flowtime in a parallel multi-stage cellular manufacturing company. **Robotics and Computer-Integrated Manufacturing**, v. 72, n. 102182, p. 1–11, 2021. Disponível em: https://doi.org/10.1016/j.rcim.2021.102182

SHAKIBAYIFAR, M.; HASSANNAYEBI, E.; MIRZAHOSSEIN, H.; TAGHIKHAH, F.; JAFARPUR, A. An intelligent simulation platform for train traffic control under disturbance. **International Journal of Modelling and Simulation**, v. 39, n. 3, p. 135–156, 2019. Disponível em: https://doi.org/10.1080/02286203.2018.1488110

TRISKA, Y.; FRAZZON, E. M.; SILVA, V. M. D. Proposition of a simulation-based method for port capacity assessment and expansion planning. **Simulation Modelling Practice and Theory**, v. 103, n. April, p. 102098, 2020. Disponível em: https://doi.org/10.1016/j.simpat.2020.102098

TURAN, H. H.; ATMIS, M.; KOSANOGLU, F.; ELSAWAH, S.; RYAN, M. J. A risk-averse simulation-based approach for a joint optimization of workforce capacity, spare part stocks and scheduling priorities in maintenance planning. **Reliability Engineering and System Safety**, v. 204, n. July, p. 107199, 2020. Disponível em: https://doi.org/10.1016/j.ress.2020.107199

UHLMANN, I. R.; FRAZZON, E. M. Production rescheduling review: Opportunities for industrial integration and practical applications. **Journal of Manufacturing Systems**, v. 49, n. September, p. 186–193, 2018. Disponível em: https://doi.org/10.1016/j.jmsy.2018.10.004

VÖLKER, S.; GMILKOWSKY, P. Reduced discrete-event simulation models for medium-term production scheduling. **Systems Analysis Modelling Simulation**, v. 43, n. 7, p. 867–883, 2003. Disponível em: https://doi.org/10.1080/0232929031000075313

XIE, H.; ZHANG, Y.; WU, Z.; LV, T. A Bibliometric Analysis on Land Degradation: Current Status, Development and Future Directions. **Land**, v. 9, n. 28, p. 2–37, 2020.

ZHAI, Y.; BIEL, K.; ZHAO, F.; SUTHERLAND, J. W. Dynamic scheduling of a flow shop with on-site wind generation for energy cost reduction under real time electricity pricing. **CIRP Annals - Manufacturing Technology**, v. 66, n. 1, p. 41–44, 2017. Disponível em: https://doi.org/10.1016/j.cirp.2017.04.099

ZHANG, T.; ROSE, O. Simulation-based overhead-crane scheduling for a manufacturing plant. *In*: 2013, WAshington DC - USA. **Winter Simulation Conference**. WAshington DC - USA: IEEE, 2013. p. 2633–2642. Disponível em: https://doi.org/10.1109/WSC.2013.6721635

ZHANG, X.-Y.; CHEN, L. Heuristics for minimizing the total tardiness in a re-entrant hybrid flow shop with non-identical machines in parallel. *In*: 2016, **IEEE International Conference on Industrial Engineering and Engineering Management**. *[S. l.: s. n.]* p. 987–991. Disponível em: https://doi.org/10.1109/IEEM.2016.7798025

ZUO, Y.; WANG, Y.; LAILI, Y.; LIAO, T. W.; TAO, F. An evolutionary algorithm recommendation method with a case study in flow shop scheduling. **International Journal of Advanced Manufacturing Technology**, v. 109, n. 3–4, p. 781–796, 2020. Disponível em: https://doi.org/10.1007/s00170-020-05471-y

