



## **PROBLEMA DE PROGRAMAÇÃO DE TAREFAS EM AMBIENTE JOB SHOP FLEXÍVEL: UMA REVISÃO BIBLIOGRÁFICA INTEGRATIVA (2024)**

Nicolle Barbosa Nacamura (IC) e Hanna Pamplona Hortencio (Orientador)

**Apoio:** PIBIC Mackpesquisa

### **RESUMO**

A Programação de Tarefas, ou Scheduling, em inglês, é o processo de alocar tarefas a recursos limitados ao longo do tempo para otimizar uma função objetivo. O ambiente de máquinas Job Shop Flexível (JSF), é uma extensão do problema Job Shop (JS) clássico, introduzindo flexibilidade no processo. São três os principais tipos de flexibilidade que podem ser encontrados na literatura: sequencial, operacional e de roteamento. A flexibilidade sequencial permite a modificação na ordem das operações, a flexibilidade operacional permite que várias máquinas estejam aptas a realizar uma operação, e a flexibilidade de roteamento possibilita que uma tarefa possa escolher entre vários caminhos para seguir nas máquinas. A adição da flexibilidade no problema torna o JSF um problema ainda mais complexo do que o JS clássico, que já é NP-hard, independente das restrições e função objetivo utilizadas. Pinedo (2016) sugere uma classificação dos problemas de programação de tarefas em três campos: alfa ( $\alpha$ ), que define o ambiente de máquinas; beta ( $\beta$ ), que descreve as restrições; e gama ( $\gamma$ ), que indica a função objetivo. A revisão bibliográfica proposta neste trabalho visa compilar e analisar as principais definições, funções objetivo, restrições e métodos de solução para o problema de programação de tarefas no JSF. Além disso, o trabalho se propõe a realizar uma análise das principais conclusões e sugestões de trabalhos futuros encontradas nas revisões de literatura de Xie et al. (2019) e Dauzère-Pérès et al. (2024).

**Palavras-chave:** Programação de tarefas, Job Shop, Job Shop Flexível.

### **ABSTRACT**

The Scheduling Problem is the process of allocating tasks to limited resources over time to optimize an objective function. The Flexible Job Shop (JSF) environment is an extension of the classic Job Shop (JS) problem, introducing flexibility into the process. There are three main types of flexibility found in the literature: sequential, operational, and routing. Sequential flexibility allows for modifications in the order of operations, operational flexibility permits multiple machines to be capable of performing a single operation, and routing flexibility enables a task to choose from various routes between machines. The addition of flexibility makes JSF a problem even more complex than the classic JS, which is already NP-hard, regardless of the constraints and objective functions applied. Pinedo (2016) proposes a classification of scheduling problems into three fields: alpha ( $\alpha$ ), which defines the machine environment; beta



( $\beta$ ), which describes the constraints; and gamma ( $\gamma$ ), which indicates the objective function.

The bibliographic review proposed in this work aims to compile and analyze the main definitions, objective functions, constraints, and solution methods for the scheduling problem in JSF. Additionally, the work intends to provide an analysis of the main conclusions and suggestions for future research found in the literature reviews by Xie et al. (2019) and Dautère-Pérès et al. (2024).

**Keywords:** Scheduling Problem, Job Shop, Flexible Job Shop

## 1. INTRODUÇÃO

A Programação de Tarefas (em inglês, scheduling), refere-se ao processo de atribuir tarefas a recursos limitados ao longo do tempo, buscando otimizar determinada função objetivo, segundo Xie et al. (2019). É um problema que desempenha um papel importante em indústrias de fabricação, eficácia de processos operacionais, processos logísticos e áreas rotineiras, sendo um processo crucial de tomada de decisão.

A Programação de Tarefas em ambiente Job Shop (JS) é um problema NP-hard que envolve um conjunto finito de tarefas a serem executadas por um conjunto finito de máquinas. O Problema de Job Shop clássico pode ser definido como um conjunto (J) de tarefas, um conjunto (M) de máquinas e um conjunto (O) de operações. Cada tarefa (j) é uma sequência de ( $n_j$ ) operações consecutivas no conjunto (O), e cada operação ( $i \in O$ ) pode ser processada em alguma máquina compatível. O tempo de processamento da operação (i) na máquina ( $k \in R_i$ ) é dado por ( $p_{ik}$ ). Cada operação (i) possui um predecessor ( $ps(i)$ ) e um sucessor direto ( $sd(i)$ ), de acordo com Pinedo (2002).

Segundo Dauzere-Peres, et al., (2024), o Job Shop clássico considera as seguintes premissas: Todas as máquinas e tarefas devem estar disponíveis no instante zero. Cada operação precisa ser concluída sem interrupção quando iniciada. E por último, as máquinas só podem realizar uma operação por vez.

Uma extensão do JS é o Job Shop Flexível (JSF), onde é adicionada uma flexibilidade no processo. Esse ambiente de máquinas pode apresentar três tipos de flexibilidade: sequencial, operacional e roteamento.

De acordo com Pinedo (2002), existem diversos métodos de resolução que podem ser estudados, explorados e aplicados para resolver esses problemas. Esses métodos geralmente empregam técnicas de enumeração implícita, como Branch-and-Bound e Branch-and-Cut, para encontrar a solução ótima do problema.

Pinedo (2016) sugere uma classificação dos problemas de programação de tarefas com objetivo de facilitar a compreensão dos pesquisadores e interessados da área. Essa classificação possui três campos  $|\alpha||\beta||\gamma|$  que indicam características específicas do problema.

O primeiro campo, denominado alfa ( $\alpha$ ) define o ambiente de máquina em que o problema está inserido, ou seja, o cenário específico em que os recursos são organizados e as tarefas devem ser processadas. Por exemplo, máquina única, máquinas paralelas, Flow Shop, Job Shop ou Job Shop Flexível.

O campo beta ( $\beta$ ) define as restrições que o problema carrega, sendo limitações ou condições que precisam ser atendidas na solução do problema. Alguns exemplos são, respeitar a capacidade máxima, atender ao prazo de entrega, garantir a precedência entre as operações ou respeitar a capacidade dos estoques entre máquinas (buffers).

