Wagner Amorim da Silva Altoé

META-HEURÍSTICAS SIMULATED ANNEALING E CLUSTERING SEARCH APLICADA AO PROBLEMA FLEXIVEL JOB SHOP SCHEDULING COM RESTRIÇÕES DE TRABALHADORES E COM TEMPOS DE SETUP ANTECIPADOS

Vitória, ES

2021

Wagner Amorim da Silva Altoé

META-HEURÍSTICAS SIMULATED ANNEALING E CLUSTERING SEARCH APLICADA AO PROBLEMA FLEXIVEL JOB SHOP SCHEDULING COM RESTRIÇÕES DE TRABALHADORES E COM TEMPOS DE SETUP ANTECIPADOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. André Renato Sales Amaral

Vitória, ES

2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Altoé, Wagner Amorim da Silva, 1991-
A469m META-HEURÍSTICAS SIMULATED ANNEALING E CLUSTERING SEARCH APLICADA AO PROBLEMA FLEXIVEL JOB SHOP SCHEDULING COM RESTRIÇÕES DE TRABALHADORES E COM TEMPOS DE SETUP ANTECIPADOS / Wagner Amorim da Silva Altoé. - 2021.
62 f. : il.

Orientador: André Renato Sales Amaral.
Dissertação (Doutorado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Otimização combinatória. I. Amaral, André Renato Sales. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Wagner Amorim da Silva Altoé

META-HEURÍSTICAS SIMULATED ANNEALING E CLUSTERING SEARCH APLICADA AO PROBLEMA FLEXIVEL JOB SHOP SCHEDULING COM RESTRIÇÕES DE TRABALHADORES E COM TEMPOS DE SETUP ANTECIPADOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Prof. Dr. André Renato Sales Amaral
Orientador

Prof. Dr. Geraldo Regis Mauri
Universidade Federal do Espírito Santo
(UFES)

Prof. Dr. Luciano Lessa Lorenzoni
Instituto Federal do Espírito Santo (IFES)

Vitória, ES
2021

Agradecimentos

Agradeço primeiramente a Deus, pois tudo que tenho na vida foi conquistado pela tua bênção.

À minha família, pelo incentivo e apoio para continuar estudando todos esses anos, que não mediram esforços nenhum para me ajudar em qualquer problema.

À minha esposa Larissa Brito, que esteve sempre me apoiando em todos esses anos.

A todos os amigos que eu conheci nesta universidade, em especial, Denis, Josimar, Rodrigo, João Vitor, Felipe Secato, João Paulo, Caio, Hans, Robim, Diego, Joabe, André, Kauan, Keiffer, Paulo Victor, DanielEQ, Caique, Renata, Ligia, Renan, Sergio e muitos outros amigos.

Ao meu orientador, professor Dr. André Renato Sales Amaral, pela disponibilidade em ajudar, pela paciência e seus conhecimentos repassados durante todo o desenvolvimento deste trabalho.

A todos os professores da UFES de Alegre/Vitória, que mesmo passando dificuldades, não se abstiveram de sempre buscar melhorar o conteúdo, motivando os alunos com conhecimentos, que são fundamentais na suas vidas.

À Universidade Federal do Espírito Santo e todos os seus funcionários, principalmente os professores, que puderam transmitir seus conhecimentos de forma empolgante, facilitando ainda mais o aprendizado.

À CAPES pela bolsa de mestrado indispensável para realização deste trabalho.

A todos, muito obrigado.

"Peça a Deus que abençoe os seus planos e eles darão certo."
(Provérbios 16:3)

Resumo

O problema conhecido como Flexivel Job Shop Scheduling com restrições de trabalhadores e com tempos de setup antecipados (WSFJSP-SDST), é uma extensão do problema *Job Shop Schedulling* (JSP). Neste ambiente de produção, são utilizadas máquinas que são operadas por trabalhadores, para assim, processarem um conjunto de *jobs*. Um *job* é caracterizado por possuir uma ordem fixa de operações, onde cada operação somente pode ser processada por trabalhadores que possuem a habilidade para executá-las utilizando uma máquina adequada. Cada trabalhador só pode executar no máximo uma operação por vez, bem como, cada máquina só pode ser operada por um trabalhador por vez, respeitando a restrição que, quando uma operação for iniciada, não poderá ser interrompida antes de sua conclusão. Além disso, considera-se que para uma operação ser executada em uma máquina, necessita-se de um tempo para preparar a máquina a ser utilizada. Neste trabalho estão descritas as meta-heurísticas Simulated Annealing (SA) e Clustering Search (CS) para solucionar o WSFJSP-SDST. As meta-heurísticas foram testadas com instâncias de uma empresa real retirada da literatura, assim como, com instâncias geradas por este estudo. Experimentos computacionais evidenciam que os algoritmos propostos possibilitaram a geração de soluções de maior qualidade com um custo computacional reduzido.

Palavras-chaves: WSFJSP-SDST, SA, CS, makespan, atraso total.

Abstract

The problem known as Worker Constrained Flexible Job Shop Scheduling Problem With sequence-dependent setup times (WSFJSP-SDST), is an extension of the problem Job Shop Schedulling (JSP). In this production environment, machines are operated by workers to process a set of jobs. A job is characterized by having a fixed order of operations, where each operation can only be processed by workers who have the ability to perform them using a suitable machine. Each worker can only execute a maximum of one operation at a time, as well as, each machine can only be operated by one worker at a time, respecting the restriction that, when an operation is started, it cannot be interrupted before its completion. In addition, it is considered that for an operation to be performed on a machine, time is needed to prepare the machine to be used. This work describes the Simulated Annealing (SA) and Clustering Search (CS) metaheuristics to solve the WSFJSP-SDST. This work describes the Simulated Annealing (SA) and Clustering Search (CS) metaheuristics to solve the WSFJSP-SDST. Meta-heuristics were tested with instances of a real company taken from the literature, as well as with instances generated by this study. Computational experiments show that the proposed algorithms enabled the generation of higher quality solutions with reduced computational cost.

Keywords: WSFJSP-SDST, SA, CS, makespan, total tardiness.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CS	*Clustering Search*
SA	*Simulated Annealing*
JSP	*Job Shop Scheduling*
FJSP	*Flexível Job Shop Scheduling*
DRCFJSP	*Dual Resource Constrained Flexible Job Shop Scheduling Problem*
WSFJSP-SDST	*Worker Constrained Flexible Job Shop Scheduling Problem With sequence-dependent setup times*

Sumário

1 INTRODUÇÃO

A preocupação em programar as atividades de forma eficiente pode ser observada desde a época das construções das pirâmides (HYATT; WEAVER, 2006). Porém, somente no fim da segunda guerra mundial, as organizações militares começaram a discutir mais a fundo sobre a necessidade do planejamento de suas atividades (ECCLES, 1954).

E com o avanço da tecnologia, fatores como o aumento da competitividade pressionaram as indústrias a buscarem estratégias que possibilitaram uma produção mais eficiente.

Com isso, nos dias atuais, as indústrias devem empenhar-se para cumprir os prazos estipulados, pois clientes insatisfeitos poderão procurar concorrentes que possuem uma produção mais eficiente, ocasionando assim, uma diminuição da necessidade de produção. Dessa forma, as empresas devem programar suas atividades de tal forma que os recursos sejam utilizados de maneira eficiente (SINGH; MAHAPATRA, 2016).

O problema de sequenciamento de produção está presente em várias indústrias, principalmente em indústrias de manufaturas. Neste tipo de contexto, encontra-se o problema denominado *job shop scheduling* (JSP) (YU; LEE, 2018).

No JSP há um conjunto de *jobs* (produto a ser produzido), cada um formado por uma sequência de operações (etapas para a produção do produto). Cada operação deve ser processada por uma máquina específica, obedecendo às restrições de precedência. Além disso, cada máquina é capaz de processar uma única operação por vez, sendo que, quando iniciada, não pode ser interrompida (preempção).

O objetivo deste problema, denominado *Job Shop Scheduling* (JSP) se resume a designar a ordem das operações de cada *job* nas máquinas previamente definidas, visando otimizar algum critério. Dentre esses critérios, destacam-se, o tempo de término da última operação completada (makespan), a minimização do tempo do fluxo total (somatório dos tempos de término das tarefas), minimização do atraso máximo (minimização do tempo da tarefa que tem o maior tempo de atraso) e minimização do atraso total, que é o somatório dos atrasos das tarefas (RODAMMER; WHITE, 1988).

Por se tratar de um problema antigo, e tendo em consideração que o avanço da tecnologia proporcionou vários tipos distintos de máquinas, assim como formas diferentes de produções, surgiu a necessidade do estudo de várias adaptações deste modelo.

Uma destas adaptações, foi a necessidade de tornar o modelo mais flexível. Desta forma, a indústria poderia ter mais de uma máquina que desempenha uma mesma função, ou mesmo a terceirização deste processo, possibilitando que uma operação possa ser

processada por máquinas distintas. Para deixar o modelo mais próximo da realidade de certas indústrias, outra adaptação bastante estudada foi a necessidade de considerar o tempo de preparação na transição de operações de uma mesma máquina. Esta preparação pode ser justificada como um tempo para fazer uma limpeza na máquina ou mesmo buscar equipamentos que serão necessários para processar a operação.

Também deve-se considerar que os trabalhadores possuem habilidades distintas, ou seja, cada um pode operar um conjunto de máquinas distintas, podendo realizar uma mesma tarefa com tempos diferentes. Além disso, os trabalhadores podem ser considerados como recurso escasso, ou seja, há um limite de trabalhadores que operam as máquinas na indústria.

Segundo KACEM, HAMMADI e BORNE (2002), o *Flexible Job Shop Problem* (FJSP) é um dos problemas de escalonamento NP-Completo mais difíceis de serem resolvidos. E essa complexidade aumenta ao considerar as retrições de trabalhadores e setup nas máquinas (MENG et al., 2019).

Para resolver esse tipo de problema computacionalmente, primeiramente é preciso abstrair a realidade do problema para um conjunto de equações que o representam de forma qualitativa. Para isso, é preciso definir quais hipóteses serão utilizadas no modelo para representar o sistema real. Esse processo de transformar um problema da vida real em um modelo matemático é definido como Programação Matemática (SIERKSMA, 2001). Neste trabalho foi utilizado o modelo matemático proposto por KRESS, MÜLLER e NOSSACK (2019).

Além de definir o modelo, deve-se decidir qual método será utilizado para resolvê-lo. Problemas desta classe já foram solucionados através de métodos exatos e heurísticos. A categoria de métodos exatos consegue garantir que a solução é ótima, porém tendem a ser ineficazes em instâncias maiores, por vezes não encontrando sequer uma solução que seja viável para o problema em tempo razoável(CARVALHO et al., 2001).

1.1 Objetivo

O objetivo geral desta dissertação é desenvolver as meta-heurísticas Clustering Search (CS) e Simulated Annealing (SA) para solucionar o problema de escalonamento flexível de tarefas com restrição envolvendo trabalhadores e com tempos de setups dependentes da sequência (*worker constrained flexible job shop scheduling with sequence-dependent setup times* - WSFJSP-SDST). Como objetivos específicos, tem-se:

- Implementar uma meta-heurística Simulated Annealing (SA) para o WSFJSP-SDST;
- Implementar uma meta-heurística CS utilizando a meta-heurística Simulated Annealing (SA) como geradora de soluções para o WSFJSP-SDST;

- Comparar os resultados obtidos com os resultados fornecidor por KRESS, MÜLLER e NOSSACK (2019).
- Desenvolver novas instâncias para o WSFJSP-SDST;
- Comparar os resultados das meta-heurísticas CS e SA.

1.2 Justificativa e Contribuições

Por se tratar de um problema muito importante na sobrevivência das indústrias, e levando em consideração que esse tema é recente e complexo, não existem muitos estudos resolvendo o WFJSP com tempos de setups dependentes da sequência. Esse fato justifica a busca de novas alternativas eficientes para a resolução deste problema.

Não foram encontradas propostas na literatura que resolvessem este problema utilizando meta-heurísticas. Este fato justifica a escolha em resolver o problema utilizando este método.

A escolha das meta-heurísticas SA e CS para solucionar o problema WSFJSP-SDST, deve-se ao fato de as mesmas terem apresentado bons resultados para solucionar o FJSP (BISSOLI et al., 2018; ALTOÉ et al., 2018). Além disso, não foram encontradas publicações utilizando a SA ou a CS para resolução do WSFJSP-SDST.

