UNIVERSIDADE FEDERAL DE OURO PRETO

Programa de Pós-Graduação em Engenharia de Produção

ABORDAGENS SIMHEURÍSTICAS PARA O PROBLEMA DE *FLOW SHOP* PERMUTACIONAL MULTIOBJETIVO

Naiara Helena Vieira

Orientador: Prof. Dr. Aloísio de Castro Gomes Júnior

Coorientador: Prof. Dr. Helton Cristiano Gomes

OURO PRETO – MG

2022

Naiara Helena Vieira

ABORDAGENS SIMHEURÍSTICAS PARA O PROBLEMA DE *FLOW SHOP* PERMUTACIONAL MULTIOBJETIVO

Dissertação apresentada ao Programa de pós-graduação em Engenharia de Produção da Universidade Federal de Ouro Preto como requisito para obtenção do título de Mestre em Engenharia de Produção.

OURO PRETO – MG

2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

V657a Vieira, Naiara Helena.
Abordagens simheurísticas para o problema de flow shop permutacional multiobjetivo. [manuscrito] / Naiara Helena Vieira. - 2022.
88 f.: il.: color., tab..

Orientador: Prof. Dr. Aloísio de Castro Gomes Júnior.
Coorientador: Prof. Dr. Helton Cristiano Gomes.
Dissertação (Mestrado Acadêmico). Universidade Federal de Ouro Preto. Departamento de Engenharia de Produção. Programa de Pós-Graduação em Engenharia de Produção.

1. Flow shop. 2. Simheurística. 3. Otimização multiobjetivo. I. Gomes, Helton Cristiano. II. Gomes Júnior, Aloísio de Castro. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 658.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
PROGRAMA DE POS-GRADUACAO EM ENGENHARIA DE PRODUCAO

FOLHA DE APROVAÇÃO

Naiara Helena Vieira

Abordagens Simheurísticas para o problema de *flow shop* permutacional multiobjetivo

Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Mestre em Engenharia de Produção

Aprovada em 17 de maio de 2022

Membros da banca

[Doutor] - Aloisio de Castro Gomes Júnior - Orientador (Universidade Federal de Ouro Preto)
[Doutor] - Helton Cristiano Gomes - co-orientador(Universidade Federal de Ouro Preto)
[Doutor] - Irce Fernandes Gomes Guimarães - (Universidade Federal de Ouro Preto)
[Doutor] - João Flávio de Freitas Almeida - (Universidade Federal de Minas Gerais)

Aloisio de Castro Gomes Júnior, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 17/05/2022

Documento assinado eletronicamente por **Aloisio de Castro Gomes Junior**, **COORDENADOR(A) DE CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO**, em 26/05/2022, às 14:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0333413** e o código CRC **DCC87874**.

Referência: Caso responda este documento, indicar expressamente o Processo nº 23109.006827/2022-38 SEI nº 0333413

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: (31)3852-8709 - www.ufop.br

AGRADECIMENTOS

Primeiramente, agradeço à Deus pelo dom da vida.

Aos meus professores Aloísio e Helton pela orientação, ensinamentos e dedicação, empreendidos para o desenvolvimento desta pesquisa.

A meu noivo, Luiz Antônio, meus pais, Gilmara e Antônio, e meus irmãos, Naiane, Maicon e Natália, por todo apoio, compreensão, incentivo e carinho.

Aos meus colegas do PPGEP que direta ou indiretamente colaboram para a realização desta pesquisa.

Aos membros da banca examinadora Prof.ª Dra. Irce Fernandes Gomes Guimarães e Prof. Dr. João Flávio de Freitas Almeida, e ao Prof. Dr. Marcone Jamilson Freitas Souza pelas contribuições apontadas para aprimoramento desta pesquisa.

À UFOP pela oportunidade e por propiciar as condições necessárias, inclusive auxílio financeiro, para o desenvolvimento desta pesquisa. E a CAPES por consolidar cada vez mais os cursos de pós-graduação.

RESUMO

O problema *flow shop* permutacional multiobjetivo (PFSP-MO) consiste no processamento de um conjunto de *jobs* em todas as máquinas do ambiente produtivo, de tal forma que dois ou mais objetivos sejam otimizados. Nesta pesquisa, o PFSP-MO considera parâmetros estocásticos, responsáveis por representar possíveis atrasos ocorridos durante o processamento, e possui como objetivo minimizar o atraso total, o *makespan* e a antecipação total esperados. Por se tratar de um problema de otimização multiobjetivo, os métodos aplicados para resolver o PFSP-MO não retornam apenas uma solução, mas sim um conjunto de soluções, as quais são avaliadas pelo conceito ótimo de Pareto. Inicialmente, o método ponderação dos objetivos foi aplicado ao PFSP-MO com dados determinísticos e executado em um conjunto de 80 instâncias adaptadas da literatura. Em seguida, foram propostas duas abordagens simheurísticas: *Variable Neighborhood Search* Multiobjetivo (MOVNS) e Pareto *Iterated Local Search* (PILS). Nas quais: a solução inicial é gerada pelo método *Greedy Randomized Adaptive Seach Procedure* e a busca local utiliza três estruturas de vizinhança: inserção, troca e *three point move.* Tanto no MOVNS quanto no PILS a simulação é aplicada de duas maneiras distintas. Primeiramente, uma simulação rápida é aplicada a todas as soluções pertencentes à Fronteira Pareto (FP). E posteriormente, uma simulação intensiva é aplicada nas soluções da FP elite. Os métodos propostos foram executados em 120 instâncias adaptadas da literatura. Os resultados encontrados pelos métodos foram comparados entre si pelas métricas de avaliação desempenho: número de soluções na FP, medida de cardinalidade, medida de distância média e máxima, taxa de erro e diferença de hipervolume. Por fim, os resultados encontrados pelas métricas foram estatisticamente avaliados pelo teste *t* a fim de comprovar a existência de diferença estatisticamente significativa entre os métodos em relação às métricas de avaliação de desempenho. Tanto o resultado das métricas de avaliação quanto da análise estatística apontam que o MOVNS se apresenta superior ao PILS.

Palavras-chave: *Flow shop*, simheurística, otimização multiobjetivo.

ABSTRACT

The multi-objective permutational flow shop problem (MOPFSP) consists in processing a set of jobs in all machines of the production environment, in such a way that two or more objectives are optimized. In this research, MOPFSP considers stochastic parameters, responsible for representing possible delays that occurred during processing, and its objective is to minimize the expected total tardiness, makespan, and total earliness. Since it is a multi-objective optimization problem, the methods applied to solve MOPFSP do not return only one solution, but a set of solutions, which are evaluated by the Pareto optimal concept. Initially, the weighted-sum method was applied to MOPFSP with deterministic values and run on a set of 80 instances adapted from the literature. Next, two simheuristic approaches were proposed: Multi-objective Variable Neighborhood Search (MOVNS) and Pareto Iterated Local Search (PILS). Which: the initial solution is generated by the Greedy Randomized Adaptive Search Procedure method and the local search uses three neighborhood structures: insertion, exchange, and three-point move. In both MOVNS and PILS the simulation is applied in two different ways. First, a fast simulation is applied to all solutions belonging to the Pareto Frontier (PF). And subsequently, an intensive simulation is applied to the solutions of the elite PF. The proposed methods were run on 120 instances adapted from the literature. The results found by the methods were compared with each other by the performance evaluation metrics: a number of solutions on the PF, cardinality measure, average and maximum distance measure, error rate, and hypervolume difference. Finally, the results found by the metrics were statistically evaluated by the t-test to prove the existence of a statistically significant difference between the methods of the performance evaluation metrics. Both the results of the evaluation metrics and the statistical analysis indicate that MOVNS is superior to PILS.

Keywords: Flow shop, simheuristic, multi-objective optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O ambiente industrial é composto por diferentes setores, os quais atuam mutuamente a fim de oferecer ao cliente final determinado produto ou serviço. No entanto, para que este produto seja entregue ao consumidor dentro do prazo estabelecido e com custo compensatório, é necessária uma adequada programação da produção. Ao dispor de uma programação da produção eficiente, a indústria pode otimizar diversos objetivos como custos de produção, atraso, tempo de processamento total, atendimento da data de entrega, entre outros (LUSTOSA *et al.*, 2008; SU *et al.*, 2016).

Segundo Pinedo (2016), o problema de programação da produção consiste na alocação de um conjunto de *jobs* (ordens de produção, operações ou tarefas) à um conjunto limitado de máquinas, no sequenciamento desses *jobs* nas máquinas e na determinação do instante de início e término do processamento dos *jobs*, procurando otimizar algum critério específico. Já o problema de sequenciamento compreende apenas em determinar a sequência de processamento ideal dos *jobs* nas respectivas máquinas com o intuito de otimizar um ou mais objetivos (GONZÁLEZ-NEIRA *et al.*, 2017a). Pinedo (2016) descreve *job* como uma sequência de determinado número de tarefas ou operações necessárias para a fabricação do produto.

Os problemas de sequenciamento consideram então que um conjunto de *jobs* que deve ser processado em um conjunto de máquinas. Sendo que, a quantidade e a forma de disposição destas máquinas no ambiente de produção é o que determina a configuração do problema. As principais configurações de problemas de sequenciamento, de acordo com Pinedo (2016), são: (*i*) máquina única, apenas uma máquina para processar todos os *jobs*; (*ii*) máquinas paralelas idênticas, os *jobs* podem ser processados em qualquer uma das máquinas pois todas são iguais; (*iii*) máquinas paralelas com diferentes velocidades, as máquinas executam o mesmo tipo de processamento com velocidades diferentes, assim os *jobs* são processados com celeridades distintas; (*iv*) *job shop*, neste caso os *jobs* a serem executados não necessariamente precisam ser processados em todas máquinas do ambiente, e a ordem de processamento nas máquinas não precisa ser a mesma; e (*v*) *flow shop*, nesta configuração os *jobs* devem ser processados em todas as máquinas do ambiente de produção, e assumindo que a ordem de processamento dos *jobs* deve ser a mesma tem-se o problema *flow shop* permutacional.

Diversos problemas de sequenciamento têm sido estudados nos últimos anos, entre eles o problema *flow shop* permutacional (PFSP). O PFSP pode ser facilmente percebido em diversos cenários industriais, onde um conjunto de *jobs* deve ser processado na mesma

sequência em todas as máquinas (GONZÁLEZ-NEIRA *et al.*, 2017a). Os principais objetivos de otimização associados à problemas de sequenciamento, incluindo o PFSP, são: instante de conclusão do último *job* na última máquina (*makespan*), tempo de fluxo, atraso total e antecipação (ARENALES *et al*., 2015). Os objetivos de otimização estão principalmente relacionados ao tempo e também aos custos. Diante da necessidade de otimizar os processos, seja minimizando tempos de produção, gargalos, custos e antecipação e diversos critérios, o PFSP pode ser classificado como: (*i*) mono-objetivo ou multiobjetivo e (*ii*) como determinístico ou estocástico.

Na maioria das literaturas disponíveis, o PFPS é abordado na versão mono-objetivo, ou seja, com apenas um objetivo a ser otimizado, como, por exemplo, em Vallada *et al*. (2015), Ta *et al*. (2018) e Pan *et al*. (2019), os quais buscam a minimização do *makespan*, atraso total e tempo de fluxo total, respectivamente. No entanto, o PFSP pode envolver diversos objetivos e estes podem ser considerados e otimizados simultaneamente. Com essa característica o PFSP é tratado como multiobjetivo. Arroyo e Pereira (2011), Xu *et al*. (2017) e Tagestiren *et al*. (2021), por exemplo, abordam o PFSP multiobjetivo.

Ao buscar a otimização do PFSP outro fator deve ser considerado, a incerteza. Atrasos inesperados podem ocorrer e quebras de máquinas são recorrentes. Sendo assim, parâmetros estocásticos devem ser aplicados a fim de buscar resultados mais realistas. A introdução de fatores, dados, incertos no problema ocorre quando os parâmetros assumem valores aleatórios não-negativos definidos por distribuições de probabilidade, por exemplo (GONZÁLEZ-NEIRA *et al*., 2017b).

Este trabalho propõe então a solução de uma variante do PFS, ainda pouco discutida, o PFSP-MO estocástico. Este estudo busca determinar um conjunto de sequenciamentos de *jobs* para o PFSP-MO estocástico de modo que o atraso total, o *makespan* e a antecipação esperados sejam minimizados. Vale ressaltar que os objetivos de minimização, atraso e antecipação, são conflitantes entre si, haja visto de que a melhora de um objetivo pode deteriorar o outro; e visam promover a entrega dos produtos (*jobs*) na data desejada, reduzindo assim o estoque de produtos acabados e consequentemente o custo de manutenção deste estoque. Permitindo assim, que o processo produtivo opere de acordo com o conceito de produção *just-in-time* (JIT). A filosofia JIT é proposta inicialmente pela Toyota Motor Company preza pelo processamento dos produtos no momento certo e na quantidade certa, reduzindo assim os estoques e melhorando o processo produtivo (RAMESH e DICKERSON, 2022).

Com natureza de otimização combinatória multiobjetivo e assumindo parâmetros estocásticos, o PFSP-MO proposto pode ser solucionado aplicando-se métodos eficientes que combinam simulação e otimização. Juan *et al.* (2014) afirmam que a hibridização entre simulação e métodos de otimização, especificamente as meta-heurísticas, constituem uma abordagem promissora: simheurística. As quais são capazes de solucionar problemas estocásticos multiobjetivos. González-Neira *et al.* (2019a) e González-Neira *et al.* (2019b), por exemplo, propõem abordagens simheurísticas para solução do PFSP-MO.

Para solucionar o problema em estudo, propôs-se duas abordagens simheurísticas distintas. A primeira abordagem desenvolvida é resultado da hibridação entre simulação e as meta-heurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Variable Neighborhood Search* (VNS). Já a segunda abordagem combina simulação e as meta-heurísticas GRASP e *Iterated Local Search* (ILS). Em ambas as abordagens usou-se o método de geração da solução inicial da meta-heurística GRASP para geração das soluções iniciais e as meta-heurísticas VNS e ILS são responsáveis pelo refino destas soluções.

1.1 Justificativa

A busca pela otimização de processos produtivos, seja minimizando ou maximizando critérios, é apurada em diferentes setores econômicos. No entanto, a otimização do processo produtivo se relaciona diretamente com a programação da produção. Diante dos objetivos a serem otimizados e das restrições a serem respeitadas, a programação da produção atua a fim de oferecer um conjunto de soluções que atenda ao processo produtivo e às necessidades da indústria.

Um fator importante a ser considerado, e ainda pouco tratado na literatura, é a presença de incertezas no processo. A representação de incertezas, adotando dados estocásticos, enriquece a análise do problema. Nota-se que problemas como este propõem uma representação mais concreta de processos reais.

Em geral, os processos produtivos contam com uma considerável quantidade de *jobs* e de máquinas, o que torna dificultoso e complexo o sequenciamento de *jobs* a fim de otimizar algum critério. Para que um processo produtivo seja eficiente e otimizado, é necessário que a programação da produção seja determinada em tempo hábil, com resultados que atendam às necessidades do processo e que busque operar em *just-in-time*.

Diante deste cenário multiobjetivo e incerto que o PFSP-MO aqui abordado está inserido. Um cenário o qual necessita de resposta rápida com relação ao sequenciamento ideal de *jobs* para aquele momento, que atenda aos critérios definidos e que satisfaça às restrições do processo.

Sendo assim, o desenvolvimento de algoritmos dedicados a solucionar o PFSP-MO estocástico e que sejam capazes de aplicar ferramentas de otimização próprias e adequadas torna-se necessário. Ao desenvolver um algoritmo nestes moldes é possível identificar o conjunto de melhores soluções para tal e com um tempo de resposta aceitável.

Apesar da grande aplicabilidade do PFSP-MO, o problema ainda não tem sido devidamente explorado na literatura, tão pouco possui dados, métodos e parâmetros para fins de análise de resultados. Por este motivo, o desenvolvimento de metodologias que atendem às demandas de processos que operam sob configuração do problema *flow shop* permutacional multiobjetivo, apresentará contribuições tanto para o contexto literário quanto para solução de problemas reais.

1.2 Objetivos

O objetivo desse trabalho é desenvolver simheurísticas para resolver o PFSP-MO, utilizando parâmetros estocásticos. A resolução do PFSP-MO busca gerar um conjunto de soluções não dominadas de tal forma que os valores esperados para o atraso total, *makespan* e antecipação total sejam minimizados.

Para alcançar tal objetivo é necessário estabelecer os objetivos específicos que também devem ser atingidos, sendo eles:

(a) Realizar um estudo bibliográfico a fim de apontar métodos capazes de solucionar o PFSP-MO estocástico;
(b) Adaptar e implementar um modelo matemático para o PFSP-MO em um *software* de otimização, baseando-se no método exato de ponderação dos objetivos e utilizando dados determinísticos;
(c) Desenvolver algoritmos em linguagem Python, baseado em abordagens simheurísticas capazes de resolver o problema PFSP-MO estocástico.

1.3 Estrutura do trabalho

Este estudo é organizado da seguinte forma:

O Capítulo 2 apresenta os conceitos relativos à otimização multiobjetivo, problemas, métodos de solução e descreve ainda o conceito geral de simheurística. O problema abordado neste estudo e trabalhos relevantes sobre o tema também são descritos.

No Capítulo 3 é apresentado o método de ponderação dos objetivos aplicado ao PFSP-MO. É descrito o modelo matemático que representa o problema, as instâncias-teste utilizadas e os resultados obtidos.

No Capítulo 4 são detalhadas as abordagens simheurísticas propostas bem como os resultados apurados. Para tanto, todas as etapas e especificações do MOVNS e PILS são caracterizadas. Em seguida, é realizada uma análise estatística dos resultados.

Por fim, no Capítulo 5, as conclusões deste trabalho e projeções futuras são apresentadas.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentado uma revisão da literatura acerca de problemas de otimização multiobjetivos, bem como métodos clássicos e métodos heurísticos e meta-heurísticos que podem ser aplicados a fim de encontrar um conjunto de soluções não dominadas, ou seja, um conjunto formado pelas melhores soluções para o problema. Conceitos sobre simulação e otimização também são apresentados, bem como, programação da produção e alguns dos principais problemas de sequenciamento de tarefas presentes em ambientes produtivos. É apresentado de forma mais detalhada o problema *flow shop* permutacional, a modelagem matemática para esse problema e métodos de solução para problemas similares ao proposto neste trabalho.

2.1 Problemas de otimização multiobjetivo

Otimização consiste em encontrar uma solução ou um conjunto de soluções ótimas para determinado(s) objetivo(s). Diversos problemas envolvem vários objetivos que devem ser considerados simultaneamente, havendo ainda a possibilidade de estes objetivos serem conflitantes, ou seja, a melhoria em um objetivo provoca a deterioração em outro. Problemas com esta característica são chamados problemas de otimização multiobjetivo (MIETTINEN, 2008).

Os problemas de otimização multiobjetivo envolvem várias funções objetivo, podendo estas serem de maximização e/ou minimização e também estão sujeitos a um conjunto de restrições que devem ser satisfeitas, da mesma forma que os problemas mono-objetivo. Miettinen (2008) afirma que além da natureza conflituosa dos objetivos, outra situação que pode surgir na otimização multiobjetivo diz respeito à incomensurabilidade entre as funções objetivo, ou seja, os objetivos podem não possuir a mesma unidade de medida. Neste caso é necessário tornar os objetivos adimensionais.

Em problemas multiobjetivo busca-se não apenas uma única solução, mas sim um conjunto de soluções eficientes que otimizem de forma simultânea os objetivos a serem atingidos. Tomando como exemplo o PFS, a partir deste conjunto de soluções, o tomador de decisões apontará o sequenciamento de jobs a ser aplicado (MIETTINEN, 2008).