O último campo, gama ( $\gamma$ ), representa a função objetivo do problema, ou seja, o objetivo que se deseja alcançar, seja maximizando ou minimizando algum critério, por exemplo: minimizar os custos, minimizar o tempo total de execução (makespan) ou maximizar a utilização dos recursos.

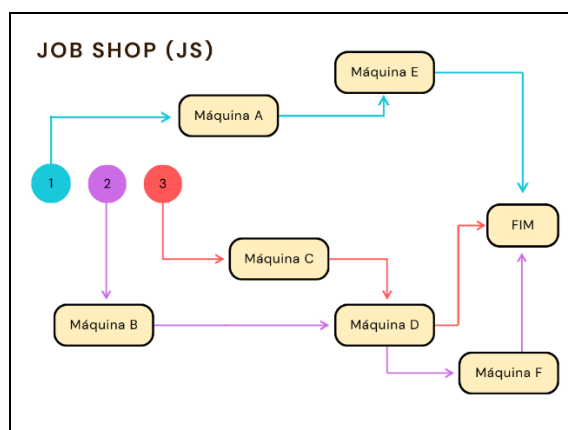
Este trabalho tem como objetivo realizar uma revisão bibliográfica integrativa sobre o problema de Job Shop Flexível (JSF). Serão apresentados os principais conceitos e definições do problema. Em seguida, serão compiladas e analisadas as mais relevantes restrições, funções objetivo e métodos de solução. E, por fim, será realizada uma avaliação das conclusões e sugestões de trabalhos futuros com base nas revisões de Xie et al. (2019) e Dauzere-Peres et al. (2024).

## 2. DESENVOLVIMENTO DO ARGUMENTO

### 2.1. JOB SHOP (JS) E JOB SHOP FLEXÍVEL (JSF)

O Job Shop (JS) envolve um conjunto finito de tarefas, ou em inglês, jobs ( $j$ ), cada uma composta por uma sequência de operações ( $o$ ) a serem realizadas em máquinas predefinidas ( $m$ ) em tempos de processamento especificados, em uma ordem dependente do trabalho (PINEDO, 2016). A Programação de Tarefas em ambiente Job Shop (JS) é um problema clássico de Pesquisa Operacional considerado um problema de otimização combinatória NP-hard desde a década de 1950 (CHAUDHRY, KHAN, 2019).

Figura 1 - Job Shop (JS)

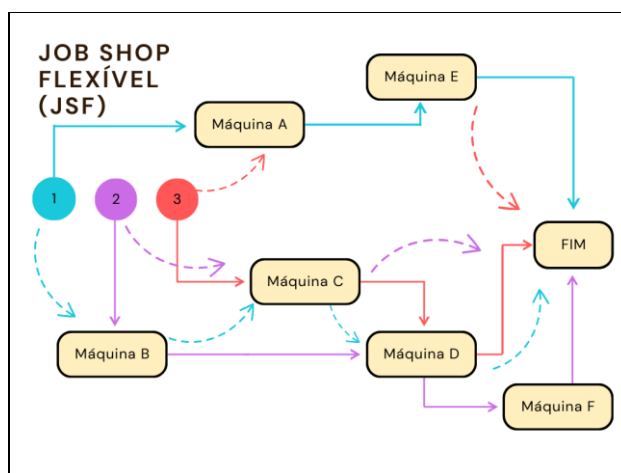


Fonte: De autoria própria.

Podemos observar uma representação do Job Shop (JS) na Figura 1, onde as operações (1, 2 e 3) possuem um único roteiro possível entre as máquinas pré definidas (A, B, C, D, E e F). Por exemplo, o trabalho 1 vai passar pela máquina A, depois pela máquina E, e então chegar ao fim do seu trajeto. O trabalho 2 inicialmente é feito pela máquina B, em seguida máquina D, máquina F e então chegar ao fim.

O Job Shop Flexível (JSF) é uma extensão do JS, no qual é adicionada uma flexibilidade no processo. No tipo mais comum de JSF, chamado de flexibilidade de roteamento, as operações podem ser processadas alternativamente em diversas máquinas, o que aumenta a complexidade do problema. Nesse cenário, além de definir a ordem do processamento das operações nas máquinas, também é necessário definir em qual máquina a operação será processada. Na Figura 2 é apresentada uma representação do JSF de roteamento. As rotas alternativas são representadas por setas pontilhadas.

Figura 2 - Job Shop Flexível (JSF)



Fonte: De autoria própria.

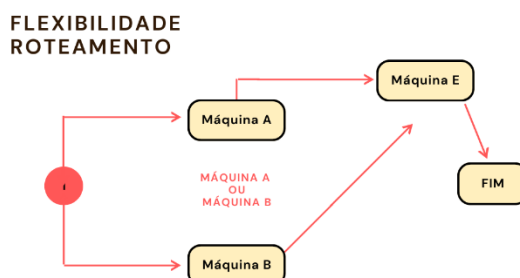
Problemas no ambiente JSF são problemas de complexidade NP-hard e tem ampla aplicação no mundo real. São exemplos de aplicação desse problema em centros cirúrgicos hospitalares, indústrias farmacêuticas, indústrias de semicondutores, indústrias de processamento de peças, indústrias de montagem de produtos, componentes de aeronaves e indústrias de fabricação de moldes, segundo Dauzère-Pérès et al. (2024); Pei et al. (2022) e Pham and Klinkert (2008). A complexidade e a relevância do JSF têm levado a uma vasta quantidade de pesquisas sobre sua modelagem e resolução (DAUZÈRE-PÉRÈS et al.2024).

No Job Shop Flexível (JSF), as operações (1, 2 e 3) podem ter vários caminhos, as operações têm a opção de realização em várias máquinas, como fica claro na Figura 2. O trabalho 1 pode começar tanto na máquina A, quanto na máquina B ou C, pode ir para D ou para o final, dependendo da função objetivo do problema.

Apesar da flexibilidade de roteamento ser mais comumente encontrada nos trabalhos sobre JSF, também existem outros tipos de flexibilidade: processamento e sequenciamento. A seguir vamos explicar cada uma delas.