Considerando que não foram encontradas diferenças significativas nos resultados da SA e CS nas instâncias propostas por KRESS, MÜLLER e NOSSACK (2019), tornou-se necessária a geração de 10 novas instâncias para o problema.

1.3 Organização do Trabalho

Esta dissertação está organizada em seis capítulos. Os tópicos a serem abordados em cada capítulo estão descritos na sequência.

O Capítulo 1 correspondente ao presente capítulo, apresentou-se a introdução ao problema WSFJSP-SDST, seguido dos objetivos e justificativa do trabalho. O Capítulo 2 aborda o problema do escalonamento de tarefas do tipo *Job Shop* e suas extensões. Posteriormente, são apresentada as definições, propriedades e formulação matemática do Flexivel *Job Shop Scheduling* com restrições de trabalhadores e com tempos de setup antecipados (WSFJSP-SDST). No Capítulo 3 aborda-se uma breve revisão bibliográfica do problema. Na sequência, descreve-se, no Capítulo 4, as meta-heurísticas SA e CS para resolver o WSFJSP-SDST. No Capítulo 5 são apresentados a calibração dos parâmetros e os resultados das meta-heurísticas implementadas neste trabalho, seguidos das novas instâncias geradas. Por fim, as conclusões e trabalhos futuros estão dispostos no capítulo 6, seguidas das referências utilizadas neste trabalho.

2 PROBLEMAS DE ESCALONAMENTO DE TAREFAS

Este capítulo aborda o problema que é o objetivo do estudo nesta dissertação: Flexivel *Job Shop Scheduling* com restrições de trabalhadores e com tempos de *setup* antecipados (WSFJSP-SDST). O WSFJSP-SDST pertence a categoria de problemas de programação de tarefas. Suas restrições levam em consideração as extensões do problema clássico *Job Shop*, Flexível *Job Shop*, *Job Shop* com tempos de *setup* e *Job Shop* considerando trabalhadores como restrições.

Primeiramente, são apresentado alguns conceitos que definem cada extensão. No seção 2.1 são alguns conceitos do problema *job shop* clássico, no seção 2.2

2.1 Job Shop Scheduling

No ambiente de produção dos problemas de escalonamento de tarefas, tem-se um conjunto de n tarefas que serão realizadas compartilhando recursos em comum. Em uma indústria de manufatura, por exemplo, essas tarefas são chamadas de *jobs*, enquanto os recursos, podem ser estações de trabalho, geralmente representadas por máquinas. Pode-se exemplificar um *job* como a produção de uma camisa, na qual, sua produção será realizada em três etapas (operações): (1) Corte do tecido; (2) Costura do tecido; e (3) Plotagem onde cada operação, é realizada por um recurso específico como por exemplo máquinas.

Figura 1 – Exemplo de um job.

Na Figura 1, tem-se um exemplo de um *job* (J_1), representado por uma produção de uma camisa, onde sua primeira operação $O_{1,1}$ é a execução do corte em um tecido. $O_{1,2}$ como sua segunda operação, constituída pela costura do tecido. $O_{1,3}$ como terceira e última operação, representada pela plotagem da estampa na camisa.

O propósito deste problema consiste em determinar o sequenciamento das tarefas (*jobs*) em cada máquina, de forma que otimize algum objetivo.

As potenciais restrições do problema clássico podem ser divididas em dois tipos. Em restrições de precedência, estabelecendo que uma operação não pode ser executada se sua antecessora ainda não tiver sido executada. Exemplo na Figura 1, onde a operação Costura do tecido não pode ser realizada antes de ter sido finalizada a operação Corte do tecido, assim como não pode processar a operação Plotagem antes de ter efetuado a operação Costura do tecido. E em restrições de preempção onde, cada máquina é capaz de processar somente uma única operação por vez, sendo que, quando iniciada, não pode ser interrompida (JAIN; MEERAN, 1999; ARENALES; ARMENTANO et al., 2006).

No JSP cada operação possui somente uma opção de máquina para processá-la. Na Tabela 1, temos o exemplo de uma instância do *JSP* onde, na primeira coluna, são definidos os jobs; na segunda coluna, estão dispostas as operações dos respectivos jobs e, nas demais colunas, temos a atribuição do tempo de processamento das operações nas máquinas. As operações que não podem ser executadas por uma máquina específica, são representadas pelo símbolo (-).

Tabela 1 – Exemplo de uma instância do JSP.

Job	Operação	M1	M2	M3
J1	$O_{1,1}$	-	-	3
	$O_{1,2}$	2	-	-
J2	$O_{2,1}$	-	1	-
	$O_{2,2}$	1	-	-

2.2 Flexible Job Shop Scheduling

O avanço da tecnologia proporcionou máquinas do tipo multifuncionais. Essas máquinas podem executar diferentes operações de manufatura, até mesmo realizar tarefas em comum, permitindo assim, que uma operação possa ser executada por um conjunto de máquinas (KRESS; MÜLLER; NOSSACK, 2019). Com isso, a aplicação do JSP é limitada devido à restrição de que uma operação só pode ser processada por uma única máquina específica. Diante a esta necessidade, surgiu o *Job Shop Scheduling* Flexível (FJSP), criando a possibilidade de máquinas distintas realizarem a mesma função, tornando assim, o modelo mais próximo da realidade (HO; TAY, 2004).

Na Tabela 2 tem-se um exemplo de uma instância do FJSP, diferentemente do JSP na Tabela 1, uma operação, por exemplo, $O_{1,1}$ pode ser processado por mais de uma máquina.

Tabela 2 – Exemplo de uma instância do FJSP.

Job	Operação	M1	M2	M3
J1	$O_{1,1}$	-	1	3
	$O_{1,2}$	2	-	1
J2	$O_{2,1}$	-	2	-
	$O_{2,2}$	1	-	1

2.3 Flexible Job Shop Scheduling com Tempos de Setup

Em certas indústrias, as máquinas devem ser preparadas antes de processar uma operação. Como exemplos de preparação relaciona-se a necessidade de limpeza do equipamento, posicionamento do material de trabalho, obtenção ou devolução dos equipamentos necessários para a realização da operação. Este tempo é denominado como *setup* (SHARMA; JAIN, 2015), que pode ser classificado de duas formas:

1. Dependentes da sequência (*setups* explícitos): quando o tempo de *setup* leva em consideração a troca de operações na máquina, ou seja, a operação atual e a operação processada anteriormente na máquina específica. Na Figura 2 é apresentado um exemplo de *setup* dependente da sequência, onde $O_{A,B}$ caracteriza a "operação B" do "job A", as setas em azul contidas no "setup 2" simbolizam a inferência das operações $O_{1,1}$ e $O_{1,2}$ no seu tempo de processamento. (ZHU; WILHELM, 2006; EREN, 2007).

Figura 2 – Tempo de setup dependente da sequência.

2. Independentes da sequência: o tempo de *setup* depende somente da tarefa a ser processada, independentemente da tarefa anterior. Na Figura 3, tem-se o tempo do "setup 2" que considera somente a operação $O_{1,2}$ no seu tempo de processamento. (ZHU; WILHELM, 2006; EREN, 2007).

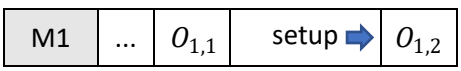

Figura 3 – Tempo de *setup* independente da sequência.

Além disso, o *setup* pode ser definido como antecipados e não antecipados:

Setup antecipado: não é necessário que a operação esteja na máquina para processá-la, podendo ser executado enquanto a operação precedente do mesmo job ainda esteja sendo executada em outra máquina. Na Figura 4 tem-se um exemplo de *setup* antecipado.

Considera-se $S_{01,12}$ o tempo de *setup* da operação O_{12} antecedido pela operação O_{01} (FUCHIGAMI et al., 2017).

M1	$S_{01,11}$	$O_{1,1}$	
M2		$S_{01,12}$	$O_{1,2}$

Figura 4 – Exemplo do setup antecipado.

Setup não antecipado: só podem ser iniciados no instante em que a operação for liberada para o processamento na máquina, neste caso, é exigido que a operação esteja presente para iniciar a configuração. Percebe-se na Figura 5 o tempo de *setup* da operação $O_{1,2}$ antecedido pela operação O_{01}, representada por $S_{01,12}$ iniciando depois do término de sua operação antecessora, $O_{1,1}$.

M1	$S_{01,11}$	$O_{1,1}$		
M2			$S_{01,12}$	$O_{1,2}$

Figura 5 – Exemplo do setup não antecipado.

2.4 Flexible Job Shop Scheduling com Restrições de Trabalhadores

Grande parte dos sistemas de manufatura precisam de trabalhadores para manusearem as máquinas, e esses recursos tendem a ser escassos. Com isso, além da restrição das máquinas, também é necessário levar em consideração a disponibilidade do trabalhador e a habilidade necessária para operar a máquina. Na literatura, a consideração de trabalhadores em máquinas no FJSP é conhecida como *The Dual Resource Constrained Flexible Job Shop Scheduling Problem* (DRCFJSP) (SUN; ZHU, 2005).

2.5 Flexible Job Shop Scheduling com Restrições de Trabalhadores com Tempos de *Setup* Antecipados (WSFJSP-SDST)

Esse estudo é baseado no problema worker constrained flexible job shop scheduling proposto por KRESS, MÜLLER e NOSSACK (2019). Como funções objetivos deste problema, foram considerados, separadamente, o tempo de término da última tarefa (*makespan*), o tempo total de atraso da entrega das tarefas (atraso total) e as seguintes suposições:

- Todas as máquinas e trabalhadores estarão disponíveis no instante 0;
- Uma vez que a operação é processada em uma máquina, ela não poderá ser interrompida (preempção);

- Os trabalhadores não poderão realizar mais de uma operação ao mesmo tempo;
- Cada operação requer, ao menos, uma máquina e um trabalhador;
- A operação só poderá ser processada em máquinas compatíveis com trabalhador que possua habilidade para processá-la;
- O tempo para realizar a operação é influenciado através da eficiência da habilidade do trabalhador e da máquina utilizada, ou seja, trabalhadores poderão realizar a mesma tarefa na mesma máquina com tempos distintos;
- O tempo de configuração das máquinas (*setup*) levará em consideração a operação anterior, a operação atual e a máquina que irá processá-la;
- As operações de configuração não exigem a atribuição de um trabalhador;
- O *setup* poderá ser antecipado quando possível.

2.6 Notação do Problema

Para resolver o FJSP computacionalmente, primeiramente, é preciso abstrair a realidade do problema para um conjunto de equações que o representem de forma qualitativa. Para isso, é preciso definir quais hipóteses serão utilizadas no modelo para representar o sistema real.

Sejam os índices I um conjunto de *jobs* onde $I \in \{1, ..., n\}$, M um conjunto de máquinas, W um conjunto de trabalhadores, cada *job* $i \in I$ possui um conjunto q_i de operações $O_i = \{i_1, ...i_{qi}\}$. M_{i_j} um conjunto de máquinas que podem processar a operação $O_i = \{i_1, ...i_{qi}\}$. M_{i_j,k_l} a interseção de M_{i_j} e M_{k_l} onde i, $k \in I$, $I_j \in O_i$, e $k_i \in O_k$. d_i representando a data de entrega do *job* $i \in I$. $P_{i,j}^{m,w}$ o tempo de processamento da operação I_j processada pela máquina m utilizando o trabalhador w. s_{i_j,k_j}^{m} o tempo de *setup* da operação I_j antecedido pela operação k_l na máquina m. C_{i_j} o instante de término do processamento da operação i_j. C_i o instante de término do processamento do *job* i. C_{max} o instante de término da última operação. T_i representando o atraso total do *job*. Na Tabela 3 é apresentado um resumo das notações utilizadas neste trabalho.

2.7 Modelo Matemático

O modelo matemático para o WSFJSP-SDST de KRESS, MÜLLER e NOSSACK (2019) foi dividido em duas partes. Na primeira parte, o problema master (MP) aborda a alocação das operações nas máquinas baseado no problema de roteamentos de veículos (VRP). Para isso, cada máquina foi considerada um veículo (Arestas) e, cada operação, um Cliente (Vértice). Representando o depósito tem-se um *job* fictício 0 com uma única

Tabela 3 – Notação utilizada neste trabalho.