Um problema de otimização multiobjetivo possui determinado número de funções objetivo as quais devem ser minimizadas e/ou maximizadas e um conjunto de restrições que

devem ser satisfeitas por todo o conjunto de soluções. Deb (2008) apresenta de forma geral o problema de otimização multiobjetivo:

$$Otimizar\ \ f(x) = \big(f(x_1), f(x_2), \dots, f(x_n)\big) \tag{2.1}$$

$$\text{Sujeito a:}\quad g(x) = \Big(g_1(x), g_2(x), \dots, g_j(x)\Big) \leq 0 \tag{2.2}$$

$$h(x) = \big(h_1(x), h_2(x), \dots, h_k(x)\big) = 0 \tag{2.3}$$

$$x_i^{(L)} \leq x_i \leq\ x_i^{(U)}, i = 1, \dots, v \tag{2.4}$$

A solução $x \in \mathbb{R}^v$ é um vetor com v variáveis de decisão $x = (x_1, x_2, \dots, x_v)^T$, sendo este uma solução do problema. As desigualdades (2.2) e (2.3) caracterizam as restrições do problema. Os valores $x_i^{(L)}$ e $x_i^{(U)}$ representam os limites inferiores e superiores para cada variável de decisão x_i, constituindo assim o conjunto, ou espaço, de variáveis de decisão. Uma solução factível deve atender as $j + k$ restrições e aos $2v$ limites das variáveis de decisão. Todas as soluções factíveis constituem a região factível F_x. As funções objetivo constituem um espaço multidimensional chamado de espaço objetivo $Z \subset \mathbb{R}^n$ (DEB, 2008). O autor acrescenta que para cada solução x no espaço de variáveis de decisão há um ponto $z \in \mathbb{R}^n$ no espaço objetivo, denotado por $f(x) = z = (z_1, z_2, \dots, z_n)^T$.

A resolução de problemas de otimização multiobjetivo não se resume na obtenção de uma única solução que otimize todos os critérios, principalmente se forem de natureza conflituosa. As soluções geradas pelos métodos de resolução utilizados são avaliadas utilizando o critério de dominância de Pareto (NOURI; LADHARI, 2018).

A determinação do conjunto de soluções Pareto-ótimas, ou soluções não dominadas, é embasado no conceito de dominância de Pareto. A dominância de Pareto compara duas soluções e verifica se estas são dominadas ou não entre si. As soluções dominadas são descartadas e, as não dominadas são armazenadas no conjunto de soluções Pareto-ótimas. O conceito de dominância de Pareto é comumente utilizado em problemas multiobjetivos quando a comparação entre soluções é necessária (NOURI; LADHARI, 2018).

O conceito de dominância de Pareto consiste na comparação entre duas soluções distintas, ou seja, dois vetores de solução. Xu *et al.* (2017) tomam como exemplo um problema de minimização tendo os vetores solução *n*-dimensionais $X = [x_1, \mathrm{x}_2, \dots, x_n]^T$ e $X' =$

$[x'_1, x'_2, \ldots, x'_n]^T$, e $f_i(X)$ e $f_i(X')$ os valores das funções objetivo de X e X', respectivamente. Daí, tem-se as definições (ARROYO; PEREIRA, 2011):

Definição 2.1: X domina X' se e somente se $f_i(X) \leq f_i(X')$ para todo objetivo i, havendo ao menos um objetivo i com $f_i(X) < f_i(X')$.

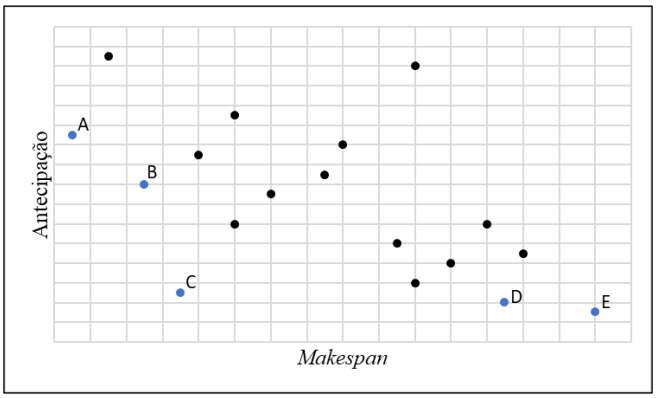
Definição 2.2: O vetor X^* é inserido no conjunto Pareto-ótimo se e somente se X^* não é dominado por qualquer outro vetor solução.

Definição 2.3: X e X' são ditas não dominadas entre si, ou indiferentes, se e somente se $f_i(X) < f_i(X')$ para algum objetivo i, e $f_i(X) > f_i(X')$ para outro objetivo i.

Ressalta-se que todo vetor solução do conjunto Pareto-ótimo possui um vetor objetivo correspondente que forma a Fronteira de Pareto (FP) (XU *et al.*, 2017).

A Figura 2.1 ilustra o exposto, onde é apresentado um exemplo fictício do PFSP-MO cujos objetivos conflituosos são a minimização do *makespan* e da antecipação. As 18 soluções encontradas para este problema estão representadas na Figura 2.1. A partir destas soluções encontradas é verificada a formação da Fronteira Dominante, ou FP, composta pelas melhores soluções para o problema. Neste caso específico, a FP é formada por um conjunto de cinco soluções, representadas pelos pontos A, B, C, D e E. Estas soluções são não dominadas entre si e dominam todas as demais soluções do espaço de soluções.

Figura 2.1: Conceito de dominância



Fonte: Elaborado pela autora (2022)

2.2 Métodos de resolução para otimização multiobjetivo

O resultado obtido em um problema multiobjetivo é um conjunto de soluções não dominadas, ou seja, não comparáveis entre si. Dessa forma, a escolha efetiva de uma única

solução a ser aplicada é responsabilidade do tomador de decisões, sendo esta escolha feita a partir de critérios por ele estabelecidos.

Sendo assim, é verificado a existência de duas etapas distintas na solução de problemas multiobjetivos: a busca de soluções e a tomada de decisão (HORN, 1997). A busca por soluções refere-se ao processo de otimização, onde métodos e conceitos são aplicados a fim de constituir um conjunto de soluções factíveis que compõem o conjunto Pareto-ótimo. Os métodos de solução podem ser divididos em clássicos e heurísticos, e são apresentados, respectivamente, nas Seções 2.2.1 e 2.2.2.

Já a etapa de tomada de decisão trata dos critérios para seleção de uma solução pertencente ao conjunto Pareto-ótimo. O critério de seleção aplicado pelo tomador de decisões irá refletir suas preferências no momento da decisão, uma vez que ele pode ponderar entre as soluções não dominadas entre si. De acordo com o processo de tomada de decisão, os métodos de resolução de problemas de otimização multiobjetivo podem ser classificados em (HORN, 1997; SUN *et al.*, 2011):

(a) *Métodos a priori:* métodos mais simples que permitem ao tomador de decisão intervir antes da resolução do problema. Nestes métodos, o tomador de decisões poderá conferir preferências aos objetivos.

   As preferências do tomador de decisões podem ser indicadas a partir da aplicação do método de ponderação dos objetivos, apresentado na próxima seção, o qual atribui pesos distintos a cada uma das funções objetivo. A aplicação deste método permite que o problema multiobjetivo seja solucionado por abordagens tradicionais de otimização mono-objetivo.

(b) *Métodos a posteriori:* o processo de tomada de decisão é efetuado após a geração do conjunto Pareto-ótimo. Nestes métodos, todas as funções objetivo do problema possuem a mesma relevância, sendo dispensadas as informações de preferência do tomador de decisões.

   A partir do conjunto Pareto-ótimo, o tomador de decisão fará a seleção de uma única solução que julgar mais adequada às suas necessidades. Os métodos *a posteriori* possuem, na literatura, mais aplicações na área de engenharia e em problemas de programação da produção quando comparado ao método *a priori*.

Estes métodos são, certamente, indicados para problemas onde há constantes mudanças de preferências, uma vez que novas execuções não serão necessárias pois tem-se as soluções Pareto-ótimas. No entanto, a principal desvantagem é o alto custo computacional, haja vista que deve-se encontrar o conjunto de todas as soluções Pareto-ótimas para o problema.

(c) *Métodos iterativos:* enfatizam principalmente o processo iterativo entre a busca de soluções e a tomada de decisão. Inicialmente, os tomadores de decisão manifestam suas preferências. No decorrer da otimização, especificamente antes de cada iteração, os tomadores de decisão podem definir novas prioridades, a fim de buscar boas soluções.

De forma simplificada, estes métodos consistem em encontrar, inicialmente, uma solução não dominada. Em seguida, é aplicada a preferência do tomador de decisões sobre esta solução não dominada, guiando assim a busca por novas soluções. Este processo é repetido até que o tomador de decisão esteja satisfeito ou nenhuma melhoria adicional é alcançada. Poucos trabalhos o aplicam estes métodos devido à sua complexidade e difícil teoria. Outro fator que dificulta ainda mais sua aplicabilidade é a grande interação entre decisor e otimizador, recorrentes intervenções humanas podem tornam o método inadequado.

2.2.1 Métodos clássicos de otimização multiobjetivo

A maior dificuldade associada aos problemas multiobjetivo consiste em encontrar soluções factíveis que atendam às necessidades do processo. No entanto, a aplicação de métodos de otimização mono-objetivo em problemas multiobjetivo é, na maioria das vezes, impraticável. Neste contexto, surgem os métodos clássicos de otimização multiobjetivo. Estes métodos têm como meta associar pesos aos objetivos. Dessa forma, os objetivos são escalonados transformando assim o problema multiobjetivo original em um problema mono-objetivo (GOMES *et al*., 2014).

A seguir, são apresentados o método da ponderação dos objetivos e ϵ-restrito, todos pertencentes a classe de métodos clássicos.

2.2.1.1 Método da ponderação dos objetivos

O método da ponderação dos objetivos consiste na atribuição de pesos distintos a cada objetivo, verificando assim uma combinação linear formada por uma função f. Arroyo (2002) apresenta o problema mono-objetivo resultante:

$$Otimizar\ f(x) = \sum_{i=1}^{r} w_i\, f_i(x) \tag{2.5}$$

Sujeito a: $x \in X^*$ (2.6)

No qual, $w_i \geq 0$ refere-se ao peso aplicado ao objetivo f_i em relação aos demais. Sendo que, estes pesos devem satisfazer a condição expressa pela Equação (2.7).

$$\sum_{i=1}^{r} w_i = 1 \tag{2.7}$$

Considerando um vetor de pesos $w = (w_1, \ldots, w_r)$, uma solução x^* é uma solução Pareto-ótima se x^* é uma solução única e $w_i > 0, \forall i = 1, \ldots, r$ (CHANKONG; HAIMES, 1983).

Para construir um conjunto Pareto-ótimo é necessário resolver iterativamente o problema aplicando diferentes vetores de pesos. Os vetores de pesos devem ser definidos pelo decisor de acordo com a prioridade dos objetivos. Ressalta-se que todos os objetivos devem estar na mesma ordem de grandeza ou escala, caso contrário é necessário tornar os objetivos adimensionais para que a solução do problema seja possível (ARROYO, 2002).

Tratando-se de um espaço objetivo não convexo, este método apresenta uma grande desvantagem, pois não consegue gerar todas as soluções Pareto-ótimas (ARROYO, 2002). Considere como exemplo a Figura 2.2 e um problema com dois objetivos. Os pesos w_1 e w_2 são aplicados na função de minimização a seguir:

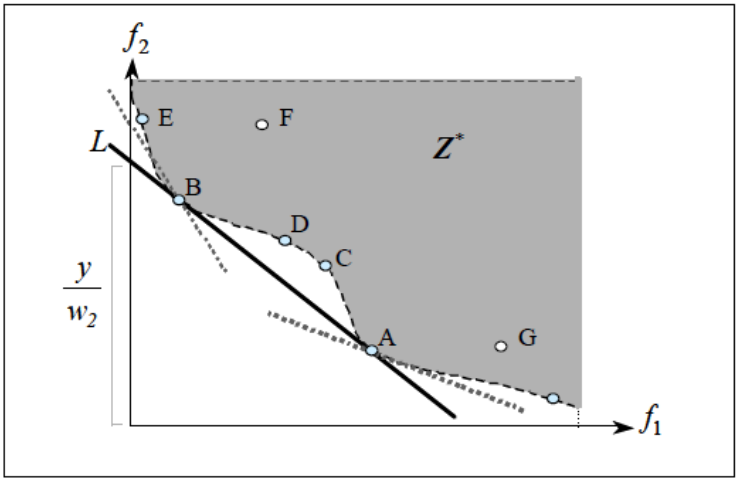
$$y = w_1\, f_1(x) + w_2\, f_2(x), \qquad x \in X^* \tag{2.8}$$

A Equação (2.8) pode ser reescrita como:

$$f_2(x) = -\frac{w_1}{w_2}\, f_1(x) + \frac{y}{w_2} \tag{2.9}$$

A inclinação da reta L é definida pelo coeficiente $-\frac{w_1}{w_2}$ e a interseção desta com o eixo f_2 ocorre em $\frac{y}{w_2}$. Esta reta é tangente ao espaço objetivo factível Z^* em um ponto Pareto-ótimo, sendo então chamada de reta suporte. O método da ponderação dos objetivos consiste em gerar diferentes retas suporte, definidas pelos valores de pesos. No entanto, nem todos os pontos Pareto-ótimos admitem retas suportes. Na Figura 2.2, claramente os pontos C e D não possuem reta suporte, não sendo possível encontrá-los através da minimização da função f, representada pela Equação (2.5).

Figura 2.2: Interpretação gráfica do método da ponderação dos objetivos



Fonte: Arroyo (2002)

2.2.1.2 Método ϵ-restrito

Este método busca a otimização do objetivo mais importante de forma que este atenda as restrições dos demais objetivos. Tomando como exemplo um problema de minimização e f_1 o objetivo prioritário, temos a formulação (ARROYO, 2002):

$$Minimizar \ f_1(x) \tag{2.10}$$

Sujeito a:

$$f_i(x) \leq \varepsilon_i, \ i = 2, \dots, r \tag{2.11}$$

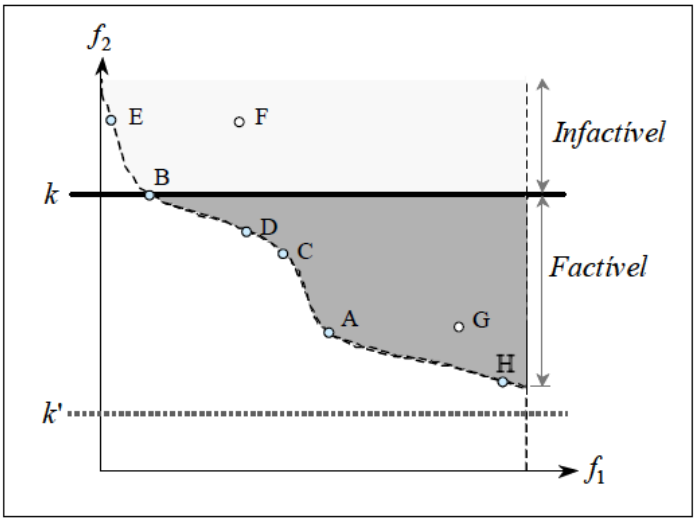
$$x \in X^* \tag{2.12}$$

Na Equação (2.11), ε_i são limites superiores para os n objetivos $f_i, \ i = 2, \dots, r$.

Variando os limites ε_i, é possível construir o conjunto Pareto-ótimo, mesmo em um espaço objetivo não convexo. Sendo as funções objetivos e as restrições lineares, tem-se um problema de programação linear, Equações (2.10), (2.11) e (2.12).

A Figura 2.3 ilustra um problema biobjetivo, onde a reta $\varepsilon_2 = k$ limita o espaço de soluções. Os pontos A, B, C, D, G e H representam as soluções factíveis do problema. Os pontos A, B, C, D e H formam a Fronteira de Pareto. O principal problema deste método é a seleção inadequada dos limitantes ε_i. Caso este não seja selecionado adequadamente, o subespaço obtido pode ser vazio, ou seja, sem soluções. Portanto, um conjunto de valores apropriados devem ser gerados inicialmente para ε_i, a fim de evitar tal situação.

Figura 2.3: Interpretação gráfica do método ϵ-restrito



Fonte: Adapta de Arroyo (2002)

2.2.2 Métodos meta-heurísticos multiobjetivo

Aplicadas com sucesso em problemas de otimização e com capacidade de encontrar boas soluções em um curto tempo computacional, as meta-heurísticas são métodos aplicados para resolver problemas complexos (LIEFOOGHE *et al*., 2012).

As meta-heurísticas foram propostas inicialmente para resolução de problemas mono-objetivo, no entanto, mediante ao êxito de sua aplicação em diversos problemas de otimização, seu emprego foi estendido a problemas multiobjetivo (ARROYO; PEREIRA, 2011). Os autores acrescentam que a meta-heurística algoritmo Genético (AG) é uma das mais aplicadas para solução de problemas multiobjetivos. E outras meta-heurísticas, como GRASP, VNS, ILS, são aplicadas em menor proporção. Fato este, que instiga o desenvolvimento de métodos de resolução aplicando GRASP, VNS e ILS, por exemplo.

Alguns estudos que utilizam meta-heurísticas para a resolução de problemas de otimização multiobjetivo são apresentados a seguir: (*i*) Arroyo e Pereira (2011), por exemplo, sugerem a resolução do problema *flow shop* permutacional através de um algoritmo multiobjetivo baseado na meta-heurística GRASP; (*ii*) Já Arroyo *et al*. (2011) propõem três algoritmos baseados na meta-heurística VNS multiobjetivo (MOVNS) para resolução do problema de sequenciamento de máquina única com sequência dependente do tempo de *setup* e janela de tempo; (*iii*) Gomes *et al*. (2014) sugerem a resolução do problema de programação em projetos com restrições de recursos e relação de precedência entre as atividades, para tanto sugerem cinco algoritmos multiobjetivos baseados nas meta-heurísticas GRASP multiobjetivo,

MOVNS e Pareto ILS; (*iv*) Por sua vez, Xu *et al*. (2017) desenvolveram um algoritmo multiobjetivo baseado na meta-heurística Pareto ILS a fim de solucionar problema *flow shop* permutacional com sequência dependente do tempo de *setup*; (*v*) Nouri e Ladhari (2018) abordam o problema *flow shop* com retenção multiobjetivo e implementam a meta-heurística AG para solução do problema.

A seguir, apresenta-se as meta-heurísticas GRASP, VNS e ILS adaptadas a problemas multiobjetivo.

2.2.2.1 GRASP

O método GRASP é caracterizado como uma meta-heurística construtiva, iterativa, que apresenta resultados rápidos, por isso tem sido amplamente utilizada em problemas de otimização combinatória (GONZÁLEZ-NEIRA *et al.*, 2019a). O GRASP é composto por duas fases: construção e busca local. A Figura 2.4 apresenta o algoritmo do GRASP.

Figura 2.4: Algoritmo GRASP

procedimento *GRASP* $(s, \alpha, maxIter)$;

1 $s \leftarrow \emptyset$;
2 enquanto $iter \leq maxIter$ faça
3 $\quad Construção(s, \alpha)$;
4 $\quad Busca\ Local(s)$;
5 fim enquanto;
6 Retorna s;

fim *GRASP*;

Fonte: Adaptado de González-Neira *et al*. (2019a)

A fase de construção é responsável por gerar as soluções iniciais, onde os elementos são inseridos no vetor solução um a um de acordo com uma função gulosa que estima o benefício da seleção de cada elemento (GOMES JR. *et al*, 2007). Nesta etapa, os melhores elementos são inseridos em uma lista restrita de candidatos (LRC). Dentre os elementos pertencentes à LRC, um deles é selecionado aleatoriamente para compor a solução inicial. A cada iteração a LRC é atualizada. Este processo é repetido até que a solução inicial esteja formada.