A flexibilidade de roteiro (ou roteamento) permite diferentes caminhos de produção para um trabalho, dependendo das máquinas disponíveis e de suas capacidades, podendo ser representada pela possibilidade de uma operação ser executada por um subconjunto de máquina (BIRGIN, FERREIRA, RONCONI, 2014). A flexibilidade de roteamento pode ser observada em um pronto socorro de um hospital que possui vários profissionais (máquinas) que podem realizar a assistência de um paciente (tarefa) no processo de atendimento médico (operação). Nesse exemplo, o paciente pode ser atendido por algumas opções de médicos aptos a realizar a operação de atendimento, no lugar de um único médico específico.

Figura 3 - *Flexibilidade de Roteamento*.



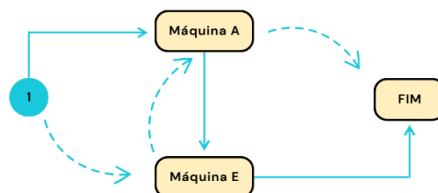
Fonte: De autoria própria.

A Figura 3 representa o exemplo, digamos que o trabalho 1 seja o paciente, a máquina A ou B sejam os médicos qualificados durante o processo de atendimento médico, que é a operação.

A flexibilidade de sequência (ou sequencial) permite que a sequência das operações dentro de uma tarefa possa variar, oferecendo maior adaptabilidade aos processos produtivos. Esse tipo de flexibilidade pode ser encontrado no processo de uma oficina mecânica, que realiza manutenção e reparos em veículos. Nesse exemplo, o carro pode passar pela etapa de calibrar o pneu e depois trocar o óleo, ou inverter a ordem das operações. Ou seja, trocar o óleo e depois calibrar o pneu. Na flexibilidade de sequência a ordem que as operações são processadas não tem importância para o resultado final da tarefa.

Figura 4 - *Flexibilidade Sequencial*.

**FLEXIBILIDADE  
SEQUENCIAL**



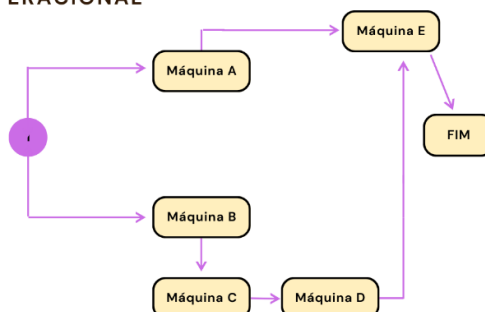
Fonte: De autoria própria.

A Figura 4 representa a flexibilidade sequencial, que pode ser exemplificada como o trabalho 1 sendo o cliente que leva seu carro para a oficina. O carro tem algumas opções de operação, podendo variar entre a máquina A ou E e em seguida, chegar ao final do processo mecânico sem complicações independente da ordem processada.

A flexibilidade de processo permite que a produção de um mesmo resultado seja realizada por operações ou sequências de operações distintas. Isso quer dizer, que a tarefa possui duas ou mais rotas diferentes entre as máquinas. A flexibilidade operacional pode ser exemplificada por uma lanchonete que possui duas opções de pães: produção própria ou pão industrializado. Nos dois cenários o resultado é igual, o pão estará disponível para a produção dos sanduíches, mas o processo para chegar ao resultado é diferente.

Figura 5 - *Flexibilidade Operacional*.

**FLEXIBILIDADE  
OPERACIONAL**



Fonte: De autoria própria.

Na Figura 5, temos o processo de fabricação de uma padaria, tanto de pão industrializado, quanto de produção própria. É possível observar que mesmo com um

processo diferente entre as duas opções de pão, o resultado final é similar, o pão estará disponível para a produção dos sanduíches.

Segundo Billauts, Moukrim e Sanlaville (2008), os problemas podem ser classificados de acordo com o grau de flexibilidade das operações: flexibilidade total, onde cada operação pode ser processada em qualquer máquina, sem especificações, e flexibilidade parcial, onde cada operação está associada a um subconjunto de máquinas.

## **2.2. FUNÇÃO OBJETIVO**

A Função Objetivo (FO) é uma função matemática que representa um indicador de desempenho que deseja-se otimizar, minimizando-a ou maximizando-a (PINEDO, 2016). Em outras palavras, é uma expressão matemática que quantifica a melhor solução de um dado problema.

No contexto do Job Shop Flexível (JSF), a escolha da função objetivo adequada é indispensável para alcançar os resultados desejados. Cada função objetivo define um critério específico para avaliar a performance do sistema, levando em consideração diferentes aspectos da operação, como tempo, custo, utilização de recursos ou flexibilidade.

Podemos classificar as funções objetivo como regulares ou irregulares. Uma função objetivo regular é uma função não decrescente dos tempos de conclusão das tarefas. Ou seja, quando um critério regular é otimizado, é sempre melhor iniciar as tarefas o mais cedo possível. Uma função objetivo irregular, pode ser melhor atrasar o início de uma operação ou inserir períodos de inatividade em uma máquina para garantir um melhor resultado para o problema (DAUZERE-PERES, et al., 2024).

Outro aspecto importante ressaltar são os trade-offs entre funções objetivo, por exemplo, minimizar o tempo de conclusão total pode levar a um aumento na carga de trabalho das máquinas, enquanto minimizar a carga de trabalho das máquinas pode aumentar o tempo total de produção, de acordo com Pinedo (2002). Nesse exemplo, os objetivos são contraditórios, ou seja, ao beneficiar um objetivo o outro será negligenciado. A análise e definição da melhor função objetivo para cada tipo de aplicação é uma tarefa fundamental para atender as necessidades do problema.

Nesta seção serão representadas as principais funções objetivo encontradas no livro do Pinedo (2016) e nas revisões de Chaudhry e Khan (2016), Xie et al. (2019) e Dauzere-Peres et al. (2024).

A função objetivo mais estudada nos trabalhos acadêmicos é o Makespan (Roshanaei, Azab, ElMaraghy, 2013), que indica o instante de término (completion time) da última tarefa ao deixar o sistema, ou seja, o instante de término máximo. Apesar de muito estudado, esse critério não é muito utilizado nos casos reais (CHAUDRY, KHAN, 2016). Variações dessa FO também podem ser utilizadas como o somatório do instante de término ponderado (Sobeyko,



Mönch, 2016), no qual cada tarefa recebe um peso dependendo da sua importância. Instante de término total (Pezzella, Ciaschetti, 2008 e Zhang et al., 2011), que soma o instante de término de todas as tarefas do sistema e o Instante de término médio (Gao, et al., 2015), que é o resultado da divisão do Instante de término total pelo número de tarefas.