I	Conjunto de *jobs*	$I \in \{1, ..., n\} \vert I\vert = n$
M	Conjunto de Máquinas	
W	Conjunto de Trabalhadores	
O_i	Conjunto de operações do *job* $i \in I$	$O_i = \{i_1, ..., i_{qi}\}$
M_{i_j}	Conjunto de máquinas que podem processar a operação O_i	$M_{i_j} \subseteq M$
M_{i_j,k_l}		
d_i	Data de Entrega do Job $i \in I$	$d_i \in \mathbb{Q}$
$P_{i_j}^{m,w}$	Tempo que o trabalhador w utiliza para processar a operação i_j utilizando a máquina m	$P_{i_j}^{m,w} \in \mathbb{Q}$
$P_{i_j}^{m,min}$	O menor tempo que um trabalhador nescessita para processar a operação i_j na m	$\min_{w \in W} P_{i_j}^{m,w}$
s_{i_j,k_j}^{m}	Tempo de *setup* Operação I_j antecedido pela operação k_l	$s_{i_j,k_j}^{m} \in \mathbb{Q}$
C_{i_j}	Instante de término do processamento da operação i_j	
C_i	Instante de término do processamento do job $i \in I$	$C_i = C_{i_j}$
C_{max}	*Makespan*	$C_{max} = \max_{I_j} Ci$
T_i	Atraso do Job I_j	$T_i = \max\{C_i - d_i, 0\}$

operação 0_1, com o tempo de processamento $P_{0,1}^{m,w} = 0$ para todas as máquinas m e trabalhadores w e *setup* igual a 0 ($s_{i_j,0_1}^{m} = 0$) para qualquer operação que a sucede.

Seguindo a formulação VRP, KRESS, MÜLLER e NOSSACK (2019) definiu o problema com um multigrafo direcionado ponderado $G = (V, E)$ onde $V = \bigcup_{i \in I} 0_i \cup \{0_1\}$ e $E = M_{i_j,k_l}$ onde, M_{i_j,k_l} representa a ligação direcional da operação i_j para a operação k_l, processada pela máquina M. Tem-se $\overline{V}$ como o conjunto de vértices desconsiderando o vértice do depósito (0_1). V_{i_j} é o conjunto de potenciais vértices sucessores de i_j, ou seja, desconsiderando os vértices antecessores do *job* i. $\tilde{V_{i_j}}$ conjunto de potênciais vértices sucessores retirando o vértice do depósito 0_1. Na Tabela 4 é apresentado um resumo das notações do WFJSP.

Tabela 4 – Notação do problema em Grafo $G = (V, E)$.

V	Conjunto de Vértices	$\bigcup_{i \in I} 0_i \cup \{0_1\}$
M_{i_j,k_l}	arco de ligação direcional da operação(Vértice) i_j para a k_l na máquina M	
$\overline{V}$	o conjunto de vértices desconsiderando o vértice depósito	$\overline{V} = V \backslash \{0_1\}$
V_{i_j}	potênciais sucessores de $i_j \in V$	$V_{i_j} = V \backslash \{i_k \vert k \leq j\}$
$\tilde{V_{i_j}}$	potênciais antecessores de $i_j \in V$	$V_{i_j} = V \backslash \{i_k \vert k \geq j\}$

Para entender melhor o modelo é importante entender as variáveis de decisão. t_{i_j} representa o tempo que iniciou o processamento da operação i_j. y_{i_j,k_l}^{m} tem o valor igual a 1

se a operação k_l é atendida imediatamente após a operação i_j.

$$\text{minimizar} \quad C_{\max} \tag{1}$$

sujeito a

$$t_{i_{qi}} + \sum_{k_l \in V_{i_{qi}}} \sum_{m \in M_{i_{qi},k_l}} y^m_{i_{qi},k_l} \cdot P^{m,\min}_{i_{qi}} \leq C_{\max} \qquad \forall\, i \in I, \tag{2}$$

$$\sum_{i_j \in \tilde{V}_{k_l}} \sum_{m \in M_{i_j,k_l}} y^m_{i_j,k_l} = 1 \qquad \forall\, k_i \in \overline{V}, \tag{3}$$

$$\sum_{i_j \in \overline{V}} y^m_{0_1,i_j} \leq 1 \qquad \forall\, m \in M, \tag{4}$$

$$\sum_{k_l \in \{a_b \in \tilde{V}_{i_j} | m \in M_{a_b}\}} y^m_{k_l,i_j} - \sum_{k_l \in \{a_b \in V_{i_j} | m \in M_{a_b}\}} y^m_{i_j,k_l} = 0 \qquad \forall\, i_j \in V, m \in M_{i_j}, \tag{5}$$

$$t_{i_j} + s^m_{i_j,k_l} + P^{m,min}_{i_j} - t_{k_l} \leq \left(1 - y^m_{i_j,k_l}\right) B \qquad \forall\, i_j \in V, k_l \in \overline{V}_{i_j}, m \in M_{i_j,k_l}, \tag{6}$$

$$t_{i_j} + \sum_{k_l \in V_{i_j}} \sum_{m \in M_{i_j,k_l}} y^m_{i_j,k_l} \cdot P^{m,\min}_{i_j} \leq t_{i_{j+1}} \qquad \forall\, i_j \in \overline{V} comj \leq q_i - 1, \tag{7}$$

$$t_{i_j} + P^{min}_{i_j} \leq t_{i_{j+1}} \qquad \forall\, i_j \in \overline{V} comj \leq q_i - 1, \tag{8}$$

$$\tilde{C}^h_{\max} \cdot \left(1 - \sum_{(i_j,k_l,m) \in P^h} (1 - y^m_{i_j,k_l})\right) \leq C_{\max} \qquad \forall\, \textit{inequações lógicas } h, \tag{9}$$

$$y^m_{i_j,k_l} \in \{0,1\} \qquad \forall\, i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l} \tag{10}$$

$$t_{i_j} \in \mathbb{R}^+_0 \qquad \forall\, i_j \in V, \tag{11}$$

$$C_{\max} \in \mathbb{R}^+_0 \qquad \forall\, i_j \in V, k_l \in V_{i_j}, m \in M_{i_j,k_l} \tag{12}$$

A função objetivo (1) visa minimizar o makespan. As restrições do tipo (2) geram o lower bound do makespan. As restrições (3) asseguram que cada operação vai ser atendida exatamente uma vez. As restrições (4) garantem que existirá no máximo uma operação para a máquina iniciar o processamento. O fluxo é assegurado na restrição (5) ou seja, se existir uma operação que precede a operação i_j vai existir uma operação que a antecede. As restrições do tipo (6) garantem o tempo de execução da operação, ou seja, se a operação k_l antecede a operação i_j na máquina m, então a operação k_l terá o seu início de processamento no término de processamento da operação i_j. As restrições do tipo (7) e (8) garantem as de precedência dos *jobs*. As inequações (9) servem para corrigir a superestimativa $P^{m,min}_{i_j}$, que busca o menor tempo para executar a operação. As restrições (10 à 12) definem a domínio das variáveis.

Para considerar a função objetivo atraso total, o modelo é adaptado em:

$$\text{minimizar} \quad \sum_{i \in I} T_i \tag{13}$$

sujeito a (3)-(8), (10), (11)

$$t_{i_{qi}} + \sum_{k_l \in V_{i_{qi}}} \sum_{m \in M_{i_{qi},k_l}} y^m_{i_{qi},k_l} \cdot P^{m,\min}_{i_{qi}} - d_i \leq T_i \qquad \forall i \in I, \tag{14}$$

$$\tilde{T}^h \cdot \left(1 - \sum_{(i_j,k_l,m) \in P^h} (1 - y^m_{i_j,k_l})\right) \leq \sum_{i \in I} T_i \qquad \forall \textit{ inequações lógicas } h, \tag{15}$$

$$T_i \in \mathbb{R}^+_0 \qquad \forall\, i_j \in V \tag{16}$$

A função objetivo (13) minimiza o atraso total. As restrições do tipo (14) geram o lower bound do atraso total. A restrição do tipo (15) é análoga a (9). As restrições (16) definem a domínio das variáveis.

O resultado deste MP fornece uma estrutura capaz de criar um grafo direcionado, composto por no máximo |M| subgrafos disjunto, onde cada subgrafo é um grafo circular. Na Figura 6a tem-se um exemplo de uma possível solução fornecida pela MP transformada em um grafo, onde as arestas pontilhadas simbolizam a máquina 2, e as arestas sólidas simbolizam a máquina 1. Com essa solução também é possível obter o agendamento da solução, disposta em 6b.

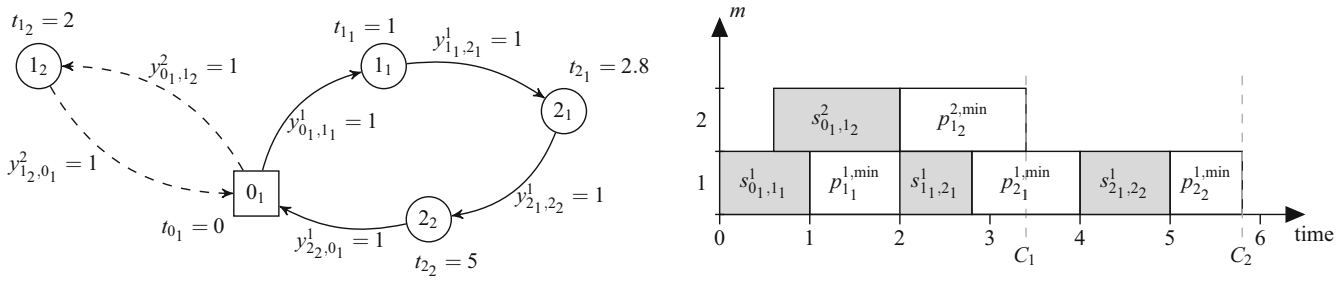

Figura 6 – Exemplo resultado. Fonte: KRESS, MÜLLER e NOSSACK (2019).

Considerando que o problema master (MP) aborda somente a alocação das operações nas máquinas, KRESS, MÜLLER e NOSSACK (2019) propuseram um subproblema que recebe uma solução do MP e resolve o problema de alocação dos trabalhadores.

Para entender o modelo desse subproblema, KRESS, MÜLLER e NOSSACK (2019) definem algumas notações: $\overline{y}^m_{i_j,k_l}$ é o vetor de solução fornecida pelo MP. $\overline{z}^m_{i_j}$ é igual a um, quando a operação i_j é processada na máquina m não sendo antecedida pela operação 0_1. A variável $x^w_{i_j}$ é igual a um quando ela é processada pelo trabalhador w. $x^w_{i_j,k_l}$ é igual a um se ambas operações i_j, k_l são processadas pelo trabalhador w. v_{i_j,k_l} é igual a um quando a operação k_i é processada antes da operação i_j. Também é definido que a operação 0_1 pode

ser atendida por qualquer trabalhador. O problema é definido pelo autor como:

$$\text{minimizar} \quad \tilde{C}_{\max} \tag{17}$$

sujeito a

$$\tilde{t}_{i_{qi}} + \sum_{m \in M_{i_{qi}}} \sum_{w \in W} \overline{z}^m_{i_{qi}} \cdot x^w_{i_{qi}} \cdot P^{m,w}_{i_{qi}} \leq C_{\max} \qquad \forall i \in I, \tag{18}$$

$$\sum_{w \in W} x^w_{i_j} = 1 \qquad \forall\ i_j \in \overline{V}, \tag{19}$$

$$\tilde{t}_{i_j} + s^m_{i_j,k_l} + P^{m,w}_{i_j} - \tilde{t}_{k_l} \leq \left(1 - \overline{y}^m_{i_j,k_l} \cdot x^w_{i_j}\right) B \qquad \forall i_j \in V, k_l \in \overline{V}_{i_j}, m \in M_{i_j,k_l}, w \in W, \tag{20}$$

$$\tilde{t}_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} \overline{z}^m_{i_j} \cdot x^w_{i_j} \cdot P^{m,w}_{i_j} \leq \tilde{t}_{i_{j+1}} \qquad \forall\ i_j \in \overline{V}\ com\ j \leq q_i - 1, \tag{21}$$

$$x^w_{i_j,k_l} \geq x^w_{i_j} + x^w_{k_l} - 1 \qquad \forall\ i_j, k_l \in \overline{V}, i_j \neq k_j, w \in W, \tag{22}$$

$$\tilde{t}_{i_j} + \sum_{m \in M_{i_j}} \sum_{w \in W} \overline{z}^m_{i_j} \cdot x^w_{i_j} \cdot P^{m,w}_{i_j} - \tilde{t}_{k_l} \leq B \cdot v_{i_j,k_l} \qquad \forall\ i_j, k_l \in \overline{V}, i_j \neq k_j, \tag{23}$$

$$x^w_{i_j,k_l} \leq 2 - v_{i_j,k_l} - v_{k_l,i_j} \qquad \forall\ i_j, k_l \in \overline{V}, i_j \neq k_j, w \in W, \tag{24}$$

$$x^w_{i_j} \in \{0,1\} \qquad \forall i_j \in V, w \in W, \tag{25}$$

$$x^w_{0_1} = 1 \qquad \forall\ w \in W, \tag{26}$$

$$x^w_{i_j,k_l} \in \{0,1\} \qquad \forall\ i_j, k_l \in \overline{V}, i_j \neq k_j, w \in W, \tag{27}$$

$$v^w_{i_j,k_l} \in \{0,1\} \qquad \forall\ i_j, k_l \in \overline{V}, i_j \neq k_j, \tag{28}$$

$$\tilde{t}_{i_j} \in \mathbb{R}^+_0 \qquad \forall\ i_j \in V, \tag{29}$$

$$\tilde{C}_{\max} \in \mathbb{R}^+_0 \tag{30}$$

Fornecido um vetor $\overline{y}$ de solução do MP, a função objetivo makespan é minimizada em (17). As restrições (18) certificam que o makespan não poderá ter valor inferior ao tempo de término de qualquer operação. As restrições (19) asseguram que cada operação terá exatamente um trabalhador. As restrições (20) garantem o tempo de execução da operação. As restrições (21) garantem a ordem de precedência dos *jobs*. As restrições (22) asseguram a integridade das variáveis $x^w_{i_j,k_l}$ com x_{i_j} e x_{k_l}. As restrições (23) asseguram a variável v_{i_j,k_l}. As restrições (24) garantem que um trabalhador não pode anteder duas operações ao mesmo tempo. As restrições (25) – (30) definem a domínio das variáveis.