Já a etapa de busca local consiste no refinamento desta solução inicial. A busca local é realizada através da exploração da vizinhança de uma solução. O procedimento de busca local é finalizado quando o critério de parada for atingido (GONZÁLEZ-NEIRA *et al.*, 2019a).

2.2.2.2 VNS

A meta-heurística VNS (*Variable Neighborhood Search*) aplicada a problemas mono-objetivo tem o propósito de encontrar uma solução ótima através de mudanças sistemáticas realizadas na estrutura de vizinhança de uma solução (DUARTE; PARDO, 2020), incluindo ainda técnicas adicionais que permitem explorar ótimos locais. Os autores afirmam ainda que, devido à sua eficácia e simplicidade de implementação, o VNS tem sido aplicado em muitos problemas de otimização, inclusive problemas multiobjetivo.

A meta-heurística VNS foi aplicada em problemas de otimização multiobjetivo pela primeira vez por Geiger (2004). Arroyo *et al.* (2011) afirmam que a abordagem multiobjetivo do VNS (MOVNS) diverge da mono-objetivo pelo fato de que as soluções não dominadas a serem exploradas na iteração e as estruturas de vizinhança pertencentes a um conjunto de estruturas de vizinhança são selecionadas de forma totalmente aleatória.

O MOVNS proposto por Geiger (2004) é apresentado na Figura 2.5. A cada iteração o algoritmo MOVNS seleciona aleatoriamente uma solução não dominada *s*, pertencente a FP, e marca esta solução como explorada para que ela não seja selecionada na próxima iteração caso ainda seja uma solução não dominada.

A cada iteração a estrutura de vizinhança *N* a ser aplicada é selecionada randomicamente e caso a solução seja não dominada é incluída à FP. Quando todas as soluções *s* pertencentes à FP forem marcadas como exploradas, a execução do algoritmo é encerrada.

Figura 2.5: Algoritmo do MOVNS

```
1  Gerar solução inicial s
2  FP = {s}
3  repetir
4      Seleciona s ∈ FP a qual N(s) ainda não foi explorada
5      Seleciona aleatoriamente uma estrutura de vizinhança vz
6      Gerar uma solução vizinha N(s)
7      Atualiza FP com toda s' ∈ N(s)
8      se s ∈ FP então
9          Marcar N(s) como explorada
10     fim se
11 até ∄s ∈ FP com N(s) a ser explorada
```

Fonte: Adaptado de Geiger (2004)

2.2.2.3 ILS

Proposta por Lourenço *et al*. (2003), a meta-heurística ILS é embasada na ideia de que um procedimento de busca local pode ser melhorado a partir da geração de novas soluções de partida, soluções estas obtidas por perturbações aplicadas em um ótimo local. Para aplicar o método em problemas multiobjetivo algumas adaptações são necessárias.

A versão multiobjetivo para o ILS é proposta inicialmente por Geiger (2006), e nomeada como Pareto ILS (PILS). Pesquisadores como Gomes *et al*. (2014) e Xu *et al*. (2017) aplicaram com êxito o PILS.

Figura 2.6: Algoritmo PILS

Determine um conjunto de soluções não-dominadas inicial D^*;

1 Selecione aleatoriamente uma solução $s \in D^*$;
2 **Enquanto** (Critério de Parada = Falso) **faça**
3 $i \leftarrow 1$;
4 **Enquanto** ($i < r \wedge$Critério de Parada = Falso) **faça**
5 **Para** (cada vizinho $s' \in N_{i(s)}$) **faça**
6 $D^* \leftarrow$ soluções não dominidas de $D^* \cup \{s'\}$;
7 **Fim_para;**
8 **Se** ($\exists\, s' \in N_i(s) | s'$ domina s) **então**
9 $s \leftarrow s'$;
10 Reordene as estruturas vizinhas $N_1, \ldots, N_r$, em uma ordem aleatória;
11 $i \leftarrow 1$;
12 **Senão** $i++$;
13 **Fim_se;**
14 **Fim_enquanto;**
15 $Mark(s) \leftarrow True$;
16 **Se** ($\exists\, s' \in D^* | s'$ ainda não foi visitada) **então**
17 $s \leftarrow s'$;
18 **Senão**
19 Selecione aleatoriamente uma solução $s' \in D^*$;
20 $s'' \leftarrow$ ***PERTURBA***ÇÃ***O***(s');
21 $s \leftarrow s''$;
22 **Fim_se;**
23 **Fim_enquanto;**
24 **Retorna** D^*;

Fonte: Gomes *et al*. (2014).

O PILS aplicado por Gomes *et al*. (2014), Figura 2.6, é iniciado com a geração do conjunto de soluções não dominadas D^*. Em seguida uma solução $s \in D^*$ é selecionada aleatoriamente e tem toda sua vizinhança explorada. Havendo uma solução vizinha $s' \in N_i(s)$ que domine a solução corrente s, então s' passa a ser a solução corrente. Em seguida as estruturas de vizinhança são reorganizadas de forma aleatória e o procedimento retorna à primeira estrutura de vizinhança da nova ordem. O procedimento é repetido até que todas as soluções em D^* sejam exploradas.

Em seguida, o procedimento de perturbação é aplicado em uma solução aleatória $s' \in D^*$. A perturbação tem como objetivo buscar outras soluções não dominadas. Este procedimento é repetido até que determinado critério de parada, como tempo, número iterações sem melhora, identificação de solução não dominada, seja atendido.

2.3 Simulação e otimização

Os problemas de otimização multiobjetivo podem ser classificados em determinísticos e estocásticos. Os determinísticos são aqueles onde a incerteza não é apontada, dessa forma todos os dados do problema são previamente conhecidos. Já nos estocásticos incertezas são consideradas. Sendo assim, não se dispõem previamente de todas as informações do problema, pois toda ou parte delas assumem valores aleatórios (PINEDO, 2016). Ao conceituar o comportamento aleatório em problemas de otimização multiobjetivo, assume-se que parte das informações do problema não são antecipadamente conhecidas podendo estas assumirem valores aleatórios não negativos.

Uma crescente tendência em pesquisar e solucionar problemas de otimização, incluindo problemas clássicos, considerando parâmetros estocásticos é verificada na literatura (JUAN *et al.*, 2015). Marković *et al.* (2020), Latorre-Biel *et al.* (2021) e Calvet *et al.* (2019), por exemplo, abordam o problema clássico de roteamento de veículos adotando parâmetros estocásticos. Ghorpade e Corlu (2020) analisam o problema do caixeiro viajante com entregas e coletas seletivas assumindo valores aleatórios. Já Juan *et al.* (2014) e González-Neira *et al.* (2019a) analisam o problema de sequenciamento *flow shop* permutacional estocástico.

A fim de solucionar problemas estocásticos, a aplicação de técnicas de simulação se faz necessária já que estas permitem recompor artificialmente sistemas complexos (JUAN *et al.*, 2015). Sistemas complexos demandam um longo período para desenvolvimento, sendo os processos de validação e verificação difíceis.

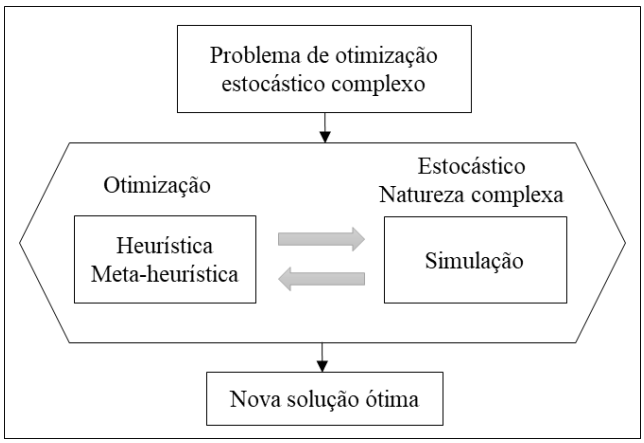
Experimentos de simulação são realizados a fim de analisar e compreender o comportamento do modelo e de suas variáveis. A solução de problemas de otimização multiobjetivo estocástico requer também a aplicação de métodos heurísticos e meta-heurísticos, capazes de conduzir uma busca eficiente por boas soluções.

A combinação entre simulação e técnicas heurísticas e/ou meta-heurísticas geram uma nova classe de algoritmos de otimização chamados simheurísticas. As simheurísticas foram

utilizadas pela primeira vez na década de 1950, no entanto sua aplicabilidade se tornou mais recorrente nos últimos 15 anos (JUAN *et al.*, 2015).

Abordagem simheurística, ilustrada na Figura 2.7, é um bom procedimento para solução de problemas difíceis de otimização combinatória, já que ambas as técnicas interagem entre si a fim de encontrar boas soluções para problemas complexos e estocásticos (JUAN *et al.*, 2015).

Figura 2.7: Visão geral da abordagem simheurística



Fonte: Adaptado de Juan *et al*. (2015)

As simheurísticas podem ser aplicadas em várias áreas. Juan *et al.* (2015) afirmam que alguns dos potenciais campos de aplicação desta metodologia incluem manufatura e produção (DENGIZ; ALABAS, 2000; BYRNE; HOSSAIN, 2005), logística e gestão da cadeia de abastecimento (GANSTERER *et al*., 2014) e área da saúde (BAESLER; SEPULVEDA, 2000; DENTON *et al*., 2006).

O emprego de simheurísticas para solução de problemas clássicos também é facilmente verificada. Marković *et al.* (2020) abordam o problema de roteamento de veículos considerando demandas incertas. Os autores buscam a minimização da distância total, no entanto a demanda em um nó é conhecida apenas depois que um veículo chega ao próprio nó. Esta é a condição incerta do problema. Uma combinação entre simulação, heurística de Clarke e Wright (1964) e meta-heurística *Simulated Annealing* é proposta pelos autores para solução do problema.

Ghorpade e Corlu (2020) analisam o problema do caixeiro viajante com coleta e entrega seletiva considerando parâmetros estocásticos. O objetivo do problema é minimizar os custos de transporte. Para tal, uma integração entre a meta-heurística GRASP e simulação de Monte Carlo é aplica pelos autores. A adoção de parâmetros estocásticos em problemas clássicos, bem como suas respectivas soluções associando simulação e otimização, podem ser vistos também

em Juan *et al.* (2014), Latorre-Biel *et al.* (2021), Calvet *et al.* (2019) e González-Neira *et al.* (2019a).

Um algoritmo simheurístico é desenvolvido para processar de forma inteligente instâncias de problemas de otimização estocásticos (JUAN *et al*., 2015). Os parâmetros estocásticos podem estar presentes na função objetivo ou nas restrições. Diversos parâmetros podem ser considerados como incertos, usualmente é verificado valores estocásticos para o tempo de processamento, demandas de clientes, datas de entrega, distância entre nós, custos, etc. A lógica básica associada à abordagem simheurística, indicada por Juan *et al.* (2015), é apresentada na Figura 2.8.

Dada uma instância de um problema de otimização estocástico, a abordagem simheurística faz a leitura desta instância e executa uma pequena simulação a fim de encontrar valores esperados para os parâmetros estocásticos. Os valores esperados são resultado da média verificada para o parâmetro estocástico na simulação. Os parâmetros aleatórios são então substituídos pelos valores esperados encontrados na pequena simulação, o problema passa a ter uma versão determinística.

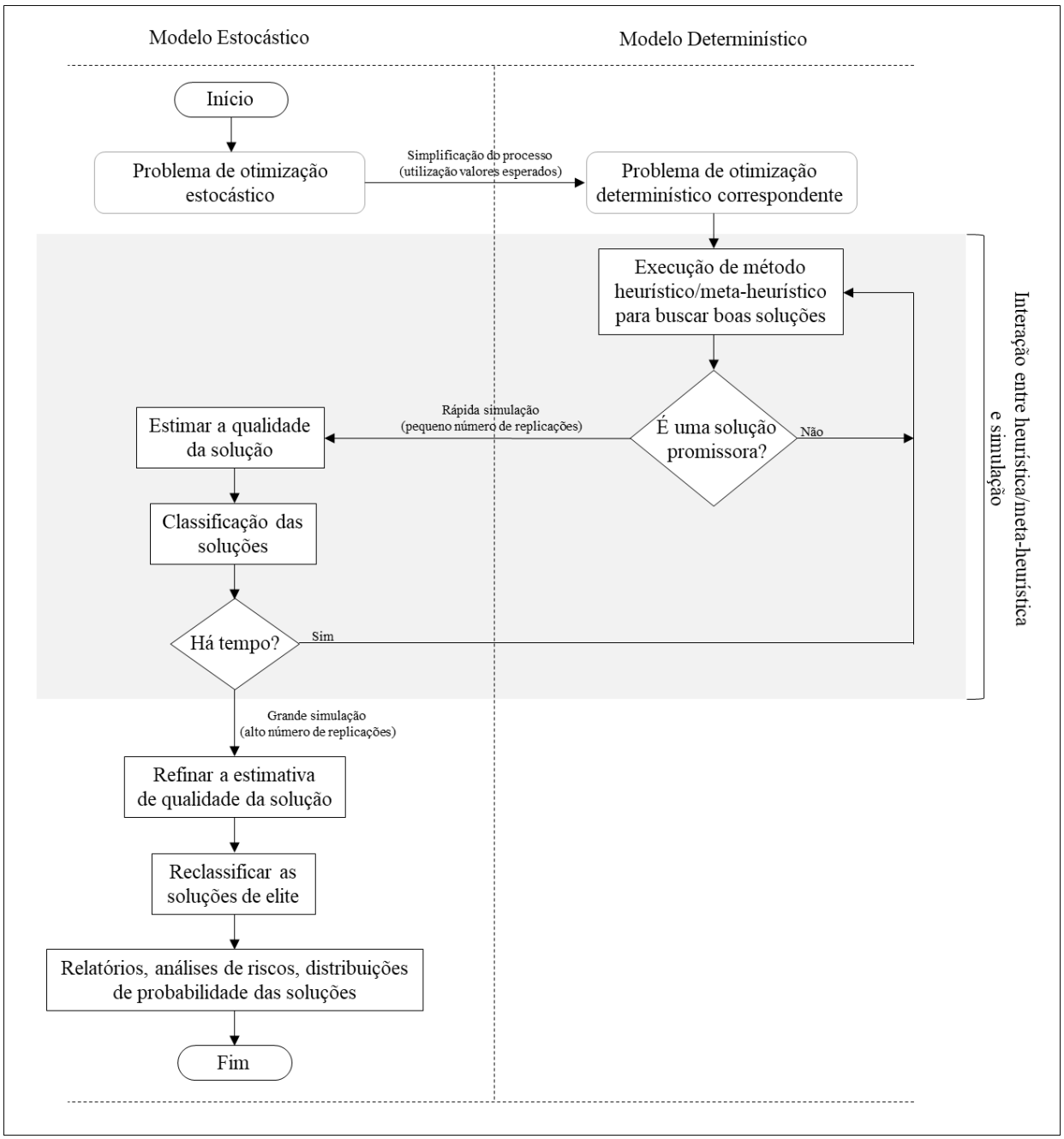
Uma observação importante se faz necessária, Juan *et al.* (2015) afirmam que tratando-se de cenários incertos, soluções de alta qualidade para a versão determinística correspondente ao problema de otimização, também são soluções de alta qualidade para sua versão estocástica.

Técnicas de otimização são então aplicadas, heurísticas ou meta-heurísticas, a fim de encontrar um conjunto de soluções viáveis, dentro do espaço de soluções da versão determinística do problema de otimização. Esta pesquisa iterativa visa encontrar um conjunto de boas soluções. Durante o procedimento de busca o algoritmo deve estimar ou avaliar a qualidade de cada uma das soluções. Caso a solução seja considerada promissora, um pequeno número de replicações é realizado considerando a instância estocástica. Apenas soluções promissoras, ou seja, com bom desempenho na versão determinística são enviadas para simulação.

Caso o tempo computacional pré-definido para o processo iterativo ainda não tenha sido alcançado, representado na Figura 2.8 pela pergunta “Há tempo?”, a heurística ou meta-heurística intensifica a exploração por soluções no espaço das soluções promissoras. Ao atingir o tempo, um número maior de replicações, grande simulação, é realizado e novas estimativas de valores são verificados. Assim, as soluções são reclassificadas.

As simulações finais podem ser utilizadas também para a obtenção de informações adicionais sobre a distribuição de probabilidade da qualidade das soluções, sendo estas informações amparo para a tomada de decisões (JUAN *et al.*, 2015).

Figura 2.8: Lógica básica das simheurísticas



Fonte: Adaptado de Juan *et al*. (2015)

2.4 Programação da produção

Uma indústria é composta por diversos setores que interagem entre si, os quais devem possuir um bom planejamento com o objetivo de entregar ao cliente um produto e/ou serviço. Esta possui recursos e tarefas. Os recursos podem ser os equipamentos e ferramentas disponíveis para utilização, já as tarefas são as operações necessárias ao longo do processo produtivo. É chamado de programação da produção (*scheduling*) a alocação das tarefas nos recursos em períodos determinados com o objetivo de otimizar um ou mais objetivos do processo de produção (PINEDO, 2016).

Uma programação da produção eficiente pode alcançar diversos objetivos tais como minimização dos custos de produtos, atendimento da data de entrega, minimização do atraso e redução do tempo de processamento total (LUSTOSA *et al*., 2008). Os autores acrescentam ainda que as decisões a serem tomadas na programação da produção possuem um horizonte de tempo limitado seja por horas, dias ou semanas.

Pinedo (2016) sintetiza que a programação da produção realiza a alocação de tarefas as máquinas em um período de tempo. Porém, o processamento das tarefas pode sofrer atrasos inesperados devido as incertezas inerentes ao processo, como equipamento danificado por exemplo. Por isso é importante desenvolver uma programação detalhada das tarefas para se ter maior controle sobre as operações.

Em problemas de sequenciamento, afirma Pinedo (2016), menciona-se *job* como uma sequência de determinado número de tarefas ou operações necessárias para a fabricação de um produto. O autor acrescenta que um problema de sequenciamento é descrito a partir de informações referentes (*i*) aos *jobs*, (*ii*) ambiente de máquinas, e (*iii*) objetivos de otimização.

O problema de sequenciamento considera um determinado número de *jobs* e de máquinas, sendo o número de *jobs* indicado por n e o número de máquinas por m. Neste estudo, o índice i se refere ao *job* e k refere-se a máquina. Dessa forma, o par (i, k) indica o processamento do *job* i na máquina k. Os *jobs* possuem um conjunto de informações, características e restrições, próprias e individuais referentes a seu processamento, as quais são fundamentais para determinação do sequenciamento ideal. As principais informações associadas aos *jobs*, descritas por Pinedo (2016), são apresentadas a seguir:

(a) Tempo de processamento (p_{ik}): o tempo de processamento do *job* i na máquina k. Se o tempo de processamento do *job* i independe da máquina ou se este é processado em uma única máquina, o índice k é omitido.

(b) Data de disponibilidade (r_i): a data de disponibilidade do *job* i é a data em que a solicitação do *job* chegou ao sistema, ou seja, é a data em que o *job* i pode iniciar seu processamento.

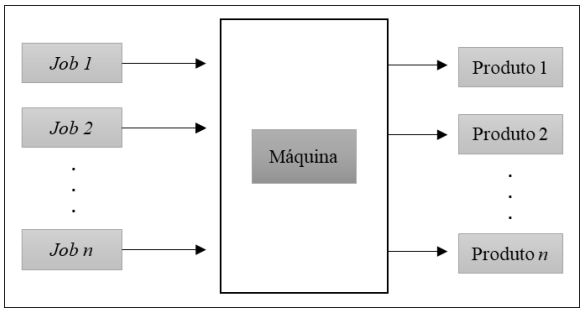
(c) Data de entrega (d_i): denominada também como data de conclusão, é a data da finalização desejada do *job* para entrega ao cliente. Um *job* pode ser concluído após a data de entrega, porém este é finalizado com atraso. Quando a data de entrega deve ser impreterivelmente atendida, esta é referida como prazo final seguindo a mesma denotação.