Outro tipo de função objetivo bastante explorada na literatura são aquelas associadas a data de entrega, ou em inglês due date. Três tipos de função objetivo podem ser encontradas nesse contexto. O primeiro tipo é o desvio da data de entrega (lateness) (Chen, et al., 2008), que adiciona à função objetivo a variação de tempo entre a data de entrega e o término da tarefa quando a tarefa está atrasada, e subtrai essa variação quando a tarefa é concluída antes do prazo.

O segundo tipo considera apenas o atraso (Baykasoglu, 2002) (tardiness), ou seja, se uma tarefa for concluída após a data de entrega, a função objetivo recebe uma adição correspondente à variação positiva entre o término da tarefa e a data de entrega. Já o terceiro tipo considera apenas o adiantamento (earliness) (Gomes, et al., 2013), de modo que, se a tarefa for concluída antes do prazo, a variação positiva entre a data de entrega e o término da tarefa é adicionada à função objetivo.

As funções objetivo de desvio, atraso e adiantamento podem sofrer variações para se adequar melhor à realidade do problema. Além da soma total desses indicadores para cada tarefa, também pode-se utilizar a soma ponderada, no qual cada tarefa recebe um peso relativo à sua importância. Outra opção é considerar somente o número de tarefas que foram finalizadas fora da data de entrega, desconsiderando a variação entre o término da tarefa e sua data de entrega. Além disso, também pode-se considerar somente o desvio, atraso ou adiantamento máximo, nesse caso, só será considerado na função objetivo a tarefa com maior variação entre seu término e a data de entrega.

Um caso especial é a soma do atraso e adiantamento total das tarefas (Rey, et al., 2015), essa função objetivo pode ser utilizada em um sistema Just In Time (JIT), uma filosofia que tem como objetivo processos mais enxutos e produção no “tempo certo” a fim de reduzir gastos com estoques e evitar desperdícios. Essa função objetivo é uma função não regular, ou seja, em alguns casos a adição de tempos ociosos nas máquinas é benéfica para garantir que a tarefa seja finalizada na data de entrega.

Outra função objetivo apontada na literatura é a minimização do tempo de fluxo das tarefas ou flow time, em inglês (Baykasoglu, 2002). Essa função objetivo representa a diferença entre o instante de término e de início da tarefa, ou seja, o tempo que a tarefa é processada no sistema. Normalmente encontramos duas apresentações desse indicador: máximo tempo de fluxo (maximum flow time) ou tempo de fluxo médio (mean flow time). Esta função objetivo busca aumentar a capacidade produtiva do sistema, uma vez que reduz o



tempo em que as tarefas estão utilizando as máquinas dando tempo para mais tarefas serem processadas.

O indicador maximização de carga de trabalho ou workload (Xia, Wu, 2005) é a segunda função objetivo mais encontrada na literatura de JSF, depois do Makespan (DAUZERE-PERES et al., 2024). Esse critério utiliza o tempo de utilização das máquinas para atingir os objetivos esperados do problema. Normalmente é encontrado em duas versões: Carga de trabalho máxima (maximum workload) ou Carga de trabalho total (total workload). Na primeira, deseja-se minimizar o tempo máximo de trabalho na máquina com a maior carga de trabalho, isso garante um sistema com carga de trabalho mais equilibrada entre as máquinas e evita que algumas máquinas fiquem sobrecarregadas. A segunda versão, é a soma da carga de trabalho de todas as máquinas do sistema. Esses critérios são classificados como não-regulares.

A função objetivo minimização do tempo ocioso (idle time, em inglês) busca reduzir o tempo em que as máquinas ficam sem processar tarefas e é calculada pela diferença entre o tempo em que a máquina está sendo utilizada e a soma dos tempos de processamento das operações realizadas na máquina (Chen, et al., 2012). Essa função objetivo é especialmente interessante para problemas que possuem a restrição de blocking (Mascis, Pacciarelli, 2002). Duas versões são comumente encontradas na literatura: tempo ocioso médio ou tempo ocioso total.

Outra função objetivo comum em problemas de programação de tarefas no ambiente JSF é a minimização do custo total (Mokhtari, Dadgar, 2015). As funções objetivo discutidas até agora podem ser convertidas em termos de custo, transformando a minimização de custos em uma função multiobjetivo, composta por várias funções objetivo. Por exemplo, pode-se definir um custo associado ao tempo ocioso de uma máquina ou ao atraso de uma tarefa. Esse critério também pode ser aplicado a uma máquina específica que esteja associada a altos custos, como uma máquina gargalo.

Atualmente, com o aumento da preocupação das organizações com a sustentabilidade, há um número crescente de trabalhos focados na minimização do consumo de energia (Lei, Zheng, Guo, 2017) e na redução da pegada de carbono (Liu, et al., 2017). Esse movimento está diretamente ligado às mudanças climáticas observadas nas últimas décadas e à percepção da necessidade de transformação nas práticas organizacionais, alinhadas aos Objetivos de Desenvolvimento Sustentável (ODS) estabelecidos pela Organização das Nações Unidas (ONU).

A função objetivo de maximização da robustez (Shen, et al., 2017) está associada com a maior capacidade de resposta a imprevistos e menor tempo de reconfiguração. Em outras



palavras, busca-se que as soluções encontradas no novo problema (após a adição do imprevisto) apresentem a menor variação possível, quando comparada à solução do problema original. Dessa forma, o tempo necessário para a reconfiguração do sistema após o imprevisto seja o menor possível. Alguns tipos de robustez são: schedule, estabilidade, desempenho e qualidade.

### **2.3. RESTRIÇÕES**

Em um problema de programação matemática, as restrições são condições que devem ser atendidas para que o sistema opere de maneira adequada. Ou seja, são exigências que definem as limitações do problema. Ao considerar essas restrições, os planejadores de produção podem encontrar soluções factíveis. Caso uma ou mais restrições não sejam atendidas dizemos que a solução é infactível e não pode ser utilizada como solução para o problema real.