Para considerar o objetivo atraso total, o subproblema é adaptado em:

$$\text{minimizar} \quad \sum_{i \in I} \tilde{T}_i \tag{31}$$

sujeito a (19)-(29)

$$\tilde{t}_{i_{qi}} + \sum_{m \in M_{i_{qi}}} \sum_{w \in W} \overline{z}^m_{i_{qi}} \cdot x^w_{i_{qi}} \cdot P^{m,w}_{i_{qi}} - d_i \leq \tilde{T}_i \qquad \forall i \in I, \tag{32}$$

$$\tilde{T}_i \in \mathbb{R}^+_0 \qquad \forall\ i_j \in V \tag{33}$$

A função objetivo atraso total é minimizada em (31). As restrições (32) garantem os limites para calcular o atraso de cada operação. As restrições (33) garantem o domínio da variável do atraso total.

3 REVISÃO DA LITERATURA

Proposto por BRUCKER e SCHLIE (1990), o primeiro estudo do *Job Shop Scheduling Flexível* (FJSP) ocorreu em 1990. Os autores implementaram um algoritmo em tempo polinomial para resolver o problema com apenas duas tarefas. Desde então, o problema vem recebendo uma quantidade significativa de publicações nos últimos anos (CHAUDHRY; KHAN, 2016; BISSOLI; ZUFFEREY; AMARAL, 2021). Alguns desses estudos demonstram a necessidade de considerar outras restrições para o modelo atender melhor a realidade das indústrias. Uma delas, foi a reflexão sobre a existência de empresas que não possuem seu sistema produtivo automatizado, e em certos casos, precisam de trabalhadores para operarem as máquinas. Na literatura, este problema é conhecido como *Dual-resource constrained FJSP* (DRCFJSP). Contudo, recentemente, os autores (KRESS; MÜLLER; NOSSACK, 2019), foram mais específicos, e em vez de tratar essa restrição como *dual*, utilizam o termo *worker*, definindo a nomenclatura como *Worker Constrained Flexible Job Shop Scheduling Problem* (WFJSP). Neste trabalho, são utilizados, respectivamente, as novas nomenclaturas WFJSP e WFJSP-SDST para referenciar os problemas DRC-FJSP e DRC-FJSP com setups dependentes da sequência.

Tornando o modelo mais genérico, o WFJSP ganhou relevância na literatura, obtendo um crescente número de publicações. Diferentes abordagens são encontradas resolvendo o WFJSP (DHIFLAOUI; NOURI; DRISS, 2018). WIROJANAGUD et al. (2007) propuseram um Programação inteira mista e linear (MILP) considerando os critérios makespan e alguns fatores humanos, como custo de treinamento e salário. Utilizando a meta-heurística genético (GA) (CHAUDHRY; DRAKE, 2009) resolveu o WFJSP minimizando o atraso total. Além disso, o problema foi resolvido de forma multi-objetivo. WANG et al. (2017) implementaram a meta-heurística *hybrid discrete particle swarm optimization* (HDPSO) para resolver o problema bi-objetivo, obtendo como resultado a curva de Pareto para os objetivos makespan e custo dos recursos (máquinas e trabalhadores).

Outra característica importante introduzida neste problema de escalonamento, a fim de torná-lo genérico, foi o tempo de configuração das máquinas (*setup*). O FJSP com tempos de setup já foi resolvido de diferentes maneiras na literatura (ALLAHVERDI et al., 2008). Considerando resoluções que utilizaram métodos exatos, RABADI, MOLLAGHASEMI e ANAGNOSTOPOULOS (2004) propuseram o algoritmo *branch-and-bound* utilizando o atraso total como objetivo. Algumas versões de algoritmos genéticos (GA) propostas obtiveram bons resultados. O GA proposto por ARMENTANO e MAZZINI (2000) obteve 93% de soluções melhores que a heurística *three phases* desenvolvida por (LEE; BHASKARAN; PINEDO, 1997). TAN e NARASIMHAN (1997) demonstraram a eficiência

da meta-heurística *simulated annealing* (SA) aplicada a este problema, encontrando, em geral, soluções mais eficientes que a *random search*.

É importante notar um bom desempenho da meta-heurística SA em encontrar boas soluções para o FJSP e suas extensões. CRUZ-CHÁVEZ, MARTÍNEZ-RANGEL e CRUZ-ROSALES (2017) implementaram a *SA-Partial Controlled* (SA-PC), que encontrou três soluções melhores que a *busca tabu* implementadas por MASTROLILLI e GAMBARDELLA (2000). BISSOLI et al. (2018) propuseram um SA para o FJSP bi-objetivo, e evidenciaram que mesmo a heurística sendo bi-objetivo, obteve quatro soluções superiores e três inferiores quando comparada com a SA-PC.

Com o objetivo de melhorar a eficiência dos resultados da SA, na literatura verifica-se a sua utilização em estratégias híbridas. AKRAM, KAMAL e ZEB (2016) propuseram uma nova abordagem híbrida da SA que combina os algoritmos *Fast Simulated Annealing* (FSA) com o *Quenching* (HFSAQ). A hibridação ocasionou na melhora dos resultados de 62 das 88 instâncias avaliadas. LI et al. (2010) combinaram o algoritmo *Colônia de Formigas* (ACO) com a SA formando o (TACOSA) para resolver o problema WFJSP. Seus resultados foram comparados com a meta-heurística *Elite Ant System* (EAS). A TACOSA obteve soluções superiores em todas as instâncias testadas. Posteriormente, ALTOÉ et al. (2018) utilizaram a SA como geradora de soluções para a meta-heurística *Clustering Search* (CS), obtendo 6 das 10 melhores soluções conhecidas para o FJSP, evidenciando ser um algoritmo eficiente para esse tipo de problema. Não foram encontradas implementações da CS para o WFJSP-SDST com tempos de setup adiantados, desta maneira, torna-se interessante sua investigação.

Existem poucos artigos na literatura que consideram, simultaneamente, a restrição de trabalhadores com custos relacionados aos tempos de setup. YAZDANI et al. (2015) propuseram um modelo MILP e as meta-heurísticas SA e *Vibration Damping Optimization* (VDO) para solucionar o WFJSP-SDST com tempos de setups independentes da sequência. O tempo de setup foi incorporado no tempo de processamento de cada tarefa, contudo, salientou-se a importância de trabalhar com setup dependentes da sequência. Em seguida, KRESS, MÜLLER e NOSSACK (2019) forneceram 10 instâncias de um problema real no ambiente WSFJSP com setup dependentes da sequência, considerando a minimização separadamente do makespan e atraso total. O problema foi solucionado através de algoritmos exatos e heurísticos, dentre eles destacam-se: um algoritmo *Branch-and-Cut* utilizando o solver CPLEX, denominado de E-DM; e duas heurísticas, H-DM que utiliza o CPLEX em duas etapas, a primeira etapa para determinar as soluções viáveis para o problema master e a segunda etapa para determinar uma solução viável dos subproblemas gerados. Diferentemente da primeira heurística, a segunda heurística H-DMB utiliza um método *beam search* na segunda etapa. Com os experimentos realizados, notou-se que, quanto maior a quantidade de jobs na instância, menos eficientes foram os métodos. Não foi

encontrado nenhum trabalho aplicando meta-heurísticas ao problema estudado. Motivados por isso, foram implementadas duas meta-heurísticas, a SA e a CS para solucionar as instâncias reais do WSFJSP dispostas por KRESS, MÜLLER e NOSSACK (2019).

4 METODOLOGIA PARA A RESOLUÇÃO DO PROBLEMA WSFJSP-SDST

4.1 Heurística Construtiva

Neste trabalho é proposta uma adaptação da heurística construtiva utilizada em BISSOLI et al. (2018), fornecendo assim, uma solução inicial para a meta-heurística utilizada neste estudo.

A heurística construtiva gera uma solução inicial baseada em uma instância do problema. Na Tabela 5, é apresentado um exemplo de uma instância do problema, onde $O_{i,j}$ simboliza a operação j do *job* i. Este exemplo é constituído por dois *jobs*. O primeiro, composto por duas operações ($O_{1,1}$,$O_{1,2}$), e o segundo, composto apenas pela operação $O_{2,1}$. O campo ID identifica cada possibilidade de processamento de uma dada operação, definindo uma máquina, um trabalhador e o tempo para processá-la. Os trabalhadores W_1 e W_2 podem processar a operação $O_{1,1}$ utilizando as máquinas M_1 e M_2, respectivamente. A operação $O_{1,2}$ só pode ser processada pelo trabalhador W_2 utilizando a máquina M_2. Por fim, o trabalhador W_3 pode processar a operação $O_{2,1}$ utilizando a máquina M_2. Também foi considerado um tempo de *setup* igual a um para todas as operações.

Tabela 5 – Exemplo do tempo de processamento de uma instância.

Job	Operação	ID	Máquina	Trabalhador	Tempo
J_1	$O_{1,1}$	1	1	1	5
		2	2	2	2
	$O_{1,2}$	1	2	2	2
J_2	$O_{2,1}$	1	2	3	3

A fim de garantir a ordem de precedência das operações, foi criado um vetor com as operações dos *jobs* denominado de "Sorte". Essas operações são substituídas pelo respectivo *job*, ou seja, as operações $O_{1,1}$, $O_{1,2}$ e $O_{2,1}$ são substituídas por 1, 1 e 2, respectivamente, conforme a Figura 7.

Sorte	1	1	2

Figura 7 – Exemplo de estrutura auxiliar para criação da solução.

A solução é representada por um vetor “Ordem” que caracteriza a ordem dos *jobs* sorteados e uma matriz de J linhas por O_J colunas, onde J representa o número de *jobs*, e O_J representa o número de operações do *job* J. A fim de facilitar a estrutura de vizinhança, o conteúdo da matriz é composto pelos IDs da instância na Figura 8.

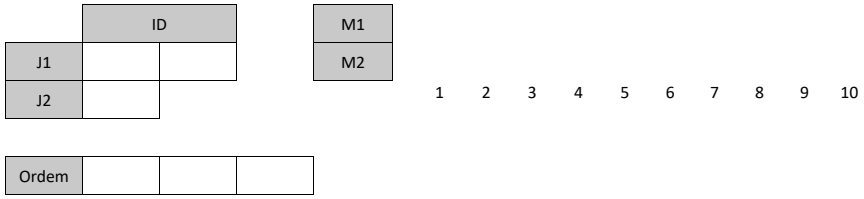

Figura 8 – Estrutura da Solução proposta Vazia.

Para criar uma solução inicial, o algoritmo, primeiramente, preenche a matriz aleatoriamente respeitando os ID possíveis da Tabela 5. Portanto, as operações $O_{1,2}$ e $O_{2,1}$ só poderão conter o ID igual a 1. Contudo, a operação $O_{1,1}$ pode conter o ID 1 ou 2. Supondo que foi sorteado o ID 1 para a operação $O_{1,1}$, a solução é representada pela Figura 9.