(d) Peso (w_i): o peso w_i de um *job* i é basicamente um fator de prioridade, denotando a importância do *job* i em relação aos demais *jobs* no sistema. O peso pode representar, por exemplo, o custo real em manter o *job* no sistema, ou pode representar o custo de manutenção ou estoque e por fim pode representar ainda o custo já adicionado ao *job*.

O ambiente de máquinas é o local onde os *jobs* são processados e pode apresentar diferentes configurações, tanto referente à disposição das máquinas quanto ao quantitativo destas. Arenales *et al*. (2015) e Pinedo (2016) descrevem os principais ambientes de máquinas disponíveis na literatura:

(a) Máquina única: é o ambiente de produção mais simples dentre os demais e pode se tornar um caso especial para todos os outros cenários. A Figura 2.9 apresenta este cenário, onde há apenas uma máquina no ambiente para realizar o processamento de n *jobs*. O problema de máquina única não só pode ser aplicado sobre seu próprio ambiente como também em ambientes mais complexos. Comumente, problemas de sequenciamento mais complexos são decompostos em subproblemas com máquina única. Um ambiente com ocorrência de um único gargalo pode ser modelado como máquina única.

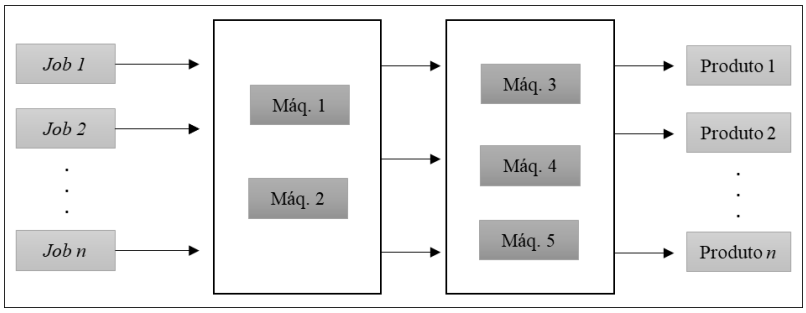
Figura 2.9: Problema de máquina única



Fonte: Autora (2022)

(b) Máquinas paralelas idênticas: é o caso que apresenta m máquinas idênticas em paralelo. O *job* i requer determinada operação e pode ser então processado em qualquer uma das m máquinas que executam tal operação. A Figura 2.10 ilustra este ambiente produtivo tendo-se duas estações de operação, sendo que em cada estação há m máquinas idênticas em paralelo.

Figura 2.10: Problema de máquinas paralelas idênticas

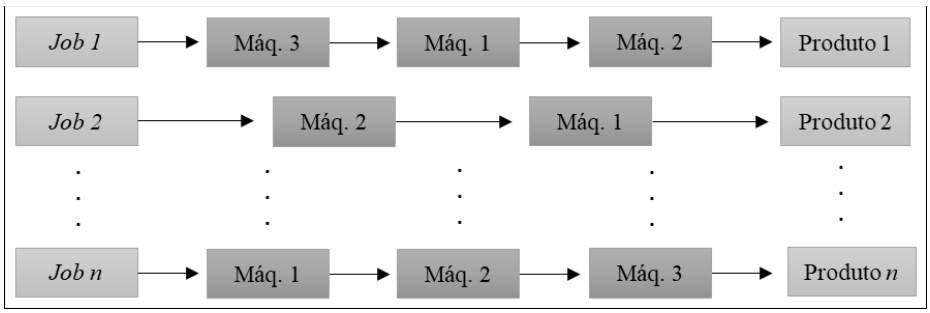


Fonte: Autora (2022)

(c) Máquinas paralelas com diferentes velocidades: ocorre com a existência de m máquinas em paralelo com velocidades diferentes. A velocidade da máquina k é denotada por v_k. O tempo de processamento p_{ik} que o trabalho i gasta na máquina k é igual a p_i/v_k, assumindo que o trabalho i recebe todo seu processamento da máquina k.

(d) Job Shop: é o caso onde há n *jobs* que devem ser processadas nas m máquinas presentes no ambiente produtivo, sendo que cada *job* já possui um roteiro preestabelecido. Ressalta-se que cada *job* não necessariamente deve ser processado em todas as máquinas, bem como a ordem de processamento dos *jobs* em cada máquina não carece ser a mesma. A Figura 2.11 apresenta um exemplo deste caso, onde o *job* 1 deve ser processado nas máquinas 3, 1 e 2, respectivamente. Por sua vez, o *job* 2 deve ser processado nas máquinas 2 e 1, nesta ordem.

Figura 2.11: Problema *job shop*

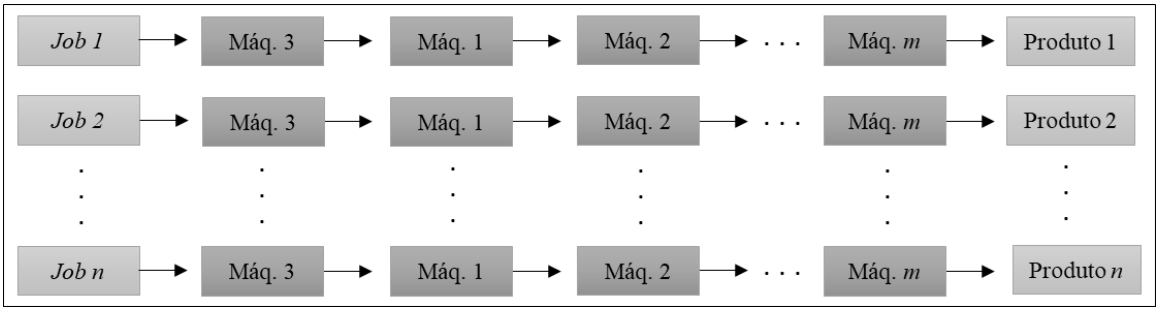


Fonte: Autora (2022)

(e) Flow Shop: é o ambiente com existência de m máquinas em série. Cada *job* deverá ser processado em cada uma das m máquinas. Todos os *jobs* possuem a mesma rota, assim

devem ser processados primeiramente na máquina 1, em seguida na máquina 2 e assim sucessivamente. Após a conclusão em uma máquina, um *job* ingressa na fila para a próxima máquina. Geralmente, presume-se que todas as filas operam sob o sistema *First In First Out – FIFO* (primeiro que entra primeiro que sai), ou seja, o primeiro trabalho da fila será o primeiro a ser processado e concluído. Se a disciplina FIFO for aplicada, tendo-se a mesma sequência dos n *jobs* em todas as m máquinas, o ambiente *Flow Shop* é dito como permutacional. O problema *flow shop* permutacional, foco desta pesquisa, será apresentado detalhadamente em seção específica. A Figura 2.12 ilustra o problema *flow shop*.

Figura 2.12: Problema *flow shop*



Fonte: Autora (2022)

Os objetivos de otimização são a razão pela qual o problema é desenvolvido, ou seja, o problema de sequenciamento é proposto com o intuito de minimizar e/ou maximizar determinadas funções objetivo. Arenales *et al*. (2015) afirmam que os objetivos de otimização estão associados principalmente ao custo e tempo, sendo os mais recorrentes:

(a) Minimizar o Makespan: denotado por C ou C_{max}, o *makespan* representa o instante de término de processamento do último *job* na última máquina, ou seja, momento em que o *job* sai do sistema. O instante de conclusão do trabalho i na máquina k pode ser denotado como C_{ik}. Um valor mínimo de *makespan* implica em boa utilização das máquinas.

(b) Minimizar o Tempo de fluxo total: é somatório do tempo decorrido entre a data disponibilização do *job i* para processamento, r_i, até o término de seu processamento, C_i. Sendo assim, é necessário calcular o fluxo de tempo do *job i*, para enfim verificar o tempo de fluxo total, Equação (2.13).

$$F = \sum_{i=1}^{n} F_i = \sum_{i=1}^{n} C_i - r_i \tag{2.13}$$

(c) Minimização do Atraso Total: relacionado diretamente com a data de entrega, o atraso T_i é calculado pela diferença entre instante de conclusão e a data de entrega, conforme a Equação (2.14):

$$T_i = \max\{0, C_i - d_i\} \tag{2.14}$$

Sendo o instante de conclusão menor que a data de entrega, ou seja, $C_i < d_i$, conclui-se que o *job* foi finalizado sem atraso, $T_i = 0$. Caso contrário, penalidades podem ser aplicadas. A partir destas informações, o atraso total pode então ser calculado pela Equação (2.15):

$$T = \sum_{i=1}^{n} T_i \tag{2.15}$$

(d) Minimização da Antecipação: possuindo também relação direta com a data de entrega, a antecipação E_i é encontrada pela diferença entre a data de entrega e instante de conclusão, Equação (2.16)

$$E_i = \max\{0, d_i - C_i\} \tag{2.16}$$

Busca-se a minimização da antecipação a fim de que o produto não fique muito tempo aguardando para ser entregue ao cliente. Se $d_i > C_i$, o *job i* foi concluído antes da data de entrega desejada e aguardará para ser enviado ao cliente. A antecipação total é então calculada pela Equação (2.17):

$$E = \sum_{i=1}^{n} E_i \tag{2.17}$$

A programação da produção pode ser representada por dois tipos de modelos, determinístico e estocástico. O modelo determinístico assume a existência de um número finito de *jobs* que precisam ser sequenciados a fim de otimizar um ou mais objetivos (VILLARINHO *et al*., 2019). Os autores ressaltam que no modelo determinístico as informações referentes ao problema, tempos de processamento, data de disponibilidade, data de entrega, e outras, são previamente conhecidos e não variam. Incertezas não são consideradas, consequentemente não é possível representar fielmente a realidade.

Já o modelo estocástico adota as informações do problema como aleatórias e não negativas, seguindo uma distribuição de probabilidade (PINEDO, 2016). A consideração de tempos aleatórios caracteriza um cenário mais realista comparado à utilização de tempos

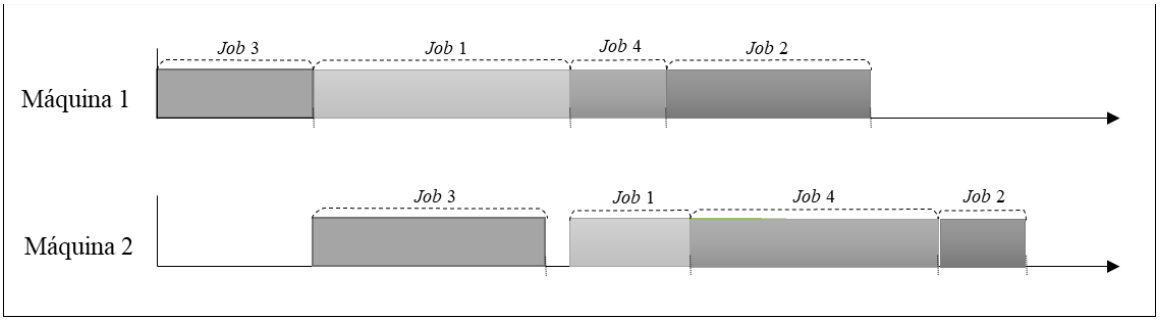
determinísticos. Eventos imprevisíveis tais como quebra ou indisponibilidade de máquina, falta de energia e escassez de matéria prima, ocorrem frequentemente e podem causar grandes mudanças na programação tornando ineficaz a decisão determinística (GONZÁLEZ-NEIRA *et al.*, 2017b).

2.5 Problema *flow shop* permutacional

O PFS consiste no processamento de n *jobs* em todas as m máquinas dispostas no ambiente produtivo. Assumindo que os *jobs* devem ser processados na mesma sequência em todas as máquinas, surge uma variante do PFS conhecida como problema *flow shop* permutacional (PFSP) (GONZÁLEZ-NEIRA *et al.*, 2017a).

O problema *flow shop* permutacional (PFSP) consiste no processamento de um conjunto de *jobs* $N = \{1, 2, \dots, n\}$ por um limitado conjunto de máquinas $M = \{1, 2, \dots, m\}$, sendo idêntica a ordem de processamento dos *jobs* nas máquinas (JUAN *et al.*, 2014). A Figura 2.13 ilustra o PFSP, onde a ordem de processamento dos *jobs* 1, 2, 3 e 4, nas máquinas 1 e 2 é a mesma.

Figura 2.13: Problema *flow shop* permutacional



Fonte: Autora (2022)

2.5.1 Modelagem matemática do PFSP

Arenales *et al*. (2015) apresentam uma formulação matemática para o PFSP, nesse modelo tem-se como função objetivo a minimização do *makespan*, C_{max}. Sendo j o índice que indica a posição dos *jobs* na sequência. O único parâmetro desse modelo é:

p_{ik} = tempo de processamento da tarefa i na máquina k.

Daí define-se as variáveis:

s_{kj} = instante de início de processamento do *job* na posição j na máquina k;

$$z_{ij} = \begin{cases} 1, \text{se a tarefa } i \text{ é designada à } j\text{-ésima posição;} \\ 0, \text{caso contrário;} \end{cases}$$

O modelo é representado pelas equações (2.18) a (2.26):

$$\min C_{max} = s_{mn} + \sum_{i=1}^{n} p_{im} z_{in} \tag{2.18}$$

$$\sum_{j=1}^{n} z_{ij} = 1\,, \quad i = 1, \dots, n \tag{2.19}$$

$$\sum_{i=1}^{n} z_{ij} = 1\,, \quad j = 1, \dots, n \tag{2.20}$$

$$s_{1j} + \sum_{i=1}^{n} p_{i1} z_{ij} = s_{1,j+1}, \quad j = 1, \dots, n-1 \tag{2.21}$$

$$s_{11} = 0 \tag{2.22}$$

$$s_{k1} + \sum_{i=1}^{n} p_{ik}\, z_{i1} = s_{k+1,1}\,, \quad k = 1, \dots, m-1 \tag{2.23}$$

$$s_{kj} + \sum_{i=1}^{n} p_{ik} z_{ij} \leq s_{k+1,j}\,, \;\; j = 2, \dots, n, \;\; k = 1, \dots, m-1 \tag{2.24}$$

$$s_{kj} + \sum_{i=1}^{n} p_{ik} z_{ij} \leq s_{k,j+1}\,, \;\; j = 1, \dots, n-1, \;\; k = 2, \dots, m \tag{2.25}$$

$$s \in R_{+}^{nm}, z \in B^{nm} \tag{2.26}$$

A Equação (2.18) define a função objetivo referente à minimização do *makespan*. As restrições (2.19) garantem que cada *job* i está associado a uma única posição, e as restrições (2.20) asseguram que cada posição j está associada a um único *job*. As restrições (2.21) estabelecem que o *job* na posição j inicie seu processamento na máquina 1 depois que o *job* predecessor tenha sido processado nessa máquina. A restrição (2.22) define que o primeiro *job* da sequência inicie seu processamento na máquina 1 no instante 0.

As restrições (2.23) asseguram que o primeiro *job* na sequência seja processado imediatamente na próxima máquina $\mathrm{k} + 1$, desde que seu processamento na máquina corrente k tenha sido completado.

As restrições (2.24) garantem que um *job* na posição j não pode iniciar seu processamento na próxima máquina $k + 1$ antes do término do seu processamento na máquina corrente k. As restrições (2.25) definem que um *job* na posição $j + 1$ não pode iniciar o processamento em uma máquina k antes que o processamento do *job* na posição j na mesma máquina k tenha sido completado. A restrição (2.26) indica o domínio e o tipo das variáveis.

2.5.2 Problema *flow shop* permutacional multiobjetivo

Usualmente as contribuições literárias para o problema *flow shop* permutacional prevêm a otimização de um único objetivo (GONZÁLEZ-NEIRA *et al.*, 2017a). São verificados como critérios de otimização: a minimização (*i*) do instante de conclusão do *job* na última máquina (ou seja, *makespan*) (VALLADA *et al.*, 2015), (*ii*) do tempo de fluxo total (PAN *et al.*, 2019) e (*iii*) do atraso total (TA *et al.*, 2018). Todavia, em um processo produtivo os tomadores de decisão necessitam otimizar mais de um critério simultaneamente e não apenas um, e estes ainda podem ser conflitantes (ARROYO *et al.*, 2011).

Neste cenário, surge então o problema *flow shop* permutacional multiobjetivo (PFSP-MO). O PFSP-MO tem o propósito de otimizar duas ou mais funções objetivo (KOMAKI *et al.*, 2018). Dentre os possíveis objetivos a serem otimizados no PFSP-MO, é possível afirmar que a minimização do *makespan* é ainda o principal critério aplicado por pesquisadores (GONZÁLEZ-NEIRA *et al.*, 2017a). Cita-se como exemplo Arroyo e Pereira (2011) que propõem em uma mesma pesquisa a minimização simultânea de dois e três objetivos: *makespan* e atraso máximo; *makespan*, atraso máximo e tempo de fluxo total, respectivamente. Xu *et al.* (2017) buscam a minimização do *makespan* e atraso total ponderado. Mishra *et al.* (2020) trabalham para que o *makespan* e o custo de atraso sejam minimizados. E por fim, Tasgetiren *et al.* (2021) associam minimização do *makespan* com a minimização do consumo de energia total.

O PFSP-MO estocástico é visto como uma generalização do PFSP-MO determinístico onde os parâmetros não são constantes, assumindo então valores aleatórios não-negativos para um ou mais parâmetros de entrada. González-Neira *et al.* (2017b) afirmam a existência de três métodos para descrever parâmetros incertos, sendo eles (*i*) *bounded form*, (*ii*) distribuição de probabilidade, e (*iii*) *fuzzy*. O método *bounded form* é utilizado quando não há informações suficientes para descrever uma função de probabilidade, mas há limites inferiores e superiores em que o parâmetro pode variar. Já as distribuições de probabilidade são utilizadas quando se

têm dados históricos suficientes para estimar tais probabilidades. São exemplos de distribuições de probabilidade: Normal, Log-Normal e Exponencial (JUAN *et al.*, 2014). Por fim, o método *fuzzy*, o qual é utilizado para geração de dados estocásticos quando não há dados históricos disponíveis para determinar as distribuição de probabilidade adequadas para tal.

De acordo com González-Neira *et al.* (2017b), 81% dos estudos sobre PFSP estocástico que analisaram em sua pesquisa possuem apenas um critério de otimização, destes 64% buscam a minimização do *makespan*. Os autores afirmam ainda que 8% das pesquisas abordam otimização de dois objetivos e 11% três objetivos ou mais. Juan *et al.* (2014), González-Neira *et al.* (2017a) e Hatami *et al.* (2018), por exemplo, tratam o PFSP mono-objetivo e buscam a minimização do *makespan*. Já nas pesquisas de González-Neira *et al.* (2019a) e González-Neira *et al.* (2019b) é possível visualizar aplicações para o PFSP-MO onde buscam a minimização do *makespan* e atraso, e minimização do atraso e desvio padrão de atraso, respectivamente.

Segundo González-Neira *et al.* (2017a), em problemas *flow shop* permutacional multiobjetivo estocástico pesquisadores investigam principalmente a minimização do *makespan* e atraso total esperados.

Por este motivo, este trabalho investigará o sequenciamento de *jobs* ideal para o PFSP-MO estocástico de modo que o atraso total, o *makespan* e a antecipação esperados sejam minimizados.

2.5.3 Métodos de solução para PFSP-MO estocástico

A fim de solucionar problemas onde a incerteza é considerada, diversos estudos têm sido realizados ultimamente (CHICA *et al.*, 2017). Por serem capazes de encontrar soluções quase ótimas em cenários incertos, os métodos que combinam simulação e otimização utilizando meta-heurísticas, as chamadas simheurísticas ganharam considerável notoriedade.