A restrição de Sequenciamento de Tarefas (de Werra, Widmer, 1991) é uma limitação entre as sequências de tarefas em um ambiente JSF, que determina a ordem em que as tarefas serão executadas em cada máquina. Ou seja, determinada tarefa depende da conclusão de sua tarefa antecedente e é uma dependência da sucessora. A restrição de Dependência (Azzouz, Ennigrou, Said, 2017) apresenta basicamente a mesma ideia, porém é apenas dependente da tarefa predecessora para continuar o processo. Ou seja, indicam relações de dependência entre as tarefas, onde uma tarefa não pode começar até que uma ou mais tarefas predecessoras tenham sido concluídas.

A restrição de Recursos (Azzouz, Ennigrou, Said, 2017) é quando existe uma quantidade limitada de recursos disponíveis para a utilização dentro do processo operacional. Ou seja, limita a quantidade de recursos disponíveis para a execução das tarefas. A restrição de Tempo é quando existe um período limitado de tempo disponível para a realização do processo operacional. Ou seja, estabelece limites de tempo para a conclusão das tarefas ou para o ciclo de produção como um todo. A restrição de Capacidade (Chang, Matsuo, Sullivan, 1989) limita o potencial de uma máquina até determinada capacidade. Dessa forma, a quantidade de trabalho que uma máquina pode processar é limitada por uma determinada capacidade.

A restrição de Bloqueio (Sundar, Suganthan, Chua, 2013) (blocking) ocorre quando uma tarefa em execução impede outra tarefa de iniciar, podendo ser por falta de espaço, recursos ou quando as duas máquinas não podem trabalhar juntas. Nesse caso, a operação que termina seu processamento em uma máquina e não pode continuar sua rota, bloqueia a máquina impossibilitando outras operações de serem executadas. Essa situação ocorre em centros cirúrgicos quando não existe vaga na UTI e esse paciente fica



ocupando o centro cirúrgico até que uma vaga seja disponibilizada. Esse fenômeno pode gerar gargalos e atrasos na produção.

A restrição Sem Espera (Li, Huo, 2009) (no wait) visa exatamente não permitir tempo de espera. Ou seja, determinada tarefa deve iniciar imediatamente após a conclusão de sua tarefa predecessora, sem tempo de espera. Portanto, uma operação não pode iniciar o processamento em uma máquina, se ao final do seu tempo de processamento, não tiver espaço disponível na próxima máquina.

A restrição de Buffer Limitado (Mati et al., 2011) é quando existe um local de armazenamento temporário para tarefas em espera dentro do processo produtivo. Normalmente, esse buffer está situado entre máquinas e representa uma restrição física do ambiente produtivo. Um caso especial é quando o Buffer é zero, ou seja, não existe espaço entre as máquinas para armazenar as tarefas. Nesse caso e quando o buffer está com sua capacidade máxima, as restrições de Blocking ou Sem Espera podem ocorrer.

A restrição de Recursos Compartilhados (Chan et al., 2006) permite que recursos possam ser utilizados por várias tarefas ao mesmo tempo. Essa é uma restrição especialmente complexa encontrada em vários ambientes produtivos.

A restrição de Setup (Rossi, Dini, 2007) exige que uma tarefa deve esperar o tempo de Setup, que é o tempo para ajustes da máquina ou da linha de produção, para dar continuidade à produção. Um caso especial é o Setup Dependente de Sequência, no qual a ordem das operações em uma máquina pode variar o tempo de setup. Essa restrição é muito comum em produtos produzidos em batelada que necessitam do processo de limpeza à cada troca de produto. Indústrias de tintas, sucos, refrigerantes e derivados do leite são exemplos.

A restrição de Transporte (Zhang, Manier, Manier, 2012) considera os tempos das movimentações de materiais, peças e produtos durante o processo produtivo. Muitas vezes essa restrição está associada a recursos compartilhados como guas, paletizadoras ou carros de transporte.

PINEDO (2016) aborda as restrições de buffer, blocking e no-wait em diferentes tipos de sistemas de produção e seus impactos no sequenciamento e eficiência do sistema. KHAN e CHAUDHRY (2016) frequentemente agrupam as restrições de blocking e no-wait como casos especiais de restrições de sincronização. **Buffer, blocking e no-wait** são restrições de sincronização interconectadas no contexto de sistemas de produção e sequenciamento de processos.



---

## 2.4. MÉTODOS DE RESOLUÇÃO

Os Métodos de Resolução variam em termos de abordagem, eficiência computacional e capacidade de encontrar soluções ótimas ou próximas da ótima. Alguns métodos são baseados em modelos matemáticos precisos, enquanto outros são heurísticas e metaheurísticas, que buscam soluções viáveis em um tempo computacionalmente razoável (PINEDO, 2016).

A escolha do método de resolução adequado depende da natureza do problema a ser resolvido, incluindo o tamanho do problema, a complexidade das restrições e a disponibilidade de recursos computacionais. Além disso, considerações sobre a qualidade da solução desejada e os prazos também influenciam na escolha do método.

Em sua obra, Chaudhry e Khan (2016) classificam os Métodos de Solução, os métodos foram listados. Também foi levado em conta os métodos de solução citados por Pinedo (2016) e Xie et al. (2019). A seguir definiremos cada um dos métodos.

### 2.4.1. PROGRAMAÇÃO MATEMÁTICA

A Programação Matemática é um método que descreve um problema com funções, equações e inequações matemáticas. Chamamos de Programação Linear (PL) quando as relações entre as variáveis são lineares. Embora nem todos os problemas de JSF possam ser formulados como Problemas de Programação Linear, muitas vezes é possível encontrar uma formulação aproximada que permita a aplicação desse método, ou seja, ajustar o problema mais complexo de forma que possibilite a resolução de forma linear.

A Programação Inteira (PI) estende a programação linear permitindo que as variáveis sejam inteiras ou binárias, tornando o problema mais complexo. Problemas de JSF frequentemente envolvem restrições que podem ser modeladas de forma mais precisa usando variáveis inteiras, como a variável binária:  $x_{ik} = 1$  se a operação  $i$  é alocada à máquina  $k$ , 0 caso contrário, ou seja, a dificuldade de encontrar uma solução viável é mais difícil.

Também existem os modelos de Programação Linear e Inteira Mista (PLIM), que são projetados especificamente para resolver problemas que envolvem tanto variáveis contínuas quanto inteiras, ou seja, é a junção dos dois casos acima. Eles podem ser úteis para problemas de JSF que exigem a modelagem de restrições mistas.