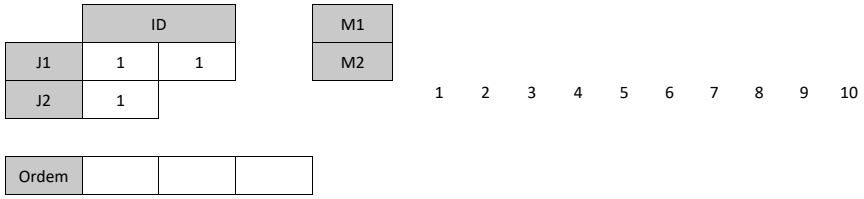

Figura 9 – Estrutura da Solução proposta parcialmente preenchida.

Na segunda etapa da criação da solução, temos a definição da ordem em que as operações serão atendidas. Para isso, é realizado um sorteio aleatório das operações dos *jobs* contidos no vetor *Sorte*. Considerando que foram sorteados os *jobs* (J_1,J_1,J_2), respectivamente, tem-se a solução representada na Figura 10.

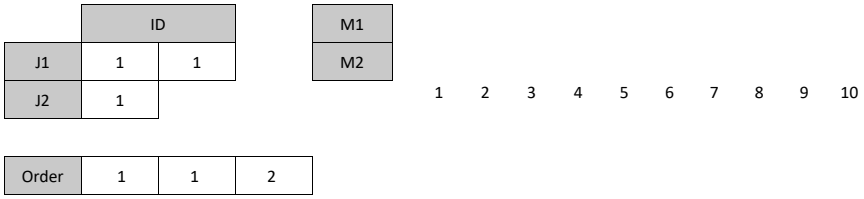

Figura 10 – Estrutura da Solução proposta preenchida.

Definindo a ordem em que as operações serão inseridas nas máquinas, é possível criar o gráfico da solução, possibilitando assim, a visualização do agendamento. Considerando o vetor *Sorte* na Figura 7, a primeira operação a ser processada será a operação disponível, respeitando a ordem de precedência do *job* J_1, ou seja, a operação $O_{1,1}$. Contudo, antes da inserção do *job* na máquina, é preciso inserir o *setup* da máquina para receber essa operação. Após a inserção do *setup*, identifica-se que a operação $O_{1,1}$ foi processada pelo ID igual a um (Figura 10), ou seja, será processada pelo trabalhador W_1, na máquina M_1, com duração igual a 5. Seguindo esta lógica, foi criado o agendamento na Figura 11.

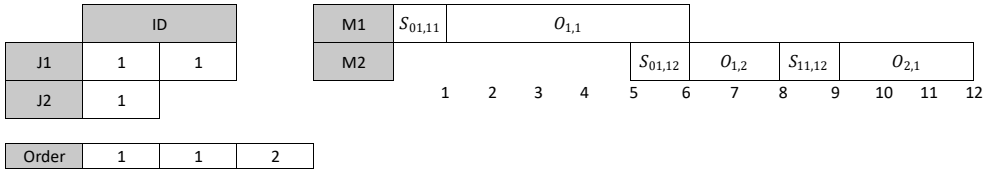

Figura 11 – Um possível agendamento do exemplo proposto.

4.2 Estrutura de Vizinhança

Com o intuito de buscar boas soluções, as meta-heurísticas possuem a característica de explorar a vizinhança de forma iterativa, realizando trocas entre os vizinhos. Neste trabalho foram utilizadas duas formas de explorar a vizinhança, N1 e N2, adaptadas para considerar trabalhadores de BISSOLI, ZUFFEREY e AMARAL (2021).

4.2.1 N1 - Troca de Codificador

A estratégia desta vizinhança constitui-se em alterar a máquina ou o operador que está processando a operação. Para isso, é realizada a troca no ID da matriz *jobs* em uma operação que possui flexibilidade, ou seja, mais de um ID, por exemplo, a operação $O_{1,1}$ da Tabela 5.

A Figura 12 representa um exemplo de troca N1. Foi sorteada a operação $O_{1,1}$ sendo realizada a alteração do seu ID de 1 para 2. Com isso, agora a operação $O_{1,1}$ passa a ser processada pelo trabalhador W_2 na máquina M_2.

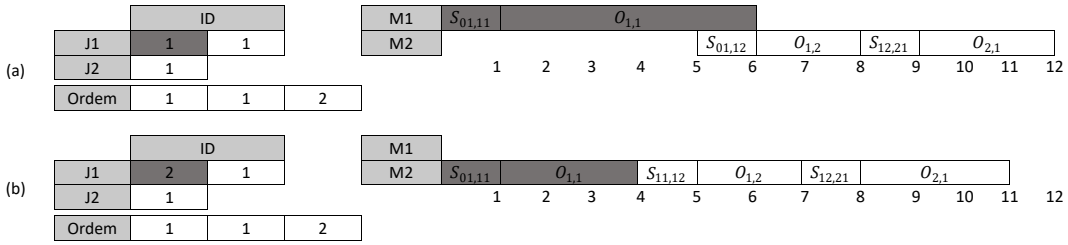

Figura 12 – Exemplo Vizinhança N1 onde (a) representa a solução antes da troca e (b) depois da troca.

4.2.2 N2 - Troca Entre Operações

A estratégia desta vizinhança constitui-se em alterar a ordem de processamento das operações na máquina. Primeiramente, são selecionadas, aleatoriamente, duas operações no vetor "Ordem" e, então, é realizada a troca de posições.

Na Figura 13, tem-se o exemplo da vizinhança N2. Em (a) tem-se um exemplo de uma solução na qual, foram selecionadas a primeira e a terceira posição do vetor "Ordem" para realizarem a troca (em cinza escuro). Em (b) temos o resultado da vizinhança. Neste

momento, o agendamento é recriado e a primeira solução a ser inserida será a $O_{2,1}$, conforme o vetor "Ordem".

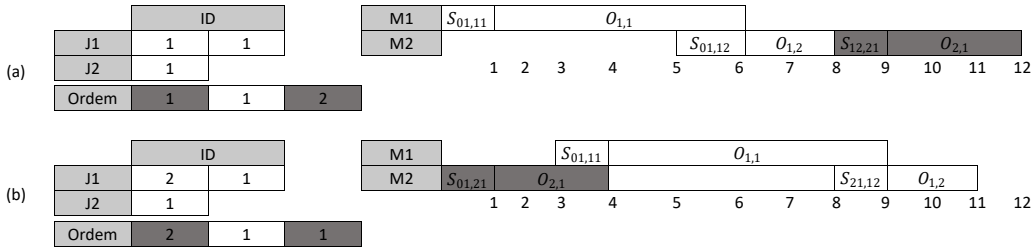

Figura 13 – Exemplo Vizinhança N2.

4.3 Simulated Annealing Aplicado ao WSFJSP-SDST

Proposta por KIRKPATRICK, GELATT e VECCHI (1983), a *Simulated Annealing* (SA) baseia-se no processo utilizado para fundir metais, o qual o metal é aquecido a uma temperatura elevada e em seguida é resfriado lentamente, tornando o produto cada vez menos flexível e mais homogêneo.

A primeira estratégia a qual a temperatura é aumentada, é denominada como caminhos aleatórios no qual, seu objetivo, é explorar o espaço de busca, aceitando soluções piores. Entretanto, quando a temperatura diminui, esse comportamento é decrescido, diminuindo a probabilidade de aceitar uma solução pior. O objetivo desta etapa é definir o metal, convergindo a solução em um ótimo local, esta etapa é conhecida como melhoramento iterativo (KIRKPATRICK; GELATT; VECCHI, 1983).

A SA recebe cinco parâmetros de entrada, sendo α a taxa de resfriamento, SA_{max} o número máximo de iterações, T_0 a temperatura inicial, T_C a temperatura de congelamento e s uma solução inicial qualquer. Neste artigo, foi utilizada como solução inicial a heurística da Seção 4.1.

O Algoritmo 1 apresenta o pseudocódigo da SA. O primeiro passo da SA consiste em guardar a solução s em uma estrutura responsável por manter a melhor solução, s^{best}, na Linha 1. Na linha 6, é gerado um vizinho aleatório de s com probabilidade de 60% de ser um vizinho do tipo N1 (Seção 4.2.1) e 40% de ser um vizinho N2 (Seção 4.2.2), representado por s'. Em seguida (Linha 7), calcula-se Δ, que é o valor da função objetivo de s decrementado de s'. Com isso, verifica-se se Δ é menor que zero, ou seja, se a solução vizinha s' é melhor que a solução atual s (Linha 8). Caso afirmativo, s passará a ser a solução vizinha s' (Linha 9). Verifica-se, também (Linha 10), se s tornou-se mais eficiente que s^{best}, caso afirmativo, s^{best} obtém uma cópia da solução s (Linha 11). Entretanto, se Δ for maior que zero, é gerado um número aleatório no intervalo [0, 1], e verifica-se se esse número é menor que $e^{(-\Delta/T)}$, para que s aceite o vizinho pior (Linhas 15, 16 e 17). A cada SA_{max} iterações, a temperatura da SA diminui (Linha 21), reduzindo, assim, as

chances de aceitar uma solução pior (Linha 16). Quando a temperatura T da SA chega a um valor próximo de zero (Linha 4), a temperatura é reiniciada (Linha 3). Todos esses passos são seguidos até que o algoritmo atinja seu tempo limite pré-estabelecido (Linha 2).

Algoritmo 1: *Simulated Annealing*

Entrada: α, SA_{max}, T_0, T_C, s
Saída: Solução s^{best}

```
1  s^best ← s;
2  enquanto tempExec < tempoLimit faça
3  |  T ← T_0;
4  |  enquanto T > T_C faça
5  |  |  para it ← 1 até SAmax faça
6  |  |  |  s' ← N(s);
7  |  |  |  Δ ← f(s') - f(s);
8  |  |  |  se Δ < 0 então
9  |  |  |  |  s ← s';
10 |  |  |  |  se f(s') < s^best) então
11 |  |  |  |  |  s^best ← s';
12 |  |  |  |  fim
13 |  |  |  fim
14 |  |  |  senão
15 |  |  |  |  Gere um número aleatório p ∈ [0,1] ;
16 |  |  |  |  se p < e^(-Δ/T) então
17 |  |  |  |  |  s ← s';
18 |  |  |  |  fim
19 |  |  |  fim
20 |  |  fim
21 |  |  T ← α * T;
22 |  fim
23 fim
```

4.4 Clustering Search Aplicado ao WSFJSP-SDST

A *Clustering Search* (CS) é uma proposta genérica da meta-heurística *Evolutionary Clustering Search* (ECS). Diferentemente do ECS, o CS pode utilizar de qualquer meta-heurística como seu mecanismo de gerar solução. É algoritmo híbrido iterativo que procura dividir o espaço de busca a fim de localizar regiões promissoras que, quando identificadas, é realizada uma pesquisa mais profunda nessas regiões (OLIVEIRA; CHAVES; LORENA, 2013).

Cada espaço de busca é conhecido como *Cluster*, que possui um centro C representado pela melhor solução presente no *Cluster*, um volume V indicando a quantidade de soluções agrupadas no *cluster* C e um índice de ineficácia ri indicando a quantidade de iterações, na qual a busca local não obteve melhora.

Segundo OLIVEIRA, CHAVES e LORENA (2013), a CS pode ser entendida em quatro partes independentes. A seguir aborda-se cada um desses componentes.

4.4.1 Uma Meta-heurística Geradora de Soluções

Inicialmente os *Clusters* são preenchidos com soluções construídas aleatoriamente e, a cada iteração da CS, é executada uma meta-heurística para se adequar a um *cluster*. Neste trabalho, o *Simulated Annealing* (SA) foi utilizado como a meta-heurística geradora de solução.

4.4.2 Agrupamento Iterativo

O objetivo desta etapa é alocar as soluções geradas pela etapa anterior no *cluster* mais próximo. Neste trabalho é utilizada a distância de Hamming como medida de distância. Este cálculo é feito através da contagem do número de desigualdades entre uma solução e o centro do *cluster*. A Figura 14 apresenta duas soluções e o cálculo da distância, os quadrados destacados na cor cinza escuro identificam os pontos distintos entre as soluções. No final desta etapa, a solução será alocada ao *cluster* com menor distância, ou seja, com menos valores distintos.