Algumas aplicações de simheurísticas ao PFSP e ao PFSP-MO têm sido verificadas na literatura. Juan *et al.* (2014), por exemplo, propõem um algoritmo simheurístico para resolução do PFSP com tempos de processamento estocásticos. Os autores combinaram simulação de Monte Carlo e a meta-heurística *Iterated Local Search* (ILS) a fim de minimizar o *makespan* esperado. A abordagem proposta foi capaz de solucionar em minutos instâncias com centenas de *jobs* e dezenas de máquinas.

Abordando o PFSP com montagem distribuída, onde os produtos são finalizados em dois estágios distintos, González-Neira *et al.* (2017a) estudam a versão estocástica do problema considerando os tempos de processamentos dos *jobs*, primeira etapa, e os tempos de montagem, segunda etapa, assumindo valores aleatórios. A fim de minimizar o *makespan* esperado, os autores combinam simulação e a meta-heurística *Greedy Randomized Adaptive Seach Procedure* (GRASP).

Seguindo a mesma linha de pesquisa Hatami *et al.* (2018) também abordam o PFSP com tempos de processamento estocásticos. Um algoritmo simheurístico é proposto pelos autores a fim de minimizar o *makespan* esperado através da combinação entre simulação e ILS.

Tratando-se do PFSP-MO, González-Neira *et al.* (2019a) utilizaram tempos de processamento incertos e desenvolveram uma abordagem simheurística aplicando simulação de Monte Carlo e a meta-heurística GRASP a fim de identificarem um conjunto de soluções não dominadas que minimizem o *makepan* e atraso esperadas.

González-Neira *et al.* (2019b) também abordam o PFSP-MO e consideram o tempo de processamento como fator incerto. Combinando Busca Tabu, estratégia de evolução de Pareto e simulação de Monte Carlo os autores buscam a minimização do atraso esperado e do desvio padrão de atrasos esperado.

3 MÉTODO DE PONDERAÇÃO DOS OBJETIVOS APLICADO AO PFSP-MO

O Capítulo 3 apresenta a aplicação do método de ponderação dos objetivos ao PFSP-MO, para isso adaptou-se um modelo de programação linear inteira mista ao problema abordado neste trabalho. A resolução do modelo matemático foi feita em *software* de otimização específico. As instâncias-teste utilizadas e suas respectivas adaptações também são apresentadas. Os resultados e análises observados após execução do modelo matemático são apresentados ao final desse capítulo.

3.1 Modelo adaptado ao PFSP-MO

O modelo matemático utilizado foi adaptado do modelo para PFS proposto por Arenales *et al*. (2015). O problema *flow shop* permutacional multiobjetivo abordado possui um conjunto de *jobs* $N = \{1, 2, \dots, n\}$ os quais devem ser processados por um conjunto de máquinas $M = \{1, 2, \dots, m\}$, sendo que a ordem de processamento dos *jobs* em todas as máquinas deve ser a mesma. Cada *job* $i \in N$ possui um tempo de processamento p_{ik} para cada máquina $k \in M$ e uma data de entrega desejada d_i, ambas informações previamente conhecidas. Assume-se que todos os *jobs* e máquinas estão disponíveis para processamento no instante zero, $r_i = 0$, sendo que após iniciarem seus respectivos processamentos estes não podem ser interrompidos.

Sendo j o índice que representa a posição de processamento do *job* nas máquinas, tem-se então os parâmetros de entrada do problema:

n: número de *jobs* a serem processados;

m: número de máquinas do ambiente produtivo;

p_{ik}: tempo de processamento do *job* i na máquina k;

d_i: data de entrega desejada para o *job* i;

p_{at}: peso por atraso na função objetivo;

p_{mk}: peso do *makespan* na função objetivo;

p_{el}: peso por antecipação na função objetivo.

As variáveis de decisão para o PFSP-MO proposto são:

s_{kj}= instante de início de processamento do *job* da posição j na máquina k;

$$z_{ij} = \begin{cases} 1, \text{se a tarefa } i \text{ é designada à } j\text{-ésima posição;} \\ 0, \text{caso contrário.} \end{cases}$$

T_i= atraso do *job* i;

E_i= antecipação do *job* i;

C_i= *makespan*.

Daí tem-se, então, a seguinte função objetivo:

$$\text{minimizar} \quad p_{at} \sum_{l \in N} T_l + p_{mk} \left(s_{mn} + \sum_{i \in N}^{n} p_{im}\, z_{in} \right) + p_{el} \sum_{l \in N} E_l \tag{3.1}$$

Sujeito às restrições:

Equações (2.19), (2.20), (2.21), (2.22), (2.23), (2.24), (2.25)

$$C_l \geq s_{ml} + \sum_{i=1}^{n} p_{im}\, z_{il} \ , \quad l = 1, \dots, n \tag{3.2}$$

$$T_l \geq C_l - \sum_{i=1}^{n} d_i\, z_{il} \ , \quad l = 1, \dots, n \tag{3.3}$$

$$E_l \geq \sum_{i=1}^{n} d_i\, z_{il} - C_l \ , \quad l = 1, \dots, n \tag{3.4}$$

$$s, T, E, C \in R_{+}^{nm}, z \in B^{nm} \tag{3.5}$$

A função objetivo representada pela Equação (3.1) busca a minimização do atraso total, do *makespan* e da antecipação total. Por se tratar de um problema de otimização multiobjetivo, atribui-se para cada objetivo um coeficiente de ponderação distinto, p_{at}, p_{mk}, p_{el}, permitindo assim a priorização, ou não, de um determinado objetivo. As últimas desigualdades do modelo proposto têm como objetivo: (3.2) estabelecer o instante de conclusão do processamento do *job i* da posição *l* na última máquina; (3.3) calcular o atraso do *job i* que ocupa a posição *l*; e (3.4) computar a antecipação do *job i* na posição *l.* Por fim, as restrições (3.5) definem o domínio e o tipo das variáveis.

3.2 Instâncias utilizadas

As instâncias utilizadas nesse trabalho foram adaptadas de Taillard (1993), as quais estão disponíveis para consulta em http://soa.iti.es/problem-instances. Ressalta-se que o conjunto de instâncias selecionadas são utilizadas na literatura e adaptadas por diversos autores como González-Neira *et al.* (2019a), Hatami *et al*. (2018), Xu *et al.* (2017), Vallada *et al.* (2015) e Juan *et al*. (2014).

O conjunto de 240 instâncias caracterizadas por Taillard (1993) como pequenas, possuem dimensão $n\ x\ m$, sendo 10, 20, 30, 40, 50 e 60 *jobs* e 5, 10, 15 e 20 máquinas. Ressalta-se que para cada dimensão há um conjunto de 10 instâncias-teste. Dentre este total, foram selecionadas de forma aleatória 5 instâncias com 10, 20, 30, 40, 50 e 60 *jobs* e 5, 10, 15 e 20 máquinas, totalizando assim 120 instâncias. As instâncias com 20 e 30 *jobs* que não foram selecionadas anteriormente foram adaptadas a fim de gerar novas instâncias com 15 e 25 *jobs* e 5, 10, 15 e 20 máquinas, sendo assim 40 novas instâncias foram geradas. Dessa forma, um conjunto de 160 instâncias com 10, 15, 20, 25, 30, 40, 50 e 60 *jobs* e 5, 10, 15 e 20 máquinas foram indicadas nesta etapa.

As instâncias utilizadas apresentam as informações referentes ao número de *jobs*, número de máquinas e tempos de processamento de cada *job* em cada máquina, p_{ik}. As instâncias não tinham informações sobre a data de entrega. Dessa forma, utilizando as informações disponíveis foram geradas randomicamente as datas de entrega a partir da Equação (3.13):

$$d_i = P_i * (1 + randon * m) \tag{3.6}$$

No qual P_i é a soma dos tempos de processamento do *job i* em todas as máquinas, $random$ é um valor aleatório entre [0, 1] e m é a quantidade de máquinas da instância.

O método de ponderação dos objetivos é aplicado a este problema, pois busca-se a otimização de três objetivos distintos. O método de ponderação dos objetivos consiste na atribuição de coeficientes de ponderação distintos, ou não, para cada objetivo. Observa-se que a soma dos pesos atribuídos ao atraso p_{at}, *makespan* p_{mk} e a antecipação p_{el} devem ser iguais a 1, ou seja, $p_{at} + p_{mk} + p_{el} = 1$. Neste estudo, foram testadas 30 combinações de pesos distintas para p_{at}, p_{mk} e p_{el}. Sendo possível encontrar as melhores soluções para cada objetivo, bem como soluções que buscam um equilíbrio entre estes.

3.3 Resultados e análises

O modelo matemático proposto e descrito na Seção 3.1 foi implementado utilizando a linguagem AMPL e resolvido pelo solver CPLEX, versão 20.1. Os testes foram executados em um processador Intel Core i5-8265U, CPU 3.00GHz, 8 GB de memória RAM, Windows 10. Nesta etapa, foram executadas 5 instâncias de cada dimensão, 10, 15, 20 e 25 *jobs* e 5, 10, 15 e 20 máquinas, 80 instâncias ao todo.

Após a adaptação das instâncias, todos os parâmetros de entrada necessários à implementação do modelo, n, m, p_{ik} e d_i, estavam disponíveis. A nomenclatura das instâncias traz consigo informações quanto a quantidade de *jobs* e máquinas, bem como a referência da instância: *n_m_ref*. A Tabela 3.1 apresenta, para fins ilustrativos, a instância 10_5_1, utilizada nesta pesquisa, a qual possui 10 *jobs*, 5 máquinas e é a instância com referência 1. A identificação da referência na instância é importante, pois dentre o conjunto de instâncias disponibilizadas há 10 instâncias com esta mesma dimensão.

Tabela 3.1: Instância 10_5_1

Jobs	p_{ik}					d_i
1	45	31	54	54	64	550
2	44	7	52	66	57	724
3	26	19	65	34	27	637
4	74	83	94	76	60	585
5	19	41	31	50	33	569
6	20	28	42	63	29	759
7	23	30	10	27	26	654
8	46	19	11	5	36	170
9	76	76	34	6	91	660
10	57	31	33	8	19	486

Fonte: Adaptada de Taillard (1993)

Para cada instância foram aplicadas 30 combinações de pesos distintos, Tabela 3.2, pelo método de ponderação dos objetivos, sendo estes selecionados de forma experimental. Dessa forma um conjunto de 30 soluções para cada instância foi retornado pelo modelo.

Tabela 3.2: Pesos aplicados pelo método de ponderação dos objetivos

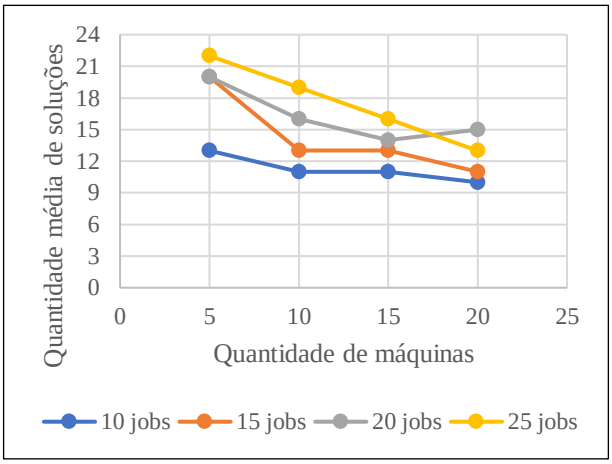
p_{at}	p_{mk}	p_{el}	p_{at}	p_{mk}	p_{el}	p_{at}	p_{mk}	p_{el}
1	0	0	0,1	0,7	0,2	0,5	0,5	0
0	1	0	0,15	0,15	0,7	0,5	0	0,5
0	0	1	0,6	0,1	0,3	0	0,5	0,5
0,9	0,05	0,05	0,1	0,6	0,3	0,5	0,2	0,3
0,05	0,9	0,05	0,2	0,6	0,2	0,3	0,4	0,3
0,05	0,05	0,9	0,2	0,2	0,6	0,2	0,4	0,4
0,8	0,05	0,15	0,25	0,25	0,5	0,4	0,2	0,4
0,05	0,8	0,15	0,2	0,5	0,3	0,4	0,4	0,2
0,1	0,1	0,8	0,25	0,5	0,25	0,33	0,33	0,34
0,7	0,1	0,2	0,1	0,5	0,4	0,3	0,3	0,4

Fonte: Elaborada pela Autora (2022)

Foi verificado que dentre as soluções retornadas pelo modelo houve ocorrência de soluções iguais e também soluções dominadas, sendo estas descartadas. O conjunto de soluções restantes formam então a FP, a qual contém as melhores soluções não dominadas encontradas para o PFSP-MO.

A quantidade média de soluções contidas na Fronteira Pareto varia consideravelmente, no entanto observa-se que quanto maior o número de máquinas menor o conjunto de soluções na FP. A Figura 3.1 apresenta informações sobre o quantitativo médio de soluções pertencentes à Fronteira Pareto das instâncias com 10, 15, 20 e 25 *jobs* e 5, 10, 15 e 20 máquinas. De forma geral, as instâncias apresentam em média 15 soluções na FP.

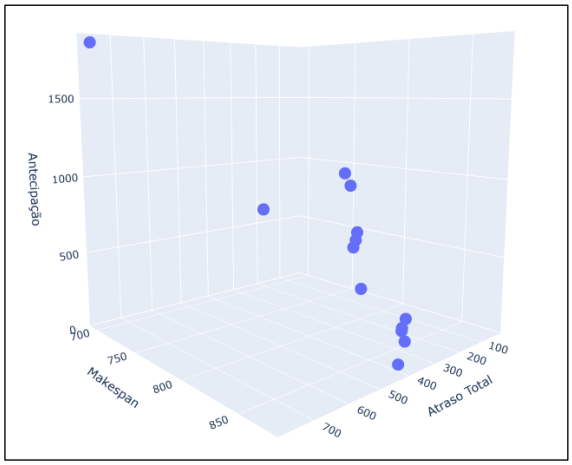
Figura 3.1: Quantidade média de soluções na FP



Fonte: Elaborada pela Autora (2022)

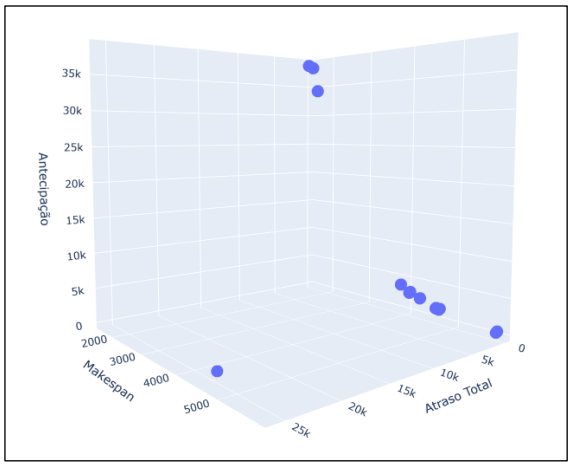
Como o problema em estudo apresenta três objetivos de otimização é possível esboçar a Fronteira Pareto em um gráfico tridimensional. Vale ressaltar que a qualidade da Fronteira Pareto não possui relação com o quantitativo de soluções nela presente. A Fronteira Pareto (FP) formada para a instância 10_5_1, por exemplo, apresenta um conjunto de 13 soluções, Figura 3.2. Já a FP formada para a instância 20_10_2 possui 19 soluções, Figura 3.3. E a instância 25_5_8 possui a FP com 24 soluções, Figura 3.4.

Figura 3.2: Fronteira Pareto instância 10_5_1



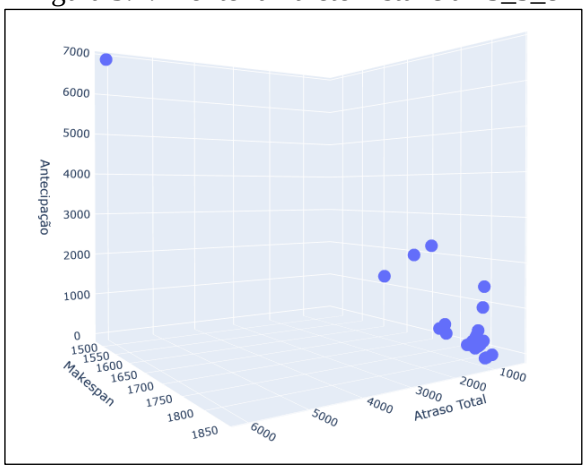
Fonte: Elaborada pela Autora (2022)

Figura 3.3: Fronteira Pareto instância 20_10_2



Fonte: Elaborada pela Autora (2022)

Figura 3.4: Fronteira Pareto instância 25_5_8



Fonte: Elaborada pela Autora (2022)

Como visto nas Figuras 3.2, 3.3 e 3.4 os valores encontrados para o atraso total, *makespan* e antecipação possuem grande variação, ou não, dependendo dos pesos p_{at}, p_{mk} e

p_{el} associados. A Tabela 3.3 apresenta os valores da função objetivo das soluções na FP para a instância 15_5_7. É possível perceber que há considerável variação para os valores de atraso total, *makespan* e antecipação decorrente da alternância entre os pesos associados.

Tabela 3.3: Soluções FP instância 15_5_7

Função Objetivo	Atraso Total	*Makespan*	Antecipação	p_{at}	p_{mk}	p_{el}
8	8	1162	2928	1	0	0
118,8	8	1901	332	0,9	0,05	0,05
146,1	8	1953	280	0,8	0,05	0,15
829,8	11	1804	519	0,4	0,4	0,2
558,5	41	1076	3268	0,5	0,5	0
474,5	52	1953	193	0,5	0,2	0,3
1020,2	94	1804	379	0,25	0,5	0,25
1163	99	1681	673	0,2	0,6	0,2
844,4	130	1901	150	0,3	0,4	0,3
114	130	1953	98	0,5	0	0,5
720,6	138	1953	90	0,33	0,33	0,34
1263,7	186	1165	2148	0,1	0,7	0,2
1191,3	186	1804	301	0,1	0,6	0,3
1020,1	186	1901	108	0,2	0,5	0,3
157,3	186	1953	56	0,05	0,05	0,9
1257,7	195	1134	2272	0,05	0,8	0,15
1097,8	475	1016	3192	0,05	0,9	0,05
939	2710	939	5350	0	1	0

Fonte: Elaborada pela Autora (2022)

Especificamente nesta instância, 15_5_7, é possível verificar que ao priorizar totalmente o atraso total e *makespan*, $p_{at} = 1$ e $p_{mk} = 1$, respectivamente, os melhores resultados foram encontrados para estes objetivos. No entanto, verifica-se que a solução que retorna melhor valor para antecipação possui o peso p_{el} igual a 0,9. Esta situação ocorre pelo fato de que a solução com $p_{el} = 1$ é dominada pela solução com $p_{el} = 0{,}9$, haja visto que essa última apresenta valor menor para o atraso total e o mesmo valor para antecipação total.

Com relação ao tempo computacional para que o modelo seja executado, é notório que instâncias com dimensões menores são executados em poucos segundos. A Tabela 3.4 apresenta algumas informações referentes à execução das instâncias como: (*i*) tempo computacional médio para cada ponderação, refere-se ao tempo médio gasto para executar cada uma das 30 combinações de pesos do método de ponderação dos objetivos; (*ii*) tempo computacional médio para cada instância, apresenta o tempo de execução médio para cada uma das 5 instâncias da

referida dimensão; por fim (*iii*) maior tempo computacional de uma única ponderação, refere-se ao maior tempo de execução de uma única combinação de pesos apurado dentre todos os tempos de execução das ponderações da referida dimensão de instância. Todos os dados referentes a tempo computacional foram aferidos em segundos.