APÊNDICE A – CÓDIGO DO ALGORITMO GENÉTICO

Classe do Indivíduo:

```
import java.util.Arrays;

/**
 * @author Djonathan Quadras
 *
 * Contém os parâmetros e métodos referentes a cada
 * Indivíduo na População
 *
 */

public class Individual implements Comparable<Individual>{

   double[] cromossomo;
   double fitness;

   public Individual(double[] cromo, double fitness){
      this.cromossomo = cromo;
      this.fitness = fitness;
   }

   public Individual(Individual individual){
      this.cromossomo = Arrays.copyOf(individual.getCromossomo(),
individual.getCromossomo().length);
      this.fitness = individual.getFitness();
   }

   public Individual(double[] cromo){
      this.cromossomo = cromo;
      this.fitness = 0;
   }

   public Individual(){}

   public double getFitness() {
      return fitness;
   }

   public double[] getCromossomo() {
      return cromossomo;
   }

   public void setFitness(double fitness) {
      this.fitness = fitness;
   }
```

```
    public void setCromossomo(double[] cromossomo) {
        this.cromossomo = cromossomo;
    }

    public void printIndividuo(){
        System.out.println("Cromossomo = " + Arrays.toString(this.cromossomo));
        System.out.println("Fitness = " + this.fitness);
    }

    @Override
    public int compareTo(Individual individual) {

        double compareQuantity = ((Individual) individual).getFitness();
        double thisCost = this.fitness;

        //ascending order
        return (int) Math.round(thisCost - compareQuantity);
    }
}
```

Classe da População:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;


/**
 * @author Djonathan Quadras
 *
 * Contém os parâmetros e métodos referentes a População
 *
 */

public class Population {

    private ArrayList<Individual> population;

    public Population(){
        ArrayList<Individual> pop = new ArrayList<>();
        this.population = pop;
    }

    public Population(ArrayList<Individual> pop){
        this.population = pop;
    }

    public void setPopulation(ArrayList<Individual> population) {
```

```
        this.population = population;
    }

    public void setIndividual(Population pop, double[] cromossomo, double fitness){
        pop.getPopulation().add(new Individual(cromossomo, fitness));
    }

    public void setIndividual(Population pop, double[] cromossomo){
        pop.getPopulation().add(new Individual(cromossomo, 0));
    }

    public ArrayList<Individual> getPopulation() {
        return population;
    }

    public Individual getIndividual(int i) {
        return this.population.get(i);
    }

    public static void printPopulation(Population pop){
        ArrayList<Individual> population = pop.getPopulation();
        for(int i = 0; i < population.size(); i++){
            System.out.println("Individuo " + (i+1) + ": Cromossomo = " +
Arrays.toString(population.get(i).getCromossomo()) +
                    "     Custo = " + population.get(i).getFitness());
        }
        System.out.println("");

    }

    public static void ranking(Population population){
        ArrayList<Individual> pop = population.getPopulation();
        Collections.sort(pop);
    }

    @Deprecated
    public void ranking(){
        Collections.sort(this.getPopulation());
    }

    public Population populationRanking(){
        ArrayList<Individual> oldPop = new ArrayList<Individual>(this.getPopulation());
        ArrayList<Individual> rankedPop = new ArrayList<>();

        while(oldPop.size() > 1){
            Individual bestFitness = new Individual(oldPop.get(0));
            int position = 0;
            for(int i = 1; i < oldPop.size(); i++){
                if(bestFitness.getFitness() > oldPop.get(i).getFitness()){
```

```
                position = i;
                bestFitness = new Individual(oldPop.get(i));
            }
        }
        rankedPop.add(bestFitness);
        oldPop.remove(position);
    }
    rankedPop.add(new Individual(oldPop.get(0)));