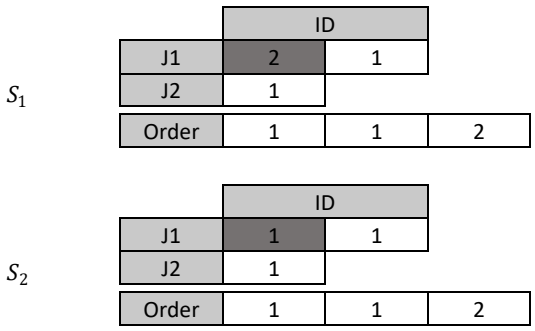

Figura 14 – Apresenta um exemplo da Distância de Hamming = 1.

4.4.3 Analisador de Agrupamentos

Após a etapa Agrupamento Iterativo, é iniciado o analisador de agrupamentos. Nesta fase, interessa saber se o *cluster* representa uma região promissora. Para isso, é verificado se o seu volume atingiu o limite máximo λ. Caso esta premissa seja verdadeira, é realizada a busca local no centro do *cluster*, incrementando o índice de ineficiência ri. Se ri for maior que r_{max}, uma perturbação é realizada ao centro do *cluster*.

Além de ser responsável por reunir as soluções similares dentro do *cluster*, o Analisador de Agrupamento se preocupa em atualizar o centro do *cluster*, mantendo a solução mais representativa como centro. Neste trabalho, a solução representativa é a seleção da solução com menor função objetivo presente no *cluster*.

4.4.4 Busca Local

Após identificada uma área promissora, deve-se refinar a solução. Para isso, é realizada uma busca local no centro do *cluster*, convergindo a solução para um ótimo local (OCHOA; VEREL; TOMASSINI, 2010; HANSEN; MLADENOVIĆ, 2006).

Como busca local, foi implementado o algoritmo melhor melhora (HANSEN; MLADENOVIĆ, 2006). Recebendo uma solução por parâmetro, o algoritmo melhor melhora gera todos os vizinhos da solução e, logo após, verifica qual destas soluções apresenta o melhor valor de função objetivo (FO). Se a FO foi melhorada, a solução com a melhor FO toma o seu lugar e o passo anterior é repetido até que não haja mais melhora (OCHOA; VEREL; TOMASSINI, 2010).

4.4.5 Clustering Search

O pseudocódigo da *Clustering Search* (CS) utilizada neste trabalho é apresentado no Algoritmo 2. A CS recebe os cinco parâmetros da SA, também recebe mais três parâmetros de entradas: γ correspondente ao número de *clusters*; λ representando o volume máximo dos *clusters* e; r_{max} sendo o índice de ineficácia. A Primeira etapa da CS consiste na criação dos γ *clusters*. Para isso, na linha 1, cada *cluster* recebe uma solução aleatória, a qual inicialmente, representa o centro do *cluster*. Com isso, o volume V de cada *cluster* é igual a um, e seu índice de ineficácia r é atribuída com 0, pois não foi realizada nenhuma busca local até o momento. Após a inicialização dos *clusters*, a SA é executada e, a cada decremento de temperatura (linha 19), é gerada uma solução s para a CS (linhas 7 a 18). Com isso, inicia-se o processo de agrupamento iterativo, onde, é selecionado o *cluster* mais próximo da solução s, utilizando a menor distância de *Hamming*, seção 4.4.2 (linha 19). A solução s é adicionada ao *cluster* i mais próximo, com isso, o volume do *cluster* i é incrementado então, verifica-se se a solução atual s é melhor que a solução que representa o centro do *cluster* C_i. Em caso afirmativo, o centro do *cluster* é atualizado com a nova solução (linha 19). Na linha 20, é avaliado se o *cluster* é uma região promissora, ou seja, se ele atingiu seu volume máximo definido por λ. Caso afirmativo, aplica-se uma busca local na solução que representa o centro do *cluster* (linha 22), e o seu volume é reiniciado (linha 21). Em seguida, realiza-se uma avaliação se a busca local obteve melhora (linha 23). Caso não obtenha melhora, o índice de ineficácia r_i é incrementado (linha 24) e, se este atingiu o índice máximo de ineficácia, o centro do *cluster* recebe uma nova solução aleatória e seu índice de ineficácia r_i é zerado. Porém, caso a busca local, melhore o centro do *cluster*, atribui-se esta nova solução ao centro do *cluster* e seu índice de ineficácia r_i é zerado (linhas 30 e 31). O algoritmo CS é executado até que o tempo limite pré-estabelecido seja

alcançado (linha 3).

Algoritmo 2: Clustering Search

Entrada: (α, SA_{max}, T_0, T_C, s, γ, λ , r_max)

```
1  C_i ← novaSolução(); v_i ← 1; r_i ← 0; ∀ i = 1,...,γ ;
2  s^best ← s;
3  enquanto execTime < timeLimit faça
4      T ← T_0;
5      enquanto T > T_C faça
6          it ← 0;
7          para it ← 1 até SAmax faça
8              s' ← N(s);
9              Δ ← f(s') - f(s);
10             se △ < 0 então
11                 s ← s';
12                 se (f(s') < f(s^best)) então
13                     s^best ← s';
14                 fim
15             fim
16             senão
17                 s ← s', with probability e^(−△/T)
18             fim
19         fim
20         T ← α * T; i ← minHamming(c, s); C_i ← best(C_i, s) ;
21         se (v_i = λ) então
22             v_i ← 0;
23             s ← localSearch(C_i);
24             se ( f(s) = f(C_i) ) então
25                 r_i ← r_i + 1 ;
26                 se (r_i = r_max) então
27                     r_i ← 0 ;
28                     C_i ← newSolution() ;
29                 fim
30             fim
31             senão
32                 r_i ← 0 ;
33                 C_i ← s ;
34             fim
35         fim
36         se ( f(s^best) > f(C_i) ) então
37             s^best ← C_i;
38         fim
39     fim
```

5 EXPERIMENTOS COMPUTACIONAIS

Os experimentos foram conduzidos em um computador com processador Intel Core i5-3450 de 3.10GHz, 8GB de RAM e sistema operacional Windows 7. O algoritmo foi implementado na linguagem C++ utilizando o compilador Visual C++.

As definições dos parâmetros do algoritmo são apresentadas na Seção 5.1, assim como a descrição das instâncias utilizadas são abordadas a seguir. Neste trabalho foram considerados dois conjuntos de instâncias. Com o objetivo de facilitar a comparação com os resultados obtidos por outros algoritmos propostos na literatura, o primeiro conjunto de instâncias utilizado foi baseado em um problema real de uma empresa de construção alemã dispostas em KRESS, MÜLLER e NOSSACK (2019). Para uma efetiva comparação entre as meta-heurísticas propostas, foram geradas 10 instâncias adaptadas de BRANDIMARTE (1993), onde os resultados são apresentados na Seção 5.2.2.

5.1 Definição dos Parâmetros

Com o intuito de melhorar a eficiência nos resultados, os parâmetros das meta-heurísticas foram calibrados baseados em RIBEIRO, LAPORTE e MAURI (2012), RABELLO et al. (2014) onde, foi definido um conjunto inicial de valores para cada parâmetro e, para cada valor, foram executadas todas as permutações com os valores definidos dos outros parâmetros. Para cada permutação dos parâmetros, o SA foi executado 5 vezes durante 10 minutos para 3 instâncias diferentes (R1,R5,R10). A Tabela 6 apresenta o conjunto de valores de cada parâmetro utilizado no teste, na qual SA_{max} representa a quantidade de iterações da SA, α indica a taxa de resfriamento, T_0 como a temperatura inicial, T_C representa a temperatura de congelamento, γ indica a quantidade de clusters, λ é o volume máximo para cada cluster e r_{max} é o índice de ineficácia para cada cluster.

Tabela 6 – Calibração parâmetros das meta-heurísticas.

SA	SA_{max}	1000	3000	5000	8000	8500	9000
	α	0,96	0,97	0,98	0,985	0,99	0,995
	T_0	100	200	400	600	800	1000
	T_C	0,1	0,08	0,05	0,01	0,0025	0,001
CS	γ	5	10	15	20	25	30
	λ	10	20	30	40	50	60
	r_{max}	2	2	3	4	5	6

Após executar este conjunto de possibilidades, identificou-se a melhor solução e o desvio padrão de cada permutação. A Tabela 7 apresenta o conjunto que obteve a melhor solução com o menor desvio padrão.

Tabela 7 – Valores dos parâmetros das meta-heurísticas.

SA	SA_{max}	8500
	T_O	1000
	T_C	0.01
	α	0.995
CS	γ	20
	λ	60
	r_{max}	6

5.2 Resultados Computacionais

Esta seção é dividida em duas subseções: Na primeira subseção (5.2.1) é apresentado o conjunto de instâncias reais e, para cada objetivo, os resultados são comparados com as melhores soluções encontradas na literatura. Na segunda subseção (5.2.2), é apresentada e disponibilizada a adaptação das novas instâncias para o problema WSFJSP. Para cada objetivo foi realizada uma análise comparando os resultados das duas meta-heurísticas propostas neste artigo. Para ambos os objetivos, as meta-heurísticas foram executadas 10 vezes por 60 minutos para cada uma das 20 instâncias.

5.2.1 Instâncias Reais

Retirado de um problema real e adaptado por KRESS, MÜLLER e NOSSACK (2019), o conjunto possui dez instâncias no total com nove trabalhadores fixos, que variam de 14 a 80 *jobs*, 11 a 16 máquinas e 34 a 188 operações. São apresentadas no formato r(J,M,W), onde J representa o número de *jobs*, M a quantidade de máquinas dispostas no problema e W o número de trabalhadores.

A Tabela 8 e a Tabela 9 apresentam, respectivamente, os resultados das instâncias para os critérios minimizar o makespan e miniminar o atraso total, os resultados são apresentado seguinte forma: A primeira coluna (r(J,M,W)) indica os nomes das instâncias utilizadas. As demais colunas apresentam, respectivamente para cada algoritmo avaliado, o melhor valor da função objetivo (FO) encontrado, seguido do tempo médio de execução para obter tal solução (t(s)), sendo que, para todos os casos foi aplicado o tempo limite de 3600 segundos para execução dos algoritmos. Os resultados dos algoritmos E-DM, H-DM e H-DMB foram obtidos em KRESS, MÜLLER e NOSSACK (2019). A sigla "tl" é a representação de que o algoritmo foi interrompido por atingir seu tempo limite de execução. Por fim, os resultados são comparados com a SA e a CS, meta-heurísticas implementadas neste trabalho. Por se tratar de algoritmos com metodologias diferentes, e para ter uma comparação mais justa, tanto a SA quanto a CS foram executadas 10 vezes por 3600 segundos para cada uma das 10, sendo a mesma estratégia utilizada por KRESS, MÜLLER e NOSSACK (2019), extraindo assim, a média dos melhores valores das melhores soluções

encontradas, juntamente com a média dos custos computacionais.

5.2.1.1 Resultados das Instâncias Reais (Makespan)

A partir dos resultados, considerando o critério minimizar makespan apresentado na Tabela 8, verifica-se que a SA e a CS foram superiores em oito soluções quando comparadas à heurística E-DM, em sete soluções quando comparados à heurística H-DM e em nove soluções quando comparados à heurística H-DMB. Ressalta-se que, tanto a SA quanto a CS alcançaram todos os melhores resultados da literatura, sendo que, obtiveram sete novas melhores soluções dentre as dez instâncias testadas. Observando o somatório dos valores das funções objetivo, pode-se perceber a eficácia e eficiência das meta-heurísticas propostas, uma vez que sempre obtiveram soluções melhores em um tempo inferior quando comparadas aos algoritmos propostos por KRESS, MÜLLER e NOSSACK (2019).

5.2.1.2 Resultados das Instâncias Reais (Atraso Total)

Similar performance das meta-heurísticas propostas foi obtida quando considerado o critério minimizar o atraso total, indicados na Tabela 9. A SA e a CS obtiveram sete soluções melhores que o método exato E-DM, quatro soluções superiores à heurística H-DM e cinco soluções superiores à heurística H-DMB. Ressalta-se que foram geradas três novas melhores soluções ainda não conhecidas na literatura para as instâncias mais complexas. Desta forma, pode-se constatar que a aplicação das meta-heurísticas desenvolvidas neste estudo proporcionou resultados mais eficientes para ambas funções objetivo consideradas, alcançando, em alguns casos, soluções de maior qualidades, ainda não encontradas.

Para analisar de forma mais ampla o desempenho das meta-heurísticas propostas neste estudo, são apresentadas nas Tabelas 10 e 11, os resultados respectivos, da minimização do makespan, e a minimização do atraso total. Nestas tabelas são apresentados o melhor valor de função objetivo encontrado (FO), a média das soluções encontradas (Avg.), o desvio padrão (DP) dos valores de função objetivo obtidos para cada instância (I), assim como o tempo médio de execução em segundos (t(s)) de cada algoritmo.