Tabela 3.4: Informações sobre execução das instâncias

n x m	Tempo computacional médio para cada ponderação (s)	Maior tempo computacional de uma única ponderação (s)	Tempo computacional médio para cada instância (s)
10 x 5	0,25	2	7,4
10 x 10	0,58	19	17,4
10 x 15	0,89	10	26,8
10 x 20	1,33	26	40
15 x 5	16,87	1111	506
15 x 10	255,66	31525	7669,8
15 x 15	157,49	3610	4724,8
15 x 20	186,86	3607	5605,8
20 x 5	398,31	3956	11949,4
20 x 10	501,92	4134	15057,6
20 x 15	513,98	3799	15419,4
20 x 20	596,02	4423	17880,6
25 x 5	2530,79	4068	75923,6
25 x 10	859,31	3686	25779,2
25 x 15	1162,84	3603	34885,2
25 x 20	1397,32	3622	41919,6

Fonte: Elaborada pela Autora (2022)

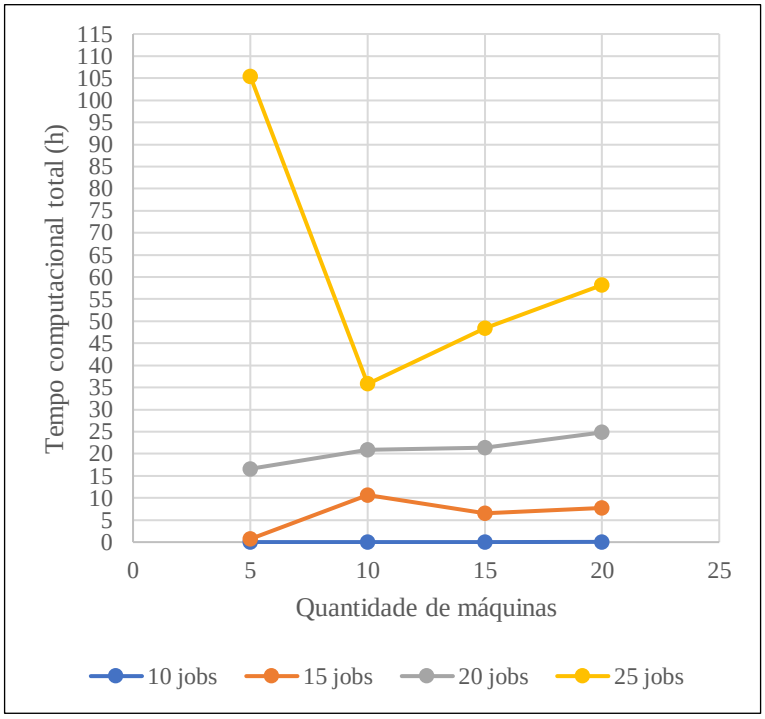
Ao executar o modelo delimitou-se o tempo de execução máximo de 3600 segundos para cada combinação de pesos. Em algumas instâncias, encontraram-se tempos superiores a este valor e isso se deve as tarefas de pré-processamento realizadas pelo CPLEX.

Verifica-se que as instâncias com dimensão igual ou inferior a 15 *jobs* e 5 máquinas são executadas em média com menos de 510 segundos. A menor instância testada, 10 *jobs* e 5 máquinas, por exemplo, foi solucionada, em média, com 7,4 segundos.

Já as instâncias com 25 *jobs* e 5 máquinas, por exemplo, apresentaram tempo computacional médio de 75923,6 segundos, ou seja, aproximadamente 21 horas foram gastas para executar cada instância, pouco mais de 105 horas aplicadas para executar as 5 instâncias com esta dimensão. É possível perceber ainda que em uma das instâncias de dimensão 15x10, instância 15_10_10 especificamente, determinada combinação de pesos foi executada por aproximadamente 9 horas.

Percebe-se então que em instâncias com maior número de *jobs* necessitam de uma grande quantidade de tempo e esforço computacional para serem resolvidos. A Figura 3.5 expõe de forma bastante clara o aumento gradativo do tempo de execução total, em horas, das instâncias à medida que a quantidade de *jobs* aumenta.

Figura 3.5: Tempo computacional total



Fonte: Elaborada pela Autora (2022)

Nesta etapa da pesquisa foi executado um conjunto de 80 instâncias com 10, 15, 20 e 25 *jobs* e 5, 10, 15 e 20 máquinas sendo necessárias 357,52 horas, ou seja, aproximadamente 15 dias computacionais consecutivos para encontrar suas respectivas soluções. Desse modo, conclui-se que a resolução deste problema, para instâncias de grande porte (com grande número de *jobs* e máquinas), é impraticável através da utilização do método de ponderação dos objetivos, por meio de *softwares* de otimização, devido ao tempo computacional necessário para obtenção das soluções. Por este motivo, instâncias com dimensão superior a 25 *jobs*, previamente selecionadas, não foram executadas.

Outra análise importante a ser feita, se refere aos melhores cenários apurados ao fim das execuções. O PFSP-MO em estudo tem como objetivos a minimização do atraso total (T), do *makespan* (C) e da antecipação total (E). As soluções retornadas pelo método são indiferentes entre si, ou seja, cada qual apresenta melhores valores para determinado objetivo. Ao priorizar certo objetivo automaticamente outro será deteriorado. A Tabela 3.5 apresenta, para cada

instância, os valores mínimos apurados para T, C e E. Vale ressaltar que estes valores não foram apurados em uma única solução, mas sim em soluções distintas.

Tabela 3.5: Melhores cenários para T, C e E

Instância	T	C	E	Instância	T	C	E
10_5_1	30	695	53	20_5_1	4	1192	49
10_5_2	0	698	152	20_5_4	606	1127	28
10_5_4	5	697	18	20_5_5	652	1339	26
10_5_7	0	728	299	20_5_8	1272	1102	55
10_5_9	6	761	44	20_5_9	0	1317	159
10_10_1	0	1097	49	20_10_2	0	1525	397
10_10_3	0	1124	625	20_10_5	0	1604	666
10_10_5	0	1093	45	20_10_7	0	1599	259
10_10_6	0	1085	898	20_10_8	32	1580	42
10_10_8	0	1113	161	20_10_9	0	1540	264
10_15_1	0	1307	296	20_15_1	0	1957	12
10_15_2	0	1399	705	20_15_3	51	1826	22
10_15_5	0	1373	137	20_15_6	0	1965	968
10_15_6	0	1329	1359	20_15_8	0	1832	682
10_15_8	0	1443	1674	20_15_9	0	1940	118
10_20_1	0	1652	164	20_20_2	0	2170	670
10_20_3	0	1726	411	20_20_3	0	2309	1102
10_20_6	0	1889	1720	20_20_6	0	2308	392
10_20_8	0	1655	228	20_20_8	0	2198	3526
10_20_10	0	1663	949	20_20_10	0	2222	273
15_5_2	332	1033	70	25_5_2	1125	1384	7
15_5_3	182	1046	8	25_5_3	621	1436	50
15_5_6	122	874	44	25_5_6	2706	1620	32
15_5_7	8	1016	56	25_5_8	343	1478	13
15_5_10	35	1082	24	25_5_10	2443	1387	134
15_10_1	0	1255	99	25_10_2	0	1898	129
15_10_3	0	1370	562	25_10_3	0	1811	72
15_10_4	0	1272	126	25_10_4	0	1790	180
15_10_6	0	1280	736	25_10_6	0	1840	95
15_10_10	0	1313	153	25_10_8	0	1691	69
15_15_2	0	1667	93	25_15_3	0	2172	42
15_15_4	0	1539	526	25_15_5	0	2138	377
15_15_5	0	1595	630	25_15_6	0	2182	315
15_15_7	0	1591	788	25_15_9	0	2140	86
15_15_10	0	1641	2834	25_15_10	0	2233	189
15_20_1	0	2053	1591	25_20_1	0	2469	1368
15_20_4	0	1945	6495	25_20_3	0	2622	844
15_20_5	0	1931	505	25_20_5	0	2499	1946
15_20_7	0	2111	1542	25_20_8	0	2609	1018
15_20_9	0	2128	2571	25_20_10	0	2650	181

Fonte: Elaborada pela Autora (2022)

Aos analisar estes dados, é possível perceber que em 18 instâncias não é verificado nenhuma solução que apresente atraso total igual a zero, ou seja, o atraso é inevitável mesmo havendo máxima priorização deste objetivo. Com relação a antecipação, é observado valores relativamente altos em algumas instâncias, o que significa que os *jobs* finalizaram seu processamento bem antes da data de entrega estabelecida.

A análise destes resultados é interessante devido a possibilidade de visualizar o melhor cenário para cada objetivo caso o tomador de decisões deseje priorizar algum critério específico. Por exemplo, na instância 10_5_1, caso o tomador de decisões tenha a necessidade de priorizar a minimização do atraso total, a opção que lhe proporcionará o menor valor para este objetivo retornará um atraso de 30 unidades de tempo (u. t.). A partir desta observação, o tomador de decisões poderá apurar os melhores cenários para cada um dos objetivos caso deseje indicar uma única preferência.

4 ABORDAGENS SIMHEURÍSTICAS

Neste capítulo são descritas, detalhadamente, as abordagens simheurísticas propostas e todas as suas etapas. Primeiramente é apresentado como o parâmetro estocástico será incluído nas instâncias, em seguida apresenta-se a representação de uma solução. É especificado ainda: o método de geração da solução inicial baseado na meta-heurística GRASP, a função de avaliação, as estruturas de vizinhança aplicadas e o processo de simulação. Além disso, os algoritmos MOVNS e PILS são expostos. Ao fim, os resultados apurados são apresentados e analisados estatisticamente e por diferentes métricas de avaliação.

4.1 Parâmetro estocástico

Para representar possíveis atrasos ocorridos durante o processamento das tarefas, um parâmetro estocástico, denominado fator de atraso a_{ik}, é gerado e adicionado ao tempo de processamento p_{ik}. Liu *et al.* (2017), por exemplo, aplicam este mesmo parâmetro estocástico em sua pesquisa a fim de representar atrasos diversos como quebra de máquinas, falta de energia, e outros.

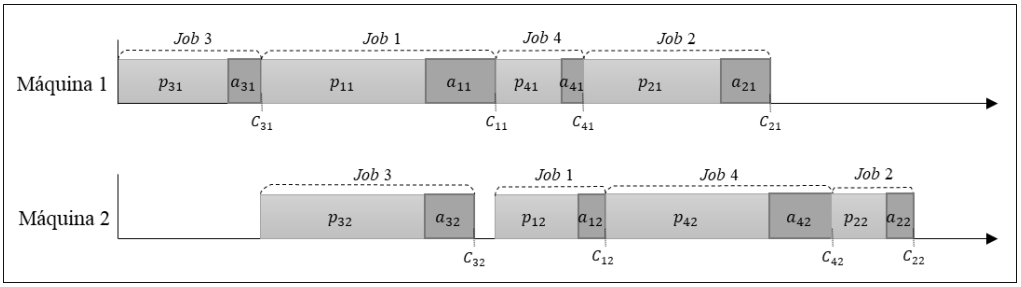
O fator de atraso a_{ik} é estimado em conjunto com os tempos de processamento para determinação do *makespan* de cada *job* sendo então calculado pela Equação (4.1).

$$a_{ik} = w/100 * p_{ik} \tag{4.1}$$

em que *w* é um valor aleatório com distribuição normal (10, 2). Este cálculo é definido de forma que o parâmetro estocástico represente aproximadamente 10% do tempo de processamento do *job*.

O comportamento do parâmetro a_{ik} no PFSP-MO é ilustrado na Figura 4.1. Neste exemplo, os *jobs* serão processados nas máquinas 1 e 2 na mesma ordem, sendo 3, 1, 4 e 2. Dessa forma, é adicionado ao tempo de processamento de cada um dos *jobs* em cada uma das máquinas um valor aleatório gerado pela Equação 4.1(a).

Figura 4.1: Comportamento parâmetro estocástico a_{ik}



Fonte: Elaborado pela autora (2022)

4.2 Representação da solução

O PFSP-MO abordado possui um conjunto de *jobs* $N = \{1, 2, \dots, n\}$ os quais serão processados, na mesma ordem, por conjunto de máquinas $M = \{1, 2, \dots, m\}$, e busca a minimização simultânea do atraso total, *makespan* e antecipação total. Dessa forma, a solução do problema é representada por um vetor solução com duas posições que apresentará a sequência de processamento dos *jobs*, o valor do atraso total, do *makespan* e da antecipação total, respectivamente.

O vetor solução apresentará em sua primeira posição um outro vetor de tamanho variando entre $[1, n]$, o qual é utilizado para representar a sequência de processamento dos *jobs* nas máquinas. Sendo, o valor alocado na primeira posição, $j = 1$, referente ao primeiro *job* a ser processado. O valor contido na posição $j = 2$ indica o *job* a ser processado em segundo lugar, e assim sucessivamente. Já na segunda posição do vetor solução são apresentados os valores apurados para atraso total, *makespan* e antecipação total, nessa ordem.

Na Figura 4.2, é ilustrado o vetor solução de uma instância com 10 *jobs*, onde o *job* 5 é o primeiro a ser processado, em seguida é processado o *job* 8, posteriormente o *job* 3 e assim sucessivamente. Sendo apurado um atraso total de 173 u.t., *makespan* de 786 u.t. e antecipação de 232 u.t.

Figura 4.2: Representação vetor solução

$$[[5, 8, 3, 7, 1, 10, 6, 9, 4, 2], [173, 786, 232]]$$

Fonte: Elaborado pela autora (2022)

4.3 Geração da solução inicial

Com a finalidade de gerar soluções iniciais para as abordagens simheurísticas, utilizou-se o método de construção da meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*). O algoritmo da metaheurística GRASP implementado é apresentado na Figura 4.3 e descrito a seguir.

Os *jobs* a serem inseridos no vetor de solução inicial *s* são alocados na lista de candidatos *LC*, linha 2, segundo o critério de ordenação menor data de entrega (MDE). Enquanto a *LC* não estiver vazia o laço entre as linhas 3 e 10 é repetido. A cada iteração os candidatos, *jobs*, são avaliados segundo uma função gulosa que determinará o tamanho da lista

restrita de candidatos (LRC), linha 6. Apenas os melhores candidatos, de acordo com o critério de seleção MDE, são inseridos na LRC.

A função gulosa tem o objetivo de avaliar a vantagem da inserção de cada candidato, naquele momento, analisando as datas de entrega dos *jobs*, d_i. A função de avaliação considera a data de entrega mais próxima d_{min} (linha 4), a data de entrega mais distante d_{max} (linha 5) e um parâmetro randômico α, sendo este o responsável por determinar quão gulosa é a função. Neste trabalho, α assume um valor aleatório no intervalo [0, 1].

Após formada a LRC, um *job* é selecionado aleatoriamente e então inserido na solução s (linhas 7 e 8). Ao fim de cada iteração a LC é atualizada (linha 9), bem como a cada iteração uma nova LRC é gerada. Este método iterativo é repetido até que todos os *jobs* sejam inseridos na solução inicial. A solução inicial gerada pelo GRASP é adicionada à FP independentemente de sua qualidade (linha 11).

Figura 4.3: Algoritmo Solução Inicial GRASP

procedimento *Sol_Inicial_GRASP* (n, d_i, s, α);

1 $s \leftarrow \emptyset$;
2 Inicialize a lista de candidatos *LC*;
3 enquanto $(LC \neq \emptyset)$ faça
4 $\quad d_{min} = min_{i \in LC}\{d_i\}$;
5 $\quad d_{max} = max_{i \in LC}\{d_i\}$;
6 $\quad LRC = \{i \in LC \mid d_i \leq d_{min} + \alpha(d_{max} - d_{min})\}$;
7 $\quad$ Seleciona aleatoriamente um *job* $i \in LRC$;
8 $\quad s \leftarrow s \cup \{i\}$;
9 $\quad$ Atualize a lista de candidatos *LC*;
10 fim enquanto;
11 $FP \leftarrow s$;
12 Retorna FP;

fim *Sol_Inicial_GRASP*;

Fonte: Adaptado de Gomes Jr. *et al* (2007)

4.4 Função de avaliação

A partir dos métodos propostos, os *jobs* terão suas respectivas datas de início e término, em cada máquina calculadas. As soluções geradas pelos métodos, bem como suas soluções vizinhas, são avaliadas pelas funções objetivo referentes ao atraso total (Equação 4.1), *makespan* (Equação 4.2) e da antecipação total (Equação 4.3).

$$T_{total} = \sum_{i \in N} T_i \tag{4.1}$$

$$C_{max} = max\, C_i \tag{4.2}$$

$$E_{total} = \sum_{i \in N} E_i \tag{4.3}$$

Na Equação (4.1) e (4.3), T_i e E_i representam o atraso e antecipação de cada *job*, respectivamente, e na Equação (4.3) C_i define o instante de conclusão. Os vetores solução serão comparados entre si aplicando o conceito de dominância de Pareto, descrito na Seção 2.1. Dessa forma, cada objetivo de uma solução *s* é comparado ao mesmo objetivo de outra solução *s'* e as soluções não dominadas são adicionadas à FP.

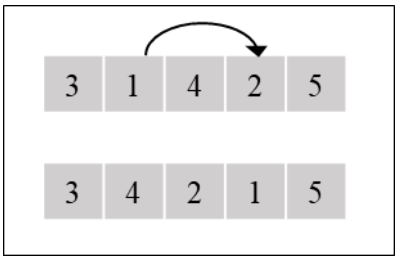
4.5 Estruturas de vizinhança

Ao explorar iterativamente a vizinhança de uma determinada solução, tem-se como objetivo alcançar um ótimo local (LIEFOOGHE *et al.*, 2012). Os autores afirmam também que os algoritmos de busca local multiobjetivo são os responsáveis por esta exploração e acrescentam que as estruturas de vizinhança propostas para tal busca tem relação direta com a qualidade das soluções vizinhas encontradas.

Uma solução vizinha s' é encontra através de transformações locais aplicadas em uma solução s. Estas transformações são chamadas estruturas de vizinhança (ARROYO *et al*., 2011). As soluções vizinhas s' são semelhantes a solução s, já que são originadas através da aplicação de simples movimentos. Neste trabalho, aplicar-se-á as seguintes estruturas de vizinhança:

(a) Inserção: este movimento faz a seleção randômica de um *job* i_p $1 \leq p \leq n$, e insere este *job* em uma nova posição l diferente de sua posição atual $j, l \neq j$. Esta estrutura gera uma vizinhança de tamanho $(n-1)^2$ (GOMES *et al*., 2014). A Figura 4.4 ilustra esta estrutura de vizinhança, na qual o *job* 1 é selecionado e inserido em uma nova posição da sequência.

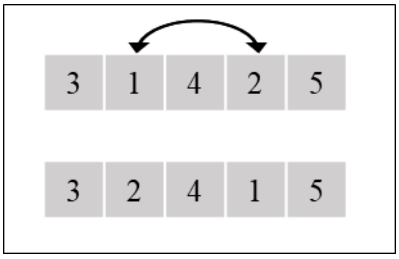
Figura 4.4: Estrutura de vizinhança inserção



Fonte: Elaborado pela autora (2022)

(b) Troca: esta estrutura de vizinhança seleciona aleatoriamente dois *jobs* i_p e i_q da sequência, tal que $1 \leq p \leq n,\ 1 \leq q \leq n,\ p \neq q$ e realiza a troca simultânea da posição destes *jobs*. Esta estrutura de vizinhança pode formar até $n(n-1)/2$ vizinhos distintos (ARROYO *et al*., 2011). A Figura 4.5 representa este movimento, onde os *jobs* 1 e 2 são selecionados e simultaneamente trocados de posição.

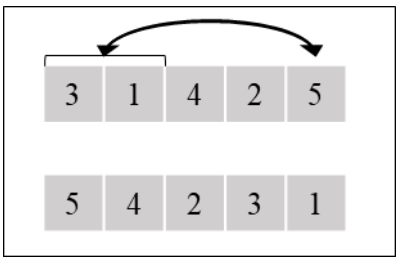
Figura 4.5: Estrutura de vizinhança troca



Fonte: Elaborado pela autora (2022)

(c) Three point move: este movimento troca a posição de dois *jobs* consecutivos com a posição de um terceiro *job* (GROËR *et al*., 2010). A Figura 4.6 apresenta esta estrutura de vizinhança, na qual os *jobs* consecutivos 3 e 1 trocam de posição com o *job* 5.