Os modelos matemáticos descritos com Programação Matemática podem ser resolvidos com métodos de otimização exatos como Simplex e Branch and Bound. Existem softwares comerciais confiáveis para resolver esse tipo de problema. Para citar alguns exemplos temos o Gurobi Optimizer e o CPLEX Optimization, da IBM.



#### **2.4.2. HEURÍSTICAS E META-HEURÍSTICAS**

As Heurísticas e Meta-heurísticas são métodos utilizados para resolver problemas de otimização que podem ser difíceis ou impossíveis de resolver de maneira exata em um tempo razoável, segundo Xie et al. (2019), ou seja, é uma forma mais abrangente de resolver os problemas. Embora ambos os métodos não garantam a obtenção da solução ótima, eles são amplamente utilizados para encontrar soluções satisfatórias dentro de prazos de tempo aceitáveis, e podem ajudar em diversos casos dentro do dia a dia. Algumas heurísticas são: Regra de Maior Tempo de Processamento (LPT - Longest Processing Time), Regra de Johnson e Regra da Maior Demanda de Máquinas (MWKR - Most Work Remaining). Já Metaheurísticas, podemos considerar: Algoritmo Genético (AG), Simulated Annealing (SA) e Algoritmo de Colônia de Formigas (ACO).

As Heurísticas Construtivas são métodos que constroem uma solução passo a passo, adicionando elementos até que uma solução completa seja formada. A qualidade da solução obtida depende da ordem e da forma como os elementos são adicionados.

As Meta-Heurísticas são estratégias de alto nível, que guiam outras heurísticas, para explorar o espaço de soluções de maneira mais eficaz, segundo Dauzere-Peres et al. (2024). Elas são usadas para escapar de ótimos locais e explorar uma variedade maior de soluções possíveis.

#### **2.4.3. MÉTODOS DE DECOMPOSIÇÃO E APROXIMAÇÃO**

Os Métodos de Decomposição e Aproximação dividem o problema em subproblemas menores e mais fáceis de resolver. Uma vez que os subproblemas são resolvidos, suas soluções são combinadas para obter uma solução para o problema original, de acordo com Dauzere-Peres et al. (2014). Por exemplo, o Problema de Roteirização que pode ser reduzido em pequenos Problemas do Caixeiro Viajante.

#### **2.4.4. MÉTODOS DE BUSCA LOCAL**

Estes Métodos de Busca Local começam com uma solução inicial, que normalmente é resultado de uma Heurística Construtiva e iterativamente buscam soluções vizinhas que são melhores em termos de uma Função Objetivo específica, ou seja, busca os pontos que estão mais próximos do ótimo. Eles são frequentemente usados como componentes meta-heurísticos, de acordo com Xie et al. (2019).

### **2.5. ANÁLISE DAS REVISÕES DE LITERATURA**

Analizamos as conclusões e sugestões de trabalhos futuros das revisões de literatura de Xie et al.(2019) e Dauzere-Peres et al. (2024) com objetivo de identificar e agrupar os pontos em comum. Nessa seção iremos apontar e analisar quais os caminhos propostos por esses autores para o desenvolvimento de novos trabalhos na área de programação de tarefas

em ambiente JSF. A partir dessa análise, organizamos os achados em cinco temas que serão apresentados a seguir.

### **2.5.1. NOVAS FUNÇÕES OBJETIVO PARA EXPLORAR**

A escolha da função objetivo é uma atividade crítica no desenvolvimento de um solução para um problema de programação de tarefas. Essa escolha impacta diretamente o resultado final do problema e é determinante para levar um projeto ao sucesso ou fracasso.

Para Xie et al. (2019), o nível de personalização nas organizações está se tornando cada vez mais complexo, o que torna os ambientes de produção cada vez mais complicados. Nesse contexto os autores apontam que as funções multiobjetivo deveriam receber mais atenção dos pesquisadores, pois conseguem retratar melhor as necessidades reais das organizações. Os autores destacam que algumas funções objetivo pouco exploradas como robustez (capacidade do sistema em manter seu desempenho satisfatório com incertezas, variabilidades e perturbações no ambiente ou nos parâmetros do problema), grau de satisfação (quanto uma solução atende às expectativas de um determinado problema ou de critérios) e estabilidade do sistema (capacidade de um sistema de retornar ao estado de equilíbrio após problemas) deveriam ser consideradas como critérios para as funções multiobjetivo. Esses critérios também vão ao encontro da realidade das organizações, pois estão associados a melhores respostas a imprevistos, menor tempo de reconfiguração e satisfação dos colaboradores.

Em Dauzere-Peres et al. (2024), os autores apontam que embora o makespan seja o critério mais estudado na literatura, é raramente relevante em ambientes práticos. Por isso deve-se considerar as funções com múltiplos critérios, que são mais comuns na prática. Os autores ainda apontam a importância de considerar critérios não-regulares como carga de trabalho total (total workload) e soma do atraso e adiantamento total das tarefas, comumente utilizado em ambientes Just In Time (JIT).

### **2.5.2. ALÉM DO PLANEJAMENTO: COMO DESENVOLVER MÉTODOS QUE AUXILIEM NO CONTROLE DA PRODUÇÃO**

Tradicionalmente, problemas de programação de tarefas tem objetivo de criar um plano de produção que será executado em algum cenário específico, uma indústria, por exemplo. Porém, durante a execução do plano, eventos dinâmicos como máquinas quebradas ou chegada de novos pedidos são comuns e inevitáveis. Nesse contexto, novos trabalhos vêm sendo desenvolvidos na área de programação de tarefas em ambiente JSF para auxiliar também no controle da produção.

Em Xie et al. (2019), os autores apontam a mudança frequente das necessidades dos clientes em um ambiente de produção real. Para esses problemas, o circuito aberto, onde as decisões são tomadas com base em informações pré-estabelecidas e não são ajustadas com



base nos resultados ou no feedback obtido após a implementação das decisões, pode levar à deterioração da utilização dos recursos. O agendamento em circuito fechado pode ser realizado dinamicamente, além de ajustar o esquema de programação de acordo com a produção em tempo real. Integrando big data, Cyber-Physical Systems (CPS), gêmeo digital e outras tecnologias, a programação em malha fechada pode monitorar a situação de produção da oficina em tempo real.