Analisando os resultados das Tabelas 10 e 11, constata-se que a performance foi semelhante para ambos os critérios de função objetivo considerados, sendo que não foram obtidos valores significativos que comprovassem a superioridade da SA sobre a CS ou vice-versa para este conjunto de instâncias. Ambas encontraram as mesmas soluções com zero de desvio padrão e com um tempo médio de execução relativo Tabela 11, o que comprova a eficiência e eficácia de ambas meta-heurísticas.

Tabela 8 – Resultados - instâncias reais considerando o objetivo *Makespan*.

r(J,M,W)	E-DM		H-DM		H-DMB		SA		CS	
	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)
r(14,13,9)	**6256**	tl	**6256**	14.3	6274	15.9	**6256**	0.3	**6256**	0.3
r(15,11,9)	5375.0	tl	5272	13.5	5392	17.5	**5089***	0.3	**5089***	0.4
r(16,14,9)	**7584**	tl	**7584**	18.7	**7584**	3	**7584**	0.0	**7584**	0.0
r(25,16,9)	8392.0	tl	**8180**	2190.4	8551	157.2	**8180**	0.7	**8180**	0.7
r(35,16,9)	11993.0	tl	9045	tl	9333	358.7	**8973***	1.2	**8973***	1.2
r(45,16,9)	25019.0	tl	12722	tl	12820	378.6	**12693***	1.5	**12693***	1.6
r(55,16,9)	29126.0	tl	15288	tl	14608	916.1	**14568***	2.0	**14568***	2.2
r(60,16,9)	33993.0	tl	18953	tl	17180	1444.4	**17148***	2.0	**17148***	2.2
r(70,16,9)	34716.0	tl	21202	tl	19590	1691.2	**19442***	2.5	**19442***	2.8
r(80,16,9)	-	-	23492	tl	21869	2694.4	**21535***	7.0	**21535***	4.5
Soma	-	-	127994	23836.9	123201	7677	121468	17.5	121468	15.9

*: Indica nova melhor solução conhecida.

Tabela 9 – Resultados - instâncias Reais considerando o objetivo Atraso Total.

	E-DM		H-DM		H-DMB		SA		CS	
r(J,M,W)	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)
r(14,13,9)	**0**	18.5	**0**	3.6	170	4.2	**0**	0.0	**0**	0.0
r(15,11,9)	**0**	6.8	**0**	0.4	**0**	0.4	**0**	0.0	**0**	0.0
r(16,14,9)	**864**	tl	**864**	25.2	**864**	5.2	**864**	0.0	**864**	0.0
r(25,16,9)	2150	tl	**0**	33.3	227	36.8	**0**	0.1	**0**	0.0
r(35,16,9)	19337	tl	**0**	50.9	**0**	18.8	**0**	0.1	**0**	0.0
r(45,16,9)	105019	tl	233	822.9	**0**	40.9	**0**	16.6	**0**	15.5
r(55,16,9)	–	–	**0**	1906.4	**0**	310	**0**	0.1	**0**	0.0
r(60,16,9)	327669	tl	947	3600	190	1300.8	**0***	0.1	**0***	0.1
r(70,16,9)	353140	tl	4675	3600	669	823.7	**0***	3.2	**0***	1.8
r(80,16,9)	–	–	10230	3600	4670	1152.5	**2253***	3.6	**2253***	4.9
Soma	-	-	16949	13642,7	6790	3693,3	3117	23.8	3117	22.3

*: Indica nova melhor solução conhecida.

Tabela 10 – Instâncias reais - Makespan (CS x SA).

	SA				CS			
I	FO	Avg.	DP	t(s)	FO	Avg.	DP	t(s)
r(14,13,9)	6256	6256	0	0.3	6256	6256	0	0.3
r(15,11,9)	5089	5089	0	0.3	5089	5089	0	0.4
r(16,14,9)	7584	7584	0	0.0	7584	7584	0	0.0
r(25,16,9)	8180	8180	0	0.7	8180	8180	0	0.7
r(35,16,9)	8973	8973	0	1.2	8973	8973	0	1.2
r(45,16,9)	12693	12693	0	1.5	12693	12693	0	1.6
r(55,16,9)	14568	14568	0	2.0	14568	14568	0	2.2
r(60,16,9)	17148	17148	0	2.0	17148	17148	0	2.2
r(70,16,9)	19442	19442	0	2.5	19442	19442	0	2.8
r(80,16,9)	21535	21535	0	7.0	21535	21535	0	4.5
Soma	121468	121468	0	17.5	121468	121468	0	15.9

Tabela 11 – Instâncias Real - Atraso total (CS x SA)

	SA				CS			
I	FO	Avg.	DP	t(s)	FO	Avg.	DP	t(s)
r(14,13,9)	0	0	0	0.0	0	0	0	0.0
r(15,11,9)	0	0	0	0.0	0	0	0	0.0
r(16,14,9)	864	864	0	0.0	864	864	0	0.0
r(25,16,9)	0	0	0	0.1	0	0	0	0.0
r(35,16,9)	0	0	0	0.1	0	0	0	0.0
r(45,16,9)	0	0	0	16.6	0	0	0	15.5
r(55,16,9)	0	0	0	0.1	0	0	0	0.0
r(60,16,9)	0	0	0	0.1	0	0	0	0.1
r(70,16,9)	0	0	0	3.2	0	0	0	1.8
r(80,16,9)	2253	2253	0	3.6	2253	2253	0	4.9
Soma	3117	3117	0	23.8	3117	3117	0	22.3

5.2.2 Comparação das Instâncias MK-WSFJSP

Propostas por BRANDIMARTE (1993), as instâncias MK01-MK10 foram criadas para o FJSP. A fim de transformá-las para o WSFJSP, neste trabalho, foram implementadas três importantes adaptações:

1. A primeira transformação consiste na limitação de trabalhadores operando as máquinas. Para isso, foram utilizados dois passos propostos por ZHENG e WANG (2016). O primeiro passo consiste em incluir trabalhadores às máquinas respectivas à sua habilidade operacional, conforme a Tabela 12. O segundo, considera que diferentes trabalhadores podem não ter a mesma eficiência temporal na realização de uma tarefa, ou seja, o tempo para realizar uma tarefa na mesma máquina pode possuir

valores distintos de acordo com sua habilidade.

$$P_{jiuk} = [P_{ji}, P_{ji} + \delta] \tag{34}$$

Onde P_{ji} é o tempo para a máquina processar a operação j do *job* i contida na instância original de BRANDIMARTE (1993), P_{ijuk} será o novo tempo de processamento do *job* j, operação i, processada pelo trabalhador k utilizando a máquina u e, por fim, δ é uma distribuição uniforme contida no intervalo [2,8].

2. A segunda transformação é a adição da data de entrega dos *jobs*. As mesmas foram adaptadas de forma similar as datas geradas por VILCOT e BILLAUT (2008).

3. A terceira adaptação foi a adição do tempo de *setup* nas máquinas. Este tempo foi gerado através de uma distribuição uniforme em um intervalo de [1, 5] conforme KRESS, MÜLLER e NOSSACK (2019).

O problema denominado WSFJSP-MK possui no total dez instâncias apresentadas em várias configurações desde: 11 a 21 *jobs*, 5 a 16 máquinas, 57 a 242 operações e 3 a 8 trabalhadores.

Com as novas instâncias geradas, foram examinados os desempenhos das meta-heurísticas SA e CS, analisando, primeiramente, o makespan na Tabela 13 e, posteriormente, o atraso total na Tabela 14, que são descritas a seguir. Para cada instância (I), lista-se o melhor valor de função objetivo encontrado (FO), a média das soluções encontradas (Avg.), o desvio padrão (DP) e o tempo médio de execução em segundos (t(s)) de cada algoritmo.

5.2.2.1 Comparação das Instâncias MK-WSFJSP (Makespan)

Considerando o critério makespan na Tabela 13, verifica-se que a CS encontrou as soluções em um tempo médio superior que a SA. Porém, suas soluções foram mais significativas, obtendo 8 de 10 soluções superiores contra 1 da SA.

5.2.2.2 Comparação das Instâncias MK-WSFJSP (Atraso Total)

Quando avaliado o critério atraso total na Tabela 14, novamente, a CS conseguiu resultados mais significativos, obtendo 6 soluções melhores que a SA, diminuindo o atraso da entrega em um total de 432. Porém, neste caso, com um custo computacional inferior. Com isso, pode-se constatar que a CS foi mais eficiente na exploração do espaço de soluções, conseguindo escapar de ótimos locais, ampliando a busca.

Tabela 12 – Associações novas instâncias Máquina-Trabalhador.

instâncias	W	M
MK1, MK2	4	$M_1 = \{W_1,W_3\}$, $M_2 = \{W_2,W_4\}$, $M_3 = \{W_1,W_4\}$, $M_4 = \{W_2,W_3,W_4\}$, $M_5 = \{W_1,W_2\}$, $M_6 = \{W_3\}$
MK3, MK4	6	$M_1 = \{W_1,W_3\}$, $M_2 = \{W_2,W_4\}$, $M_3 = \{W_4\}$, $M_4 = \{W_2,W_3\}$, $M_5 = \{W_1,W_6\}$, $M_6 = \{W_3,W_4,W_5\}$,$M_7 = \{W_4,W_5\}$, $M_8 = \{W_5,W_6\}$
MK5	3	$M_1 = \{W_1,W_3\}$, $M_2 = \{W_2,W_3\}$, $M_3 = \{W_1,W_3\}$, $M_4 = \{W_1,W_2\}$
MK6, MK10	8	$M_1 = \{W_1,W_8\}$, $M_2 = \{W_2,W_4\}$, $M_3 = \{W_3,W_4\}$, $M_4 = \{W_3,W_7\}$, $M_5 = \{W_6\}$, $M_6 = \{W_3,W_4,W_5\}$,$M_7 = \{W_2,W_5\}$, $M_8 = \{W_1,W_5,W_6\}$, $M_9 = \{W_4,W_7\}$, $M_{10} = \{W_1,W_6,W_8\}$,$M_{11} = \{W_2,W_3\}$, $M_{12} = \{W_4\}$, $M_{13} = \{W_4\}$,$M_{14} = \{W_7,W_8\}$, $M_{15} = \{W_5,W_7\}$,
MK7	4	$M_1 = \{W_1,W_4\}$, $M_2 = \{W_2,W_4\}$, $M_3 = \{W_1,W_3\}$, $M_4 = \{W_2,W_3\}$, $M_5 = \{W_1,W_4\}$
MK8, MK9	6	$M_1 = \{W_1,W_4\}$, $M_2 = \{W_2,W_6\}$, $M_3 = \{W_1,W_3\}$, $M_4 = \{W_2,W_3,W_6\}$, $M_5 = \{W_1,W_5\}$, $M_6 = \{W_5\}$,$M_7 = \{W_4,W_5\}$, $M_8 = \{W_3,W_6\}$,$M_9 = \{W_2,W_4\}$, $M_{10} = \{W_3,W_6\}$

Tabela 13 – Resultados - WSFJSP-MK Makespan.

	SA				CS			
I	FO	Avg.	DP	t(s)	FO	Avg.	DP	t(s)
WSFJSP-MK01	117	117.6	0.70	172.9	**116**	116.9	0.74	251.4
WSFJSP-MK03	289	292.2	2.15	315.9	**288**	290.6	2.17	315.9
WSFJSP-MK04	**143**	144.6	0.97	199.1	**143**	145.2	1.23	199.1
WSFJSP-MK05	386	390.7	2.58	144.3	**385**	388.6	2.07	355.7
WSFJSP-MK06	166	169.3	1.77	170.9	**165**	168.5	1.96	232.5
WSFJSP-MK07	298	299.9	1.79	198.8	**297**	299.8	1.81	298.8
WSFJSP-MK08	725	736.4	10.16	253.4	**722**	738.9	11.38	374.3
WSFJSP-MK09	**589**	597.4	3.92	296.3	590	596.2	3.43	446.7
WSFJSP-MK10	415	416.9	1.85	252.0	**410**	414.7	2.75	326.0
Soma	3230	3268.1	26.77	2155.5	3217	3261.6	28.46	3072.5

Tabela 14 – Resultados - WSFJSP-MK Atraso Total.