Figura 4.6: Estrutura de vizinhança *three point move*



Fonte: Elaborado pela autora (2022)

Acrescenta-se que tanto no MOVNS quanto no PILS propostos a exploração da vizinhança de uma solução *s* é feita de forma parcial. Sendo assim, a vizinhança é explorada a fim de que não seja mais verificadas soluções vizinhas não dominadas até o atingimento de determinado critério de parada. Ou seja, a exploração parcial é executada iterativamente e não havendo melhoria na FP é interrompida ao atingir determinado parâmetro.

4.6 Simulação

A simulação é aplicada com o intuito de encontrar valores esperados, $E[\,.\,]$, para as variáveis relativas ao atraso total, *makespan* e antecipação total. Neste estudo, a simulação será aplicada de duas formas diferentes nas simheurísticas propostas e foram nomeadas como simulação rápida e intensiva.

A simulação rápida está inserida na estrutura das simheurísticas MOVNS e PILS, sendo aplicada a cada solução promissora encontrada pelos métodos. Neste momento, uma simulação com pequeno número de replicações é realizada e os valores esperados para o atraso total, o *makespan* e a antecipação são computados. Ao fim da simulação rápida é retornado o conjunto de soluções FP-elite, o qual contêm apenas soluções não dominadas. Demais conceitos acerca da simulação rápida são apresentados na Seção 4.7.

Após o método retornar o conjunto FP-elite, a simulação intensiva é aplicada em todas as soluções nele contidas. Nesta etapa, um número maior de replicações é realizado a fim de fornecer estimativas mais precisas dos valores esperados para os objetivos. O algoritmo representado pela Figura 4.7 apresenta a simulação intensiva.

Figura 4.7: Algoritmo simulação intensiva

procedimento *Sim_Int* (FP_{elite}, $maxRep_{big}$);

1 para todo $s \in FP_{elite}$ faça
2 enquanto $nRep \leq maxRep_{big}$ faça
3 $(a_{ik}) \leftarrow Estocástico()$;
4 $E[s] \leftarrow s(a_{ik})$;
5 $nRep \leftarrow nRep + 1$;
6 fim enquanto;
7 $s \leftarrow E[s]/maxRep_{big}$;
8 fim para todo;
9 para todo $s \in FP_{elite}$ faça
10 se s é dominada por alguma solução $s \in FP$ então
11 $s \notin FP_{elite}$;
12 fim se;
13 fim para todo;
14 Retorna FP_{elite};

fim *Sim_Int*;

Fonte: Elaborado pela autora (2022)

Onde o laço entre as linhas 1 e 8 é executado para todas as soluções pertencentes ao conjunto FP-elite. Sendo que, para cada uma das soluções o laço entre as linhas 2 e 6 é aplicado

iterativamente até que $nRep$ seja igual ao número máximo de replicações, $maxRep_{big}$. Ao fim das replicações os valores estimados para o atraso total, o *makespan* e a antecipação total são calculados, linha 7. Em seguida, o laço entre as linhas 9 e 13 é executado para todas as soluções pertencentes à FP-elite, onde estas são avaliadas pelo conceito de dominância de Pareto sendo as soluções dominadas excluídas do conjunto (linhas 10, 11 e 12).

Ao fim da simulação intensiva tem-se então o conjunto FP-elite compostos apenas pelas melhores soluções verificadas para o PFSP-MO e os respectivos valores esperados para atraso total, *makespan* e antecipação total.

4.7 Abordagens simheurísticas propostas

Devido ao fato de o problema proposto nesta pesquisa não ser tão estudado na literatura, são propostas mais de uma abordagem para solução, especificamente duas, a fim de gerar dados suficientes para promover a comparação entre os resultados obtidos.

As abordagens simheurísticas propostas são baseadas na lógica apresentada por Juan *et al.* (2015), Figura 2.8, e irão combinar simulação com as meta-heurísticas (*i*) GRASP e VNS e (*ii*) GRASP e ILS. Em ambas abordagens, a meta-heurística GRASP é aplicada para geração da solução inicial, conforme descrito na Seção 4.2. Ao aplicar as meta-heurísticas são utilizados os valores determinísticos indicados nas instâncias adaptadas da literatura para os tempos de processamento dos *jobs*. Já nos procedimentos de simulação são incorporados ao tempo de processamento o fator de atraso a_{ik}, apresentado na Seção 4.1.

As simheurísticas são iniciadas a partir da leitura das instâncias do PFSP-MO onde são coletados os dados n, m, p_{ik} e d_i. A partir destas informações é construída a solução inicial pelo método GRASP. Independentemente da qualidade da solução, esta é imediatamente inserida na FP, haja visto que a FP não possui nenhum elemento.

Até que o tempo máximo de execução de 3600 segundos seja alcançado, as meta-heurísticas iniciam os procedimentos de busca local e perturbação, esta última aplicada apenas na PILS, a fim de buscar soluções não dominadas. Cada nova solução encontrada nesta fase é avaliada pela função de avaliação representada pelas Equações (4.1), (4.2) e (4.3). Se a solução em análise não for dominada por nenhuma solução pertencente a FP, esta é então adicionada à FP, caso contrário é descartada.

Após a formação do FP com todas as soluções promissoras, a simulação rápida é aplicada, gerando assim valores esperados para os três objetivos considerados. Neste momento o conjunto FP-elite é criado a partir das soluções não dominadas verificadas após a simulação rápida.

Ao fim da aplicação dos métodos meta-heurísticos, a simulação intensiva é executada em todas as soluções pertencentes à FP-elite. Sendo descartadas as soluções dominadas apuradas após tal simulação. Dessa forma, o algoritmo é capaz de retornar a FP-elite contendo as melhores soluções não dominadas e os valores esperados para atraso total, *makespan* e antecipação.

Nas próximas seções, as simheurísticas propostas, MOVNS e PILS, são apresentadas.

4.7.1 MOVNS

A simheurística MOVNS aqui proposta aplica o método GRASP para gerar a solução inicial e a meta-heurística VNS multiobjetivo como método de busca local.

Neste método, a busca local aplica conceitos decorrentes da meta-heurística VNS para problemas multiobjetivo. A Figura 4.8 apresentada o algoritmo do procedimento MOVNS, onde o laço entre as linhas 1 e 21 é executado até que o critério de parada, número máximo de iterações (*maxIter*) ou tempo, não seja atendido.

Primeiramente uma solução $s \in FP$ é selecionada aleatoriamente e então é marcada como explorada (linhas 2 e 3). A partir da seleção desta solução s, soluções vizinhas s' são exploradas a fim de buscar novas soluções não dominadas. As soluções vizinhas s' são geradas pela aplicação das estruturas de vizinhança, linhas 4 e 5, selecionadas aleatoriamente dentre as pertencentes ao conjunto V, descritas na Seção 4.5.

O valor para a função de avaliação é calculada para s', Seção 4.4. Caso s' seja não dominada por qualquer solução pertencente a FP, esta é adicionada ao conjunto (linhas 6 a 8). Em seguida, outra estrutura de vizinhança, diferente da estrutura aplicada anteriormente, é selecionada e uma solução s'' vizinha de s' é gerada, linhas 9 e 10. Para a solução s'' também é calculado o valor da função de avaliação, sendo s'' adicionada à FP caso seja não dominada (linhas 11 a 13).

Após a aplicação das duas estruturas de vizinhança, caso todas as soluções pertencentes a FP tenham sido exploradas, a marcação de solução explorada é então retirada, linhas 14 a 16, para que as soluções possam ser exploradas nas próximas iterações.

Em seguida, é computado o número de iterações, linhas 17 a 20, aplicando-se o conceito de iterações sem melhora na FP. Sendo assim, caso as soluções vizinhas s' e s'', geradas na referida iteração, tenham sido adicionadas à FP, o número de iteração, *nIter*, receberá valor igual a 1. Caso contrário, soma-se 1 ao então valor corrente de *nIter*.

O método retornará a FP formada após a busca local, linha 22, a qual será submetida à simulação rápida descrita na Seção 4.6.

Figura 4.8: Algoritmo MOVNS

procedimento *MO_VNS* $(n, m, p_{ik}, d_i, FP, V, maxIter)$;

1 enquanto *nIter* $\leq$ *maxIter* ou $tempo \leq 3600$ faça
2 Seleciona aleatoriamente uma solução não explorada $s \in FP$;
3 Marca s como explorada;
4 Seleciona aleatoriamente uma estrutura de vizinhança $V_x,\ x \in V$;
5 Gerar uma solução vizinha $s' \leftarrow V_x(s)$;
6 se s' é não dominada então
7 $FP \leftarrow FP \cup \{s'\}$;
8 fim se;
9 Seleciona aleatoriamente uma estrutura de vizinhança $V_z,\ z \in V, z \neq x$;
10 Gerar uma solução vizinha $s'' \leftarrow V_z(s')$;
11 se s'' é não dominada então
12 $FP \leftarrow FP \cup \{s''\}$;
13 fim se;
14 se toda solução $s \in FP$ foi explorada então
15 Todas as marcações são removidas;
16 fim se;
17 se $s' \in FP$ ou $s'' \in FP$ então
18 *nIter* $\leftarrow$ 1;
19 se não
20 *nIter* $\leftarrow$ *nIter* + 1;
21 fim enquanto;
22 Retorna FP;

fim *MO_VNS*;

Fonte: Adaptado de Arroyo *et al*. (2011)

A simulação rápida, Sim_Rap, é aplicada então a todas as soluções pertencentes à FP. A simulação rápida executa os mesmos procedimentos da simulação intensiva, algoritmo apresentado na Figura 4.7, no entanto possui um número menor de replicações. Ao executar a simulação rápida é possível verificar a ocorrência de soluções dominadas, haja visto que um

fator de atraso a_{ik} é adicionado a cada tempo de processamento. Sendo assim, após a simulação rápida as soluções são avaliadas pelo conceito de Pareto e apenas as soluções não dominadas são adicionadas ao conjunto FP_{elite}.

A simheurística MOVNS proposta, representada na Figura 4.9, é executada pelo período de 3600 segundos, linhas 1 a 4, onde os procedimentos decorrentes do método MOVNS, linha 2 (Figura 4.8), e da simulação rápida, linha 3, são executados. Ao fim deste intervalo de tempo é executada a simulação intensiva, linha 5 (Figura 4.7), e o conjunto FP_{elite} é retornado, linha 6.

Figura 4.9: Algoritmo simheurística MOVNS

procedimento *SimHeu_MO_VNS* $(n, m, p_{ik}, d_i, FP, V, maxIter, maxRep_{peq})$;

1 enquanto $tempo \leq 3600$ faça
2 *MO_VNS* $(n, m, p_{ik}, d_i, FP, V, maxIter)$;
3 *Sim_Rap* $(FP, maxRep_{peq})$;
4 fim enquanto;
5 *Sim_Int* $(FP_{elite}, maxRep_{big})$;
6 Retorna FP_{elite};

fim *SimHeu_MO_VNS*;

Fonte: Autora (2022)

4.7.2 PILS

O PILS desenvolvido neste trabalho combina a meta-heurística GRASP, responsável pela geração da solução inicial (Seção 4.2), e o ILS para otimização de problemas multiobjetivo. O algoritmo PILS é apresentado na Figura 4.10. O PILS é dividido em etapas, a saber: busca local e perturbação, e até que o critério de parada, número máximo de iterações (*maxIter*) ou tempo, não seja atendido executa o laço entre as linhas 1 e 26.

Primeiramente, uma solução ainda não explorada $s \in FP$ é selecionada aleatoriamente e recebe a marcação de solução explorada (linhas 2 e 3). Inicia-se então, a busca local na solução s selecionada a fim de que esta seja explorada. Uma estrutura de vizinhança é então selecionada aleatoriamente dentro do conjunto V e uma solução vizinha s' é gerada, linhas 4 e 5. Caso a solução vizinha s' seja não dominada por nenhuma solução já pertencente à FP, esta é então adicionada à FP (linhas 6 a 8). Independentemente se a solução vizinha s' seja adicionada à FP, ou não, procedimentos de perturbação (linhas 9 a 20) são executados a partir de s' até que o número máximo de iterações, *numIter*, ou o tempo de 3600 segundos seja atingido.

De forma aleatória, é selecionada uma estrutura de vizinhança $x \in V$ e uma solução vizinha de s', denominada s'', é gerada (linhas 10 e 11). A solução vizinha s'' é submetida à avaliação do conceito de Pareto e se for não dominada é adicionada à FP, linhas 12 a 14. Em seguida, uma busca local é também aplicada nesta última solução, onde uma estrutura de vizinha, diferente da anterior, é selecionada e uma nova solução s^* é gerada (linhas 15 e 16). Caso s^* seja não dominada ela é adicionada à FP, caso contrário a solução é descartada, linhas 17 a 19.

Os conceitos de iteração sem melhora também são aplicados neste método. Onde verifica-se, nas linhas 20 a 23, que o contador de iterações receberá valor igual a 1 quando algumas das soluções vizinhas geradas forem adicionadas à FP, ou receberá a soma entre 1 e seu valor corrente caso nenhuma das soluções vizinhas tenham sido adicionadas à FP. Aqui é finalizado os procedimentos de perturbação.

Entre as linhas 25 e 27 é verificado se todas as soluções já foram exploradas e, caso tenham sido, a marcação de solução explorada é retirada. Por fim, entre as linhas 28 e 31 é verificado se houve melhora na FP, ou seja, se s' foi adicionada à FP. Caso $s' \in FP$ o parâmetro *mIter*, contador de iterações, receberá valor igual a 1, caso contrário receberá a soma entre seu respectivo valor e 1. O método é finalizado com o retorno da FP formada, linha 33, a qual será submetida à simulação rápida.

A simulação rápida, apresentada na Seção 4.6, é executada, por um pequeno número de replicações, em todas as soluções não dominadas contidas na FP sendo responsável por estimar valores esperados, considerando os parâmetros estocásticos do problema, para as três funções objetivo consideradas.

Figura 4.10: Algoritmo PILS

procedimento $P\_ILS\ (n, m, p_{ik}, d_i, FP, V, maxIter, numIter)$;

1 enquanto *mIter* ≤ *maxIter* ou $tempo \leq 3600$ faça
2 Seleciona aleatoriamente uma solução não explorada $s \in FP$;
3 Marca s como explorada;
Busca Local
4 Seleciona aleatoriamente uma estrutura de vizinhança $V_x,\ x \in V$;
5 Gera uma solução vizinha $s' \leftarrow V_x(s)$;
6 se s' é não dominada então
7 $FP \leftarrow FP \cup \{s'\}$;
8 fim se;
9 enquanto $nIter \leq numIter$ ou $tempo \leq 3600$ faça
Perturbação
10 Seleciona aleatoriamente uma estrutura de vizinhança $V_x,\ x \in V$;
11 Gera uma solução vizinha $s'' \leftarrow V_x(s')$;
12 se s'' é não dominada então
13 $FP \leftarrow FP \cup \{s''\}$;
14 fim se;
Busca Local
15 Seleciona aleatoriamente uma estrutura de vizinhança $V_z,\ z \in V, z \neq x$;
16 Gera uma solução vizinha $s^* \leftarrow V_z(s'')$;
17 se s^* é não dominada então
18 $FP \leftarrow FP \cup \{s^*\}$;
19 fim se
20 se $s'' \in FP$ ou $s^* \in FP$ então
21 *nIter* ← 1;
22 se não
23 *nIter* ← *nIter* + 1;
24 fim enquanto;
25 se toda solução $s \in FP$ foi explorada então
26 Todas as marcações são removidas;
27 fim se;
28 se $s' \in FP$ então
29 *mIter* ← 1;
30 se não
31 *mIter* ← *mIter* + 1;
32 fim enquanto;
33 Retorna FP;

fim *P_ILS*;

Fonte: Adaptado de Xu *et al.* (2017)

A simheurística PILS proposta neste trabalho, apresentada na Figura 4.11, é executada durante 3600 segundos, linhas 1 a 4. Neste período, o método executa procedimentos relativos ao PILS, linha 2 (Figura 4.10) a fim de apurar soluções não dominadas. Ao término, o método é capaz de constituir a FP, a qual é submetida à simulação rápida, linha 3. Terminado o tempo

de execução, a simheurística executa a simulação intensiva, linha 5, descrita na Figura 4.7, e ao fim retorna o conjunto FP_{elite}, linha 6.

Figura 4.11: Algoritmo simheurística PILS

procedimento *SimHeu_P_ILS* $(n, m, p_{ik}, d_i, FP, V, maxIter, numIter, maxRep_{peq})$;

1 enquanto $tempo \leq 3600$ faça
2 $P\_ILS\ (n, m, p_{ik}, d_i, FP, V, maxIter, numIter)$;
3 $Sim\_Rap\ (FP, maxRep_{peq})$;
4 fim enquanto;
5 $Sim\_Int\ (FP_{elite}, maxRep_{big})$;
6 Retorna FP_{elite};

fim *SimHeu_P_ILS*;

Fonte: Autora (2022)

4.8 Resultados e análises

As abordagens simheurísticas propostas nesta pesquisa foram implementadas em linguagem Python e executadas em um computador com processador AMD *Ryzen* 7 1800X *Eight-Core* 3.60GHz, 64GB de memória RAM e sistema operacional Windows 10 Pro de 64 bits.

As instâncias, os parâmetros e as métricas utilizadas, os resultados obtidos e a análise estatística deles são apresentadas a seguir.

4.8.1 Instâncias utilizadas

Além das instâncias caracterizadas como pequenas, Taillard (1993) disponibiliza ainda 30 instâncias com 100 jobs e 20, 40 e 60 máquinas, as quais foram adaptadas a fim de gerar instâncias com 80 e 100 jobs e 10, 15 e 20 máquinas. As datas de entrega para as instâncias com 80 e 100 jobs também foram geradas de acordo com a Equação (3.13).

Sendo assim, os testes computacionais para as duas simheurísticas propostas foram executados utilizando-se instâncias com 20, 25, 30, 40, 50, 60, 80 e 100 jobs e 10, 15 e 20 máquinas. Instâncias com menor número de jobs não foram testadas já que boas soluções são obtidas pelo método de ponderação dos objetivos.

4.8.2 Métricas de avaliação de desempenho

As soluções obtidas pelos métodos MOVNS e PILS foram comparadas entre si pelas métricas descritas a seguir. Sendo, a FP formada por cada método nomeadas FP_1 e FP_2, respectivamente.

A fim de reunir as melhores soluções encontradas nesta pesquisa para o PFSP-MO, um conjunto de referência é formado por todas as soluções não dominadas obtidas pelos dois métodos propostos, conforme expresso na Equação (4.4). Este conjunto de referência é chamado *Ref* (ARROYO *et al.*, 2011).

$$Ref = \{s \in FP_1 \cup FP_2 | s \text{ é uma solução não dominada}\} \quad (4.4)$$

A performance dos métodos MOVNS e PILS é medida em termos de qualidade das soluções não dominadas obtidas comparando-as às soluções do conjunto *Ref*.

Neste estudo, aplicaram-se as seguintes medidas de desempenho:

(a) número de soluções na Fronteira Pareto: computa, em cada método e cada instância, o quantitativo de soluções não dominadas que compõem a Fronteira Pareto. Assim, será possível verificar quantas soluções Pareto-ótimas os métodos foram capazes de encontrar dentro do limite de tempo estabelecido.

(b) medida de cardinalidade (MC): indica o número de soluções não dominadas obtidas pelo respectivo método que fazem parte do conjunto referência (ARROYO *et al.*, 2011).

$$MC = |FP_i \cap Ref|, \quad i = 1, 2 \quad (4.5)$$

Ressalta-se que quanto maior o valor de MC melhor é o desempenho do método, haja visto que quanto maior este valor mais soluções encontradas pelo método fazem parte de *Ref*.