Dauzere-Peres et al. (2024) apontam que pesquisas significativas ainda são necessárias para lidar com eventos aleatórios e dinâmicos em problemas de programação de tarefas. O tempo de processamento de uma operação pode ser mais incerto em uma máquina do que em outra, e é possível atribuir uma operação para outra máquina se a mesma quebrar. Além disso, muitas configurações práticas são dinâmicas, no sentido de que um cronograma otimizado é calculado em um horizonte contínuo, ou seja, é regularmente recalculado para integrar a chegada de novos trabalhos ou a mudança de status das máquinas.

Para Xie et al. (2019), novas abordagens, como online, dynamic ou real time scheduling, podem ser empregadas para prever a ocorrência de eventos dinâmicos. Já para Dauzere-Peres et al. (2024) ainda são necessárias pesquisas significativas para lidar com eventos aleatórios e dinâmicos na programação de tarefas.

Outra abordagem que tem ganhado notoriedade são aquelas que propõem integração com outros sistemas, por exemplo Real Time Logistic System. Essa proposta tornou-se uma nova tendência de pesquisa e são utilizados para otimizar a seleção de rotas, encurtar o ciclo de produção, reduzir o retrabalho, avaliar e melhorar a flexibilidade do sistema de fabricação (XIE et al., 2019).

### **2.5.3. NOVAS APLICAÇÕES: ADEQUANDO À REALIDADE DAS EMPRESAS**

O problema de programação de tarefas foi idealizado e é utilizado até hoje para auxiliar na tomada de decisões em um ambiente de manufatura. Uma evidência desse fato é que os recursos são até hoje chamados de máquinas, ainda que possam ser pessoas ou ferramentas. Nesse contexto, os autores propõem abordagens para adequar o problema de programação de tarefas à realidade das empresas.

Além das novas funções objetivas abordadas na seção 3.1, também podemos citar novas restrições que buscam aproximar os modelos ao dia-a-dia das empresas. Para Dauzere-Peres et al. (2024), incluir a degradação das máquinas e múltiplos recursos é uma necessidade das organizações que ainda não foi estudado com profundidade na literatura.

Xie et al. (2019) aponta duas novas tendências da manufatura que têm ganhado espaço e devem ser incorporadas aos problemas de programação de tarefas no ambiente JSF. A manufatura distribuída é uma forma descentralizada de produção na qual empresas compartilham instalações dispersas geograficamente. Nesse novo tipo de manufatura, se faz



necessário uma coordenação entre empresas utilizando tecnologias da informação. Esse modelo promete uma redução de custos através da alocação racional, combinada e compartilhamento de recursos. Outra tendência são as Empresas Virtuais que são alianças temporárias entre organizações, nas quais são compartilhadas habilidades, competências e recursos com objetivo de desenvolver uma resposta rápida e eficaz para novas oportunidades de negócios e estabelecer vantagens competitivas. Por fim, os autores apontam outras áreas em que, devido às semelhanças dos problemas, podem ser interpretados ou modelados como um problema de programação de tarefas em ambiente JSF. São exemplos dessas áreas transporte, gestão, aeroespacial, saúde e rede de comunicações.

#### **2.5.4. NOVAS TECNOLOGIAS E ABORDAGENS PARA O AVANÇO DA PESQUISA**

Nesta seção iremos apresentar algumas novas tecnologias e abordagens para aprimorar as pesquisas e contribuir para o desenvolvimento da área.

Na revisão de Xie et al. (2019), os autores apontam a utilização da Data Analysis e da Inteligência Artificial para extrair informações da produção que podem auxiliar no desenvolvimento de métodos para problemas difíceis de modelar matematicamente. Um exemplo seria modelos de consumo de energia e emissão de carbono, que são mais eficientes quando analisados a partir dos dados em conjunto com um modelo teórico. Essas novas tecnologias também podem auxiliar na previsão de eventos dinâmicos como a indisponibilidade de uma máquinas e a chegada de novos pedidos

Outras tecnologias que podem auxiliar na aplicação e teste de novos modelos e métodos no FJS com utilização de dados reais são o Cyber-Physical System (CPS) e o Digital Twin. O CPS é um sistema que integra informações avançadas e tecnologia de controle automático, incluindo detecção, computação, comunicação e controle, para construir um sistema complexo que combina o ambiente de produção real com dados de produção. O Digital Twin ou Gêmeo Digital constrói uma entidade e um sistema virtual que pode representar a oficina de produção real (informação) em espaço virtual.

Em Dauzère-Pérès et al. (2024), os autores apontam que a maioria dos trabalhos com a FJSP tem como foco o desenvolvimento de abordagens adaptadas a um problema específico. Nesse contexto, os autores invocam os leitores a desenvolver trabalhos que explorem a estratégia “de cima para baixo”, que refere-se a uma abordagem metodológica em que se começa com uma visão geral do problema antes de se aprofundar nos detalhes específicos. Isso significa criar modelos ou métodos que sejam amplamente aplicáveis a diferentes tipos de problemas dentro do FJSP, sem a necessidade de adaptações significativas para cada caso particular. Em vez de desenvolver soluções sob medida para problemas específicos, a ideia é construir uma base sólida e genérica que possa ser

facilmente estendida ou ajustada para atender a diferentes necessidades. Em vez de focar diretamente em problemas ou casos específicos, como é comum em muitas abordagens tradicionais, essa estratégia propõe o desenvolvimento de soluções gerais e estruturadas que possam ser aplicadas a uma ampla gama de situações. Isso beneficiará o campo da programação de tarefas, ao apresentar abordagens prontas para uso em extensões genéricas do FJSP.

Os autores apontam que entender conceitualmente o que torna uma abordagem de resolução eficiente seria um ganho para a área. Essa análise poderia ser teórica, analisando as estruturas de vizinhança e os movimentos mais eficientes, e empírica, criando novas instâncias para compreender as abordagens de resolução.

### **2.5.5. SUGESTÕES DE COMO MELHORAR OS MÉTODOS DE RESOLUÇÃO**

Xie et al. (2019) destaca que mais pesquisas podem focar na solução do FJSP por meio de novos Algoritmos de Otimização Inteligentes. Eles explicam que devido a complexidade dos ambientes produtivos, muitas vezes algumas características não podem ser modeladas usando os métodos convencionais. Esses novos algoritmos podem apresentar uma programação de tarefas melhor e em menor tempo.