	SA				CS			
I	FO	Avg.	DP	t(s)	FO	Avg.	DP	t(s)
WSFJSP-MK01	**162**	170.4	4.33	2236.0	**162**	166	3.16	1784.2
WSFJSP-MK02	**0**	0	0	6.6	**0**	0	0	3.8
WSFJSP-MK03	**0**	0	0	9.3	**0**	0	0	5.6
WSFJSP-MK04	233	244.6	7.32	1774.2	**224**	236.8	8.28	1664.6
WSFJSP-MK05	**1164**	1222.5	28.89	2648.0	1181	1200.7	18.71	1554.8
WSFJSP-MK06	**0**	0	0	13.1	**0**	0	0	8.0
WSFJSP-MK07	677	706.8	14.46	988.4	**647**	676.7	19.54	917.9
WSFJSP-MK08	3783	3922	66.25	3027.4	**3617**	3777.5	105.12	2293.6
WSFJSP-MK09	2624	2702.4	55.69	695.7	**2404**	2525.7	63.71	993.6
WSFJSP-MK10	212	270.7	31.61	852.5	**188**	249.6	34.87	958.1
Soma	8855	9239.4	208.55	12251.2	8423	8833.0	253.39	10184.2

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente estudo apresentou duas meta-heurísticas, *Simulated Annealing* (SA) e *Clustering Search* (CS), para solucionar o *Worker Constrained Flexible Job Shop Scheduling Problem With Sequence-Dependent* (WSFJSP). Dois critérios foram analisados como função objetivo deste problema, considerando, separadamente, o tempo de término da última tarefa (makespan) e o somatório dos atrasos das entregas das tarefas (atraso total).

As meta-heurísticas foram comparadas com as instâncias proprostas por KRESS, MÜLLER e NOSSACK (2019) e, como resultado, obteve-se dez novas melhores soluções em um total de vinte instâncias avaliadas, sendo sete soluções considerando o critério makespan e três considerando o critério atraso total, com um custo computacional significativamente inferior. Além disso, foram geradas dez novas instâncias para analisar o desempenho individual da implementação das duas meta-heurísticas para os dois critérios. Os resultados computacionais evidenciaram a superioridade da CS nestas instâncias, que alcançou treze melhores soluções, sendo inferior em apenas uma, dentro de um conjunto de vinte testes. A seguir, são apresentadas algumas sugestões para estudos futuros:

- Estudar a possibilidade de desenvolver novos modelos matemáticos para o WSFJSP.
- Desenvolver modelos matemáticos, assim como, meta-heurísticas para resolver WSFJSP multiobjetivo.
- Generalizar o adiantamento dos *setups*, podendo assim, para cada sequência de operação, indicar se o *setup* pode ou não ser adiantado.

Referências

AKRAM, K.; KAMAL, K.; ZEB, A. Fast simulated annealing hybridized with quenching for solving job shop scheduling problem. *Applied Soft Computing*, v. 49, p. 510 – 523, 2016. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494616304318>. Citado na página 36.

ALLAHVERDI, A. et al. A survey of scheduling problems with setup times or costs. *European journal of operational research*, Elsevier, v. 187, n. 3, p. 985–1032, 2008. Citado na página 35.

ALTOÉ, W. A. S. et al. A clustering search metaheuristic for the bi-objective flexible job shop scheduling problem. In: *2018 XLIV Latin American Computer Conference (CLEI)*. [S.l.: s.n.], 2018. p. 158–166. ISSN null. Citado 2 vezes nas páginas 23 e 36.

ARENALES, M.; ARMENTANO, V.; OTHERS. *Pesquisa operacional*. [S.l.]: Elsevier Brasil, 2006. Citado na página 26.

ARMENTANO, V. A.; MAZZINI, R. A genetic algorithm for scheduling on a single machine with set-up times and due dates. *Production Planning & Control*, Taylor Francis, v. 11, n. 7, p. 713–720, 2000. Disponível em: <https://doi.org/10.1080/095372800432188>. Citado na página 35.

BISSOLI, D. C. et al. A simulated annealing metaheuristic for the bi-objective flexible job shop scheduling problem. In: *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. [S.l.: s.n.], 2018. p. 1–6. Citado 3 vezes nas páginas 23, 36 e 39.

BISSOLI, D. C.; ZUFFEREY, N.; AMARAL, A. R. S. Lexicographic optimization-based clustering search metaheuristic for the multiobjective flexible job shop scheduling problem. *International Transactions in Operational Research*, v. 28, n. 5, p. 2733–2758, 2021. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12745>. Citado 2 vezes nas páginas 35 e 41.

BRANDIMARTE, P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, v. 41, n. 3, p. 157–183, Sep 1993. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/BF02023073>. Citado 3 vezes nas páginas 47, 52 e 53.

BRUCKER, P.; SCHLIE, R. Job-shop scheduling with multi-purpose machines. *Computing*, Springer, v. 45, n. 4, p. 369–375, 1990. Citado na página 35.

CARVALHO, M. et al. Jcp, and soares, ja uma introdução sucinta a algoritmos de aproximação. *IMPA-Instituto de Matemática Pura e Aplicada*, 2001. Citado na página 22.

CHAUDHRY, I. A.; DRAKE, P. R. Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 42, n. 5-6, p. 581, 2009. Citado na página 35.

CHAUDHRY, I. A.; KHAN, A. A. A research survey: review of flexible job shop scheduling

techniques. *International Transactions in Operational Research*, v. 23, n. 3, p. 551–591, 2016. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/itor.12199>. Citado na página 35.

CRUZ-CHÁVEZ, M. A.; MARTÍNEZ-RANGEL, M. G.; CRUZ-ROSALES, M. H. Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research*, v. 24, n. 5, p. 1119–1137, 2017. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12195>. Citado na página 36.

DHIFLAOUI, M.; NOURI, H. E.; DRISS, O. B. Dual-resource constraints in classical and flexible job shop problems: A state-of-the-art review. *Procedia Computer Science*, Elsevier, v. 126, p. 1507–1515, 2018. Citado na página 35.

ECCLES, H. E. Logistics-what is it? *Naval Research Logistics Quarterly*, v. 1, n. 1, p. 5–15, 1954. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800010104>. Citado na página 21.

EREN, T. A multicriteria scheduling with sequence-dependent setup times. *Applied Mathematical Sciences*, v. 1, n. 58, p. 2883–2894, 2007. Citado na página 27.

FUCHIGAMI, H. et al. Modelos matemáticos para programação de job shop com tempos de setup independentes da sequência. *Produção Online*, v. 17, p. 245–267, 03 2017. Citado na página 28.

HANSEN, P.; MLADENOVIĆ, N. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, Elsevier, v. 154, n. 5, p. 802–817, 2006. Citado na página 45.

HO, N. B.; TAY, J. C. Genace: an efficient cultural algorithm for solving the flexible job-shop problem. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. [S.l.: s.n.], 2004. v. 2, p. 1759–1766 Vol.2. Citado na página 26.

HYATT, C.; WEAVER, P. A brief history of scheduling. *Melbourne, Australia: Mosaic Project Services Pty Ltd*, 2006. Citado na página 21.

JAIN, A.; MEERAN, S. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, v. 113, n. 2, p. 390 – 434, 1999. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221798001131>. Citado na página 26.

KACEM, I.; HAMMADI, S.; BORNE, P. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, v. 60, n. 3, p. 245 – 276, 2002. ISSN 0378-4754. Intelligent Forecasting, Fault Diagnosis, Scheduling, and Control. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378475402000198>. Citado na página 22.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. Citado na página 42.

KRESS, D.; MÜLLER, D.; NOSSACK, J. A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. *OR Spectrum: Quantitative Approaches in Management*, v. 41, n. 1, p. 179–217, March 2019. Disponível em: <https://ideas.repec.org/a/spr/orspec/v41y2019i1d10.1007_s00291-018-0537-z.html>. Citado 16 vezes nas páginas 13, 22, 23, 26, 28, 29, 30, 32, 35, 36, 37, 47, 48, 49, 53 e 57.

LEE, Y. H.; BHASKARAN, K.; PINEDO, M. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, Taylor Francis, v. 29, n. 1, p. 45–52, 1997. Disponível em: <https://doi.org/10.1080/07408179708966311>. Citado na página 35.

LI, J. et al. A hybrid algorithm for scheduling of dual-resource constrained job shop. In: *2010 International Conference on Computing, Control and Industrial Engineering*. [S.l.: s.n.], 2010. v. 1, p. 235–238. Citado na página 36.

MASTROLILLI, M.; GAMBARDELLA, L. M. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, v. 3, n. 1, p. 3–20, 2000. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1425%28200001/02%293%3A1%3C3%3A%3AAID-JOS32%3E3.0.CO%3B2-Y>. Citado na página 36.

MENG, L. et al. Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access*, v. 7, p. 68043–68059, 2019. ISSN 2169-3536. Citado na página 22.

OCHOA, G.; VEREL, S.; TOMASSINI, M. First-improvement vs. best-improvement local optima networks of nk landscapes. In: SPRINGER. *International Conference on Parallel Problem Solving from Nature*. [S.l.], 2010. p. 104–113. Citado na página 45.

OLIVEIRA, A. C. M. D.; CHAVES, A. A.; LORENA, L. A. N. Clustering search. *Pesquisa Operacional*, scielo, v. 33, p. 105 – 121, 04 2013. ISSN 0101-7438. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382013000100007&nrm=iso>. Citado 2 vezes nas páginas 43 e 44.

RABADI, G.; MOLLAGHASEMI, M.; ANAGNOSTOPOULOS, G. C. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, Elsevier, v. 31, n. 10, p. 1727–1751, 2004. Citado na página 35.

RABELLO, R. L. et al. A clustering search metaheuristic for the point-feature cartographic label placement problem. *European Journal of Operational Research*, v. 234, n. 3, p. 802 – 808, 2014. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221713008448>. Citado na página 47.

RIBEIRO, G. M.; LAPORTE, G.; MAURI, G. R. A comparison of three metaheuristics for the workover rig routing problem. *European Journal of Operational Research*, v. 220, n. 1, p. 28 – 36, 2012. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221712000665>. Citado na página 47.

RODAMMER, F. A.; WHITE, K. P. A recent survey of production scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 18, n. 6, p. 841–851, Nov 1988. ISSN 0018-9472. Citado na página 21.

SHARMA, P.; JAIN, A. A review on job shop scheduling with setup times. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 01 2015. Citado na página 27.

SIERKSMA, G. *Linear and integer programming: theory and practice*. [S.l.]: CRC Press, 2001. Citado na página 22.

SINGH, M. R.; MAHAPATRA, S. A quantum behaved particle swarm optimization for flexible job shop scheduling. *Computers Industrial Engineering*, v. 93, p. 36 – 44, 2016. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S036083521500474X>. Citado na página 21.

SUN, Z.; ZHU, J. Intelligent optimization for job shop scheduling of dual-resources. *Journal of Southeast University (Natural Science Edition)*, v. 35, p. 376–381, 05 2005. Citado na página 28.

TAN, K.-C.; NARASIMHAN, R. Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega*, Elsevier, v. 25, n. 6, p. 619–634, 1997. Citado na página 35.

VILCOT, G.; BILLAUT, J.-C. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, v. 190, n. 2, p. 398 – 411, 2008. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221707006327>. Citado na página 53.

WANG, W. et al. A hybrid particle swarm optimisation for multi-objective flexible job-shop scheduling problem with dual-resources constrained. *International Journal of Computing Science and Mathematics*, v. 8, p. 526, 01 2017. Citado na página 35.

WIROJANAGUD, P. et al. Modelling inherent worker differences for workforce planning. *International journal of production research*, Taylor & Francis, v. 45, n. 3, p. 525–553, 2007. Citado na página 35.

YAZDANI, M. et al. Two meta-heuristic algorithms for the dual-resource constrained flexible job-shop scheduling problem. *Scientia Iranica. Transaction E, Industrial Engineering*, Sharif University of Technology, v. 22, n. 3, p. 1242, 2015. Citado na página 36.

YU, J. M.; LEE, D. Solution algorithms to minimise the total family tardiness for job shop scheduling with job families. *European J. of Industrial Engineering*, v. 12, p. 1, 01 2018. Citado na página 21.

ZHENG, X.-L.; WANG, L. A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem. *International Journal of Production Research*, Taylor & Francis, v. 54, n. 18, p. 5554–5566, 2016. Citado na página 52.

ZHU, X.; WILHELM, W. E. Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, Taylor Francis, v. 38, n. 11, p. 987–1007, 2006. Disponível em: <https://doi.org/10.1080/07408170600559706>. Citado na página 27.