    return(new Population(rankedPop));
  }
}
```

Classe do Algoritmo Genético:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

/**
 * @author Djonathan Quadras
 *
 * Aplicação do GA, separada em três partes.
 * A parte 1 cria a população.
 * A parte 2 realiza crossover e mutação.
 * A parte 3 classifica e avança a geração.
 * Da parte 2 a população é enviada para a simulação.
 * A simulação envia a população para a parte 3.
 * Caso não tenha sido encerrado o GA, a parte 3 envia a nova
 * população para a parte 2.
 *
 */

public class GA {

  public static Population partI(
        Population population,
        int[] upperLimit,
        double crossoverRate,
        double mutProb,
        int[] lowerLimit){

    Population sons = crossover(population, crossoverRate);
    Population mutated = mutation(population, sons, mutProb, lowerLimit, upperLimit);
    return mutated;
  }
```

```
    public static Population partII(Population pop, int sizePop){

        Population population = pop.populationRanking();
        population = nextGeneration(population, sizePop);
        return population;
    }

    public static Population createPopulation(int sizePop, int[] upperLimit){
        Random random = new Random(); //Necessary to create random values
        Population population = new Population();
        for(int j = 0; j < sizePop; j++){
            double[] cromossomo = new double[upperLimit.length]; //Create the individuals
            for(int i = 0; i < upperLimit.length; i++){cromossomo[i] =
random.nextInt(upperLimit[i])+1;}
            population.setIndividual(population, cromossomo);
        }
        return (population);
    }

    private static Population crossover(Population popReal, double crossoverRate) {
        Random random = new Random(); //To reach random numbers
        ArrayList<Individual> pop = popReal.getPopulation();
        Population population = new Population();
        int crossoverTimes = (int) Math.round(pop.size()*crossoverRate);

        for(int i = 0; i < crossoverTimes; i++){
            int randInd1; int randInd2;
            double[] newcromo1 = new double[pop.get(1).getCromossomo().length];
            double[] newcromo2 = new double[pop.get(1).getCromossomo().length];
            int crossPoint;
            Individual son1 = new Individual();
            Individual son2 = new Individual();

            do{
                randInd1 = random.nextInt(pop.size()); // Select the first individual
                randInd2 = random.nextInt(pop.size()); // Select the second individual
            }while(randInd1 == randInd2);

            double[] cromo1 = pop.get(randInd1).getCromossomo();
            double[] cromo2 = pop.get(randInd2).getCromossomo();

            crossPoint = random.nextInt(cromo1.length - 2);
            if (crossPoint <= 1){crossPoint = crossPoint + 2;}

            for(int j = 0; j < cromo1.length; j++){
                if(j < crossPoint){
                    newcromo1[j] = cromo1[j];
                    newcromo2[j] = cromo2[j];
                }
```

```
            if (j >= crossPoint){
                newcromo1[j] = cromo2[j];
                newcromo2[j] = cromo1[j];
            }
        }
        population.setIndividual(population, newcromo1);
        population.setIndividual(population, newcromo2);
    }
    return(population);

}

   private static Population mutation(Population parents, Population sons, double mutProb,
int[] lowerLimit, int[] upperLimit){

    Random random = new Random();
    ArrayList<Individual> pop = sons.getPopulation();
    ArrayList<Individual> popParents = parents.getPopulation();
    Population mutatedSons = new Population();
    for(int i = 0; i < popParents.size(); i++){
        mutatedSons.setIndividual(mutatedSons, popParents.get(i).getCromossomo(),
popParents.get(i).getFitness());
    }

    int[] valores = {-3, -2, -1, 1, 2, 3};

    for(int i = 0; i < pop.size(); i++){
        double prob2 = random.nextDouble();
        double[] cromossomo = Arrays.copyOf(pop.get(i).getCromossomo(),
pop.get(i).getCromossomo().length);

        if(prob2 <= mutProb){
            int position = random.nextInt(cromossomo.length);
            int alfa = valores[random.nextInt(valores.length)];
            cromossomo[position] =  cromossomo[position] + alfa;
            if(cromossomo[position] > upperLimit[position]){
                cromossomo[position] = upperLimit[position];
            }
            if(cromossomo[position] <= lowerLimit[position]){
                cromossomo[position] = lowerLimit[position];
            }
            mutatedSons.setIndividual(mutatedSons, cromossomo);
        } else {
            mutatedSons.setIndividual(mutatedSons, pop.get(i).getCromossomo());
        }
    }
    return(mutatedSons);
}
```

```
    public static Population nextGeneration(Population pop, int popSize){
        Population nextGeneration = new Population();
        ArrayList<Individual> newPop = new
ArrayList<Individual>(pop.getPopulation().subList(0,popSize));
        nextGeneration.setPopulation(newPop);
        return nextGeneration;
    }

}
```