Em Dautère-Pérès et al. (2024), os autores explicam que, devido à flexibilidade das máquinas, as vizinhanças a explorar normalmente são muito grandes, e sugerem o desenvolvimento de eficientes algoritmos de busca local que incluam procedimentos de viabilidade e avaliação de movimentos para evitar o custo computacional de verificar todos os movimentos possíveis. Além disso, os autores apontam algumas restrições mais difíceis de resolver, pois levam a muitas soluções ineficazes. Exemplos dessas restrições são restrições de blocking e atrasos máximos.

## **3. CONSIDERAÇÕES FINAIS**

Com base na pesquisa sobre programação de tarefas, especialmente no contexto do Job Shop Flexível (JSF), podemos concluir que é um problema que desempenha um papel essencial em indústrias de fabricação, eficácia de processos operacionais, processos logísticos e áreas rotineiras.

A programação de tarefas atribui tarefas a recursos limitados ao longo do tempo e visa a maximizar ou minimizar uma função objetivo específica, como tempo de execução ou custo. O JSF expande o tradicional Job Shop ao introduzir flexibilidade nas operações (roteamento, sequencial ou operacional), permitindo diferentes sequências, operações alternativas em várias máquinas e rotas de produção, o que aumenta significativamente a complexidade do problema.

As revisões de literatura destacam a importância de métodos de resolução como modelos matemáticos, heurísticas e meta-heurísticas para encontrar soluções eficientes. A



escolha do método depende da natureza do problema e dos recursos computacionais disponíveis, além de considerações sobre a qualidade da solução desejada e os prazos envolvidos.

Sugestões de trabalhos futuros incluem buscar mais trabalhos para completar a revisão, explorar novas abordagens que combinam diferentes técnicas de otimização, além de fazer uma análise das conclusões da revisão de 2016.

#### **4. REFERÊNCIAS**

AZZOUZ, A.; ENNIGROU, M.; SAID, L. B. A self-adaptive hybrid algorithm for solving flexible job-shop problem with sequence dependent setup time. *Procedia Computer Science*, v. 112, p. 457-466, 2017.

BAYKASOGLU, A. Linguistic-based meta-heuristic optimization model for flexible job shop scheduling. *International Journal of Production Research*, v. 40, n. 17, p. 4523-4543, 2002.

BIRGIN, E. G.; FERREIRA, J. E.; RONCONI, D. P. List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility. *European Journal of Operational Research*, Amsterdam, v. 237, n. 3, p. 791-801, 2014.

CHAN, F. T. S.; WONG, T. C.; CHAN, L. Y. Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, v. 44, p. 2071-2089, 2006.

CHAUDHRY, I. A.; KHAN, A. A. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, July 2015.

CHEN, J.; CHEN, K.; WU, J.; et al. A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *International Journal of Advanced Manufacturing Technology*, v. 39, n. 3-4, p. 344-354, 2008.

CHEN, J. C.; WU, C.-C.; CHEN, C.-W.; et al. Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm. *Expert Systems with Applications*, v. 39, n. 11, p. 10016-10021, 2012.

DAUZÈRE-PÉRÈS, S.; DING J.; SHEN L.; TAMSSAOUET, K. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, v. 314, n. 2. Disponível em: <https://www.sciencedirect.com/science/article/pii/S037722172300382X>.

DE WERRA, D.; WIDMER, M. Loading problems with tool management in flexible manufacturing systems: a few integer programming models. *International Journal of Flexible Manufacturing Systems*, v. 3, n. 1, p. 71-82, 1991.

GAO, K. Z.; SUGANTHAN, P. N.; TASGETIREN, M. F.; et al. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. *Computers & Industrial Engineering*, v. 90, p. 107-117, 2015.

GOMES, M. C.; BARBOSA-PÓVOA, A. P.; NOVAIS, A. Q. Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach. *International Journal of Production Research*, v. 51, n. 17, p. 5120-5141, 2013.

LEI, D.; ZHENG, Y.; GUO, X. A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *International Journal of Production Research*, v. 55, n. 11, p. 3126-3140, 2017.



LI, L.; HUO, J.-Z. Multi-objective flexible job-shop scheduling problem in steel tubes production. *Systems Engineering*, v. 29, p. 117-126, 2009.

MATI, Y.; LAHLOU, C.; DAUZERE-PERES, S. Modelling and solving a practical flexible job-shop scheduling problem with blocking constraints. *International Journal of Production Research*, v. 49, p. 2169-2182, 2011.

MOKHTARI, H.; DADGAR, M. A flexible job shop scheduling problem with controllable processing times to optimize total cost of delay and processing. *International Journal of Supply and Operations Management*, v. 2, n. 3, p. 871-887, 2015.

PEZZELLA, G. M.; CIASCETTI, G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, v. 35, n. 10, p. 3202-3212, 2008.

PINEDO, M. L. *Scheduling Theory, Algorithms, and Systems*. 5. ed. New York: Springer, 2016.

ROSHANAEI, V.; AZAB, A.; ELMARAGHY, H. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. *International Journal of Production Research*, v. 51, n. 20, p. 6247-6274, 2013.

ROSSI, A.; DINI, G. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robotics and Computer-Integrated Manufacturing*, v. 23, p. 503-516, 2007.

SCIARINI, M.; MELLO, J. C. C. B.; GUERRINI, F. M. Regras de prioridade eficientes que exploram características do Job Shop Flexível para a minimização do atraso total. *Production*, São Carlos, v. 30, e20200004, 2020. Disponível em: <https://www.scielo.br/j/prod/a/RN9xJX5QSSHVnc5FZ5ywZRf/>.

SHEN, X.-N.; HAN, Y.; FU, J.-Z. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing*, v. 21, n. 21, p. 6531-6554, 2017.

SOBEYKO, O.; MÖNCH, L. Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers & Operations Research*, v. 68, p. 97-109, 2016.

XIA, W.; WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, v. 48, n. 2, p. 409-425, 2005.

XIE, J.; GAO, L.; PENG, K.; LI, X.; LI, H. Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, [s.l.], The Institution of Engineering and Technology, 2019.

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

ZHANG, Q.; MANIER, H.; MANIER, M. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, v. 39, p. 1713-1723, 2012.