(c) medidas de distância: mede o quão próximo as soluções não dominadas encontradas pelos métodos, FP_i, estão em relação às soluções de *Ref* (GOMES *et al*., 2014). Nesta métrica é possível calcular (*i*) d_{med}: média das distâncias de cada solução $s \in Ref$ em relação a solução mais próxima $s' \in FP_i$ encontrada pelo método, Equação (4.6), e (*ii*) d_{max}: fornece informações referentes ao máximo das distâncias mínimas de uma solução $s \in Ref$ em relação as soluções $s' \in FP_i$, Equação (4.7).

$$d_{med}(FP_i) = \frac{1}{|Ref|} \sum_{s \in Ref} min\{d(s', s) | s' \in FP_i\} \quad (4.6)$$

$$d_{max}(FP_i) = max_{s \in Ref}\{min\{d(s', s) | s' \in FP_i\}\} \quad (4.7)$$

Sendo $|Ref|$ a cardinalidade de *Ref*, e $d(s', s)$ calculado pela Equação (4.8).

$$d(s',s) = \sqrt{\left(f_1^*(s) - f_1^*(s')\right)^2 + \left(f_2^*(s) - f_2^*(s')\right)^2 + \left(f_3^*(s) - f_3^*(s')\right)^2} \quad (4.8)$$

Onde a função objetivo f_i é normalizada de acordo com o conjunto de soluções de *Ref*, Equação (4.9):

$$f_i^*(s') = 100 \text{ x } \frac{f_i(s') - f_i^{min}}{f_i^{max} - f_i^{min}} \quad (4.9)$$

Sendo f_i^{max} e f_i^{min} os valores máximo e mínimo da função objetivo i no conjunto de referência *Ref*.

(d) Taxa de erro: indica o percentual de soluções pertencentes a FP_i que não estão presentes no conjunto *Ref*. Baseada em Veldhuizen (1999), a taxa de erro, TE_i, é calculada pela Equação (4.10).

$$TE_i = \frac{|FP_i| - |Ref \cap FP_i|}{|FP_i|} \text{ x } 100 \quad (4.10)$$

Onde $|FP_i|$ se refere a cardinalidade do conjunto $FP_i, i = 1,2$, e $|FP_i \cap Ref|$ ao número de soluções pertencentes a Ref oriundas de FP_i.

Vale ressaltar que, quanto menor a porcentagem apurada pela taxa de erro melhor é o desempenho do método. Sendo assim, quando $TE_i = 0$ conclui-se que todas as soluções apontadas pelo método analisado fazem parte do conjunto Ref. De forma análoga, quando $TE_i = 100$ apura-se que nenhuma solução encontrada pelo método em análise faz parte do conjunto Ref.

(e) Diferença de hipervolume: introduzida por Zitzler e Thiele (1998), o hipervolume é responsável por calcular o volume do espaço coberto pelas soluções pertencentes a FP_i, ou seja, a porção do espaço dominado por FP_i. Para calcular o hipervolume é necessário definir um ponto de referência. Em problemas de minimização com dois objetivos, o ponto de referência (x, y) é utilizado para limitar o espaço coberto, sendo x e y os limites superiores para as funções objetivo f_1 e f_2, respectivamente. De forma similar, em problemas de minimização com três objetivos define-se o ponto de referência (x, y, z) como limitante superior para as funções objetivo f_1, f_2 e f_3. Dessa forma, o hipervolume, denotado por $H(FP_i)$, de um conjunto de soluções FP_i será calculado em relação ao ponto de referência.

A métrica diferença de hipervolume $H^*(FP_i)$ apura então diferença entre o volume do espaço dominado pelas soluções pertencentes à Ref, $H(Ref)$, e o volume do espaço coberto pelas soluções de FP_i, $H(FP_i)$. O valor de $H^*(FP_i)$ é calculado pela Equação (4.11).

$$H^*(FP_i) = H(Ref) - H(FP_i) \tag{4.11}$$

Sendo $H(Ref) > H(FP_i)$ e quanto menor o valor de $H^*(FP_i)$ melhor será a qualidade do conjunto FP_i (GOMES *et al*., 2014).

Estas métricas são indicadores muito utilizados em problemas de otimização multiobjetivo, por isso serão aplicadas nesta pesquisa.

4.8.3 Definição dos parâmetros

Ao longo do desenvolvimento das abordagens propostas foi necessário definir alguns parâmetros e critérios de parada a fim de que os métodos possam retornar um bom conjunto de soluções. Alguns parâmetros e critérios são comuns a ambos os métodos e outros específicos de cada um.

O primeiro parâmetro a ser definido foi o critério de parada, tanto para a simheurística MOVNS quanto para a PILS. O critério de parada aplicado foi o tempo de 3600 segundos, mesmo critério de parada e valor empregado no método de ponderação dos objetivos. Entende-se que o tempo de 3600 segundos para gerar um conjunto de sequenciamento de *jobs* que minimizem os critérios estabelecidos é aceitável, principalmente quando há um maior número de *jobs*.

A simulação é aplicada de duas maneiras distintas nos métodos. Primeiramente, é aplicada uma simulação rápida, 30 replicações, nas soluções pertencentes à FP. Posteriormente, nas soluções de FP_{elite} uma simulação intensiva, 100 replicações, é executada.

Os valores para os parâmetros: (*i*) número máximo de iteração no MOVNS, $maxIter_{MOVNS}$; (*ii*) número máximo de iterações na fase de busca local do PILS, $maxIter_{PILS}$; e (*iii*) número máximo de iterações na etapa de perturbação do PILS, $numIter_{PILS}$, foram definidos experimentalmente de forma correlacionada ao número de *jobs* (n) a fim ampliar o número de movimentos de trocas, aplicação das estruturas de vizinhança, à medida que aumenta o número de *jobs* das instâncias.

Para realização dos experimentos, a fim de determinar valores para $maxIter_{MOVNS}$, $maxIter_{PILS}$ e $numIter_{PILS}$, foram utilizadas as instâncias 20_10_9, 25_15_3, 30_15_4, 40_20_1, 50_15_1, 60_10_2, 80_10_3, 100_20_2. A seleção dessas instâncias, para cada quantidade de *jobs*, ocorreu de forma aleatória. Em cada uma das instâncias foram testados seis conjuntos de valores para os parâmetros. Os valores testados foram $maxIter_{MOVNS}$ foram: $\frac{n}{2}, n, 2n, 3n, 4n$ e $5n$. Já para $maxIter_{PILS}$ e $numIter_{PILS}$ foram testados, respectivamente, os conjuntos de valores: $\frac{n}{2}$ e $\frac{n}{4}$; n e $\frac{n}{2}$; n e $2n$; $2n$ e n; $3n$ e $2n$; $\frac{n}{4}$ e $\frac{n}{6}$. Onde n representa o número de *jobs* na instância.

Todas as instâncias foram executadas por 3600 segundos, retornando ao fim da execução o conjunto FP_{elite}. As soluções encontradas para cada dimensão de instância pelos métodos MOVNS e PILS foram comparadas pelas métricas de desempenho: número de soluções na FP, medida de cardinalidade, medida de distância média, medida de distância máxima, taxa de erro e diferença de hipervolume.

A análise para determinação dos valores a ser aplicado nos testes finais foi feita observando-se o quantitativo de melhores e piores resultados indicados pelas métricas de desempenho para cada conjunto de valores.

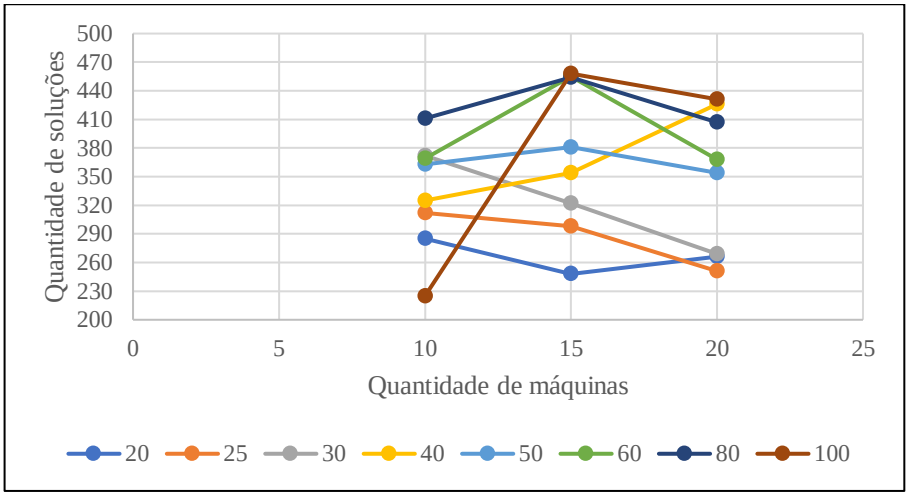
Baseado nos experimentos realizados e nos resultados encontrados, definiu-se: $maxIter_{MOVNS} = 5n$, $maxIter_{PILS} = \frac{n}{4}$ e $numIter_{PILS} = \frac{n}{6}$.

4.8.4 Resultados obtidos

Para cada uma das dimensões de instâncias utilizadas nesta pesquisa (20, 25, 30, 40, 50, 60, 80 e 100 *jobs* e 10, 15 e 20 máquinas) foram selecionadas 5 instâncias-teste distintas. Por esse motivo, todos os resultados apresentados se referem à média dos dados apurados para cada instância e dimensão.

Apesar de não apontar a qualidade da Fronteira Pareto, a análise do quantitativo de soluções presentes no conjunto Ref permite identificar a existência, ou não, de alguma relação entre este dado e a dimensão das instâncias. A Figura 4.12 apresenta as informações sobre o quantitativo médio de soluções em Ref.

Figura 4.12: Quantidade média de soluções em Ref



Fonte: Autora (2022)

De forma geral, com poucas exceções, quanto maior o número de *jobs* na instância, maior a quantidade de soluções em Ref. Com relação a quantidade de máquinas, é verificado que nas instâncias com 20, 25, 30, 50, 60 e 80 *jobs* e 20 máquinas há um menor número de soluções no conjunto referência quando comparadas com os resultados apurados com 10 máquinas. Este comportamento evidencia o aumento do grau de dificuldade em gerar soluções não dominadas em decorrência do aumento do número de *jobs* e de máquinas.

Ao analisar o número médio de soluções na FP em cada um dos métodos, Tabela 4.1, é possível observar que ambos os métodos foram capazes de gerar um maior número de soluções não dominadas em instâncias com maior número *jobs*.

Tabela 4.1: Quantidade média de soluções na FP

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	258	78	50_10	308	196
20_15	231	67	50_15	335	138
20_20	251	57	50_20	344	126
25_10	284	95	60_10	342	187
25_15	271	73	60_15	420	204
25_20	233	61	60_20	331	170
30_10	342	132	80_10	394	224
30_15	297	81	80_15	426	247
30_20	246	55	80_20	381	228
40_10	298	138	100_10	231	138
40_15	322	118	100_15	456	277
40_20	400	115	100_20	422	250

Fonte: Autora (2022)

Analisando a quantidade média de soluções na FP, Tabela 4.1, verifica-se que o MOVNS, por exemplo, foi capaz de encontrar em média 456 soluções não dominadas para instâncias de dimensão 100×15. Já o PILS constituiu Fronteiras Pareto variando entre 55 e 277 soluções. Um comportamento interessante das abordagens, na maioria das instâncias, que vale destacar é a proporcionalidade direta em gerar soluções não dominadas de acordo com o aumento do número de *jobs*, e a diminuição da quantidade destas soluções quando ocorre o aumento do número de máquinas. Fato este decorrente do aumento do grau de dificuldade em sequenciar os *jobs* em uma maior quantidade de máquinas a fim de minimizar os objetivos definidos.

Explorando os dados apurados pela métrica medida de cardinalidade, Tabela 4.2, é evidente que, em todas as instâncias, a maior quantidade de soluções pertencentes a Ref foram geradas pelo MOVNS. Em instâncias com 100 *jobs* e 15 máquinas, por exemplo, o MOVNS incrementou Ref com 455 soluções não dominadas, já o PILS com apenas 3 soluções.

Tabela 4.2: Resultados da métrica medida de cardinalidade

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	245	40	50_10	269	93
20_15	206	42	50_15	331	50
20_20	248	17	50_20	338	16
25_10	255	57	60_10	280	88
25_15	262	36	60_15	402	54
25_20	229	22	60_20	331	37
30_10	323	49	80_10	359	52
30_15	288	34	80_15	419	35
30_20	242	27	80_20	381	26
40_10	277	47	100_10	205	20
40_15	322	33	100_15	455	3
40_20	397	29	100_20	422	10

Fonte: Autora (2022)

Com relação a métrica medida de distância, é apresentado na Tabela 4.3 os resultados sob a perspectiva de distância média e na Tabela 4.4 os dados em relação a distância máxima, sendo que quanto menor o valor calculado para esta métrica melhor a qualidade das soluções na FP. Como as funções objetivo deste problema são aferidas em u.t., todos os dados apurados pelas métricas medidas de distância média e máxima também apresentam-se em u.t.

Tabela 4.3: Resultados da métrica medida de distância – distância média

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	0,59	5,89	50_10	1,57	3,87
20_15	0,49	5,53	50_15	0,82	7,86
20_20	0,11	5,57	50_20	0,28	8,79
25_10	1,04	6,82	60_10	0,99	4,15
25_15	0,86	11,97	60_15	0,67	5,91
25_20	0,77	7,64	60_20	1,11	8,59
30_10	0,72	4,62	80_10	0,60	5,63
30_15	0,58	7,21	80_15	0,71	7,65
30_20	1,04	14,18	80_20	0,64	9,95
40_10	0,75	5,44	100_10	0,70	12,86
40_15	0,64	7,55	100_15	0,07	9,89
40_20	0,52	9,41	100_20	0,26	12,72

Fonte: Autora (2022)

Tabela 4.4: Resultados da métrica medida de distância – distância máxima

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	11,85	45,90	50_10	24,88	28,50
20_15	11,34	46,77	50_15	17,31	39,11
20_20	4,96	41,06	50_20	9,70	45,90
25_10	17,97	47,28	60_10	19,68	27,88
25_15	14,75	55,06	60_15	17,27	29,07
25_20	24,22	39,84	60_20	17,57	32,85
30_10	14,43	27,77	80_10	28,74	34,42
30_15	17,92	52,12	80_15	19,68	35,77
30_20	17,93	54,25	80_20	24,43	38,35
40_10	16,50	41,73	100_10	30,87	37,73
40_15	15,25	34,17	100_15	20,18	28,34
40_20	11,16	40,61	100_20	14,62	32,00

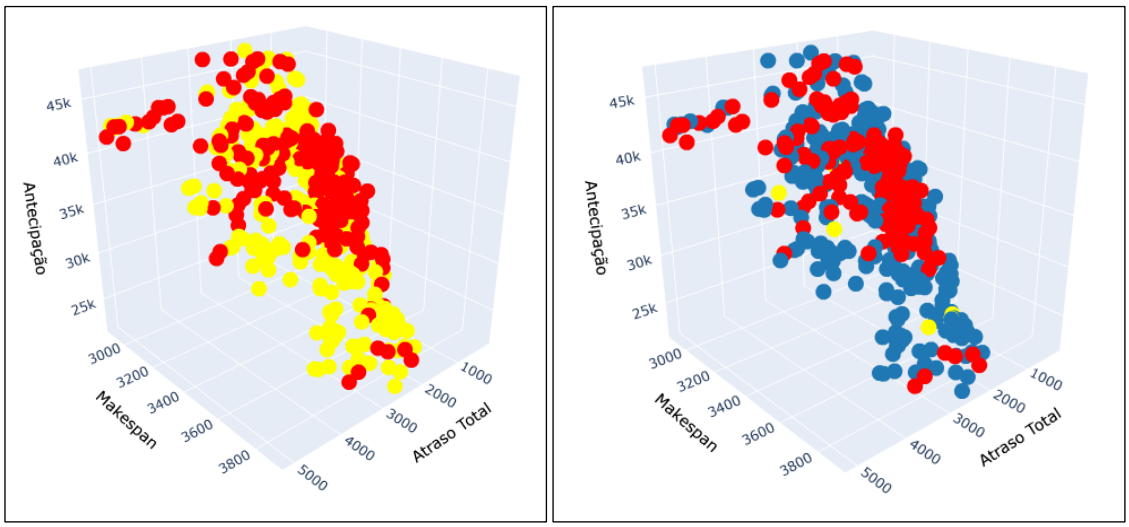
Fonte: Autora (2022)

Analisando as informações contidas nas Tabelas 4.3 e 4.4, observa-se que o PILS apresenta valores maiores tanto para distância média quanto para distância máxima em todas as instâncias. O MOVNS, por exemplo, apresenta valores bem próximos de zero para distância média na maioria das instâncias. Tanto para distância média como distância máxima, quanto menor o valor apurado melhor o desempenho do método. Sendo assim, o MOVNS se apresenta superior em todas as instâncias. Nas instâncias com 40 *jobs* e 10 máquinas, por exemplo, a distância média apurada é de apenas 0,75, já a distância máxima identificada é de 16,50, considerando o MOVNS. Já o PILS, para esta mesma instância, apresenta valores,

aproximadamente, 725% e 253% superiores ao MOVNS nas medidas de distância média e máxima, respectivamente.

Na Figura 4.13 é representada a FP formada pelo MOVNS para a instância 40_10_1, pontos amarelos, e também pelo PILS, pontos vermelhos, e também as soluções pertencentes a *Ref*, pontos azuis. Cada ponto representa uma solução. Observa-se que na Figura 4.13(a) não é identificado nenhuma solução do conjunto *Ref*, obviamente, haja visto que *Ref* é formado apenas pelas melhores soluções dos métodos. Já na Figura 4.13(b) é possível visualizar claramente que a maioria das soluções da FP do MOVNS faz parte de *Ref*, restando poucas soluções distantes. Em contrapartida, boa parte das soluções do PILS estão distantes, não pertencendo a *Ref*. Fato este acentuado pelos dados das Tabelas 4.5 e 4.6.

Figura 4.13: Ilustração medida de distância – instância 40_10_1



(a) (b)

Fonte: Autora (2022)

Com relação à taxa de erro, foram computados os resultados apresentados na Tabela 4.5, em termos percentuais. Quanto mais próximo zero é o valor da taxa de erro, menor o número de soluções do método que não fazem parte de *Ref*. Nota-se que o PILS em instâncias 100×15 apresenta uma taxa de erro superior a 98%, ou seja, pouco mais de 1% das soluções encontradas pelo método fazem parte de *Ref*.

Tabela 4.5: Resultados da métrica taxa de erro (%)

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	4,70	48,04	50_10	13,41	48,90
20_15	9,10	43,24	50_15	1,08	62,46
20_20	0,97	68,79	50_20	1,58	87,16
25_10	12,22	38,95	60_10	17,91	52,74
25_15	4,32	47,29	60_15	4,19	73,05
25_20	2,22	63,04	60_20	0,27	72,52
30_10	6,20	58,92	80_10	7,58	77,82
30_15	2,78	54,29	80_15	1,60	86,46
30_20	2,15	43,21	80_20	0,17	88,88
40_10	7,11	63,99	100_10	10,22	80,00
40_15	0,07	71,79	100_15	0,09	98,95
40_20	0,71	72,44	100_20	0,09	96,24

Fonte: Autora (2022)

A última métrica analisada é a diferença de hipervolume, cujos resultados são exibidos na Tabela 4.6. Essa métrica é responsável por computar o volume do espaço de soluções dos respectivos métodos que não fazem parte, estão fora, do espaço de soluções de *Ref*. Verifica-se que o algoritmo MOVNS apresentou menores valores médios para a diferença de hipervolume em todas as instâncias quando comparado ao PILS. Significa, então, que a FP do MOVNS possui maior cobertura do espeço de soluções de *Ref*.

Tabela 4.6: Resultados da métrica diferença de hipervolume (10^{12})

Instância	MOVNS	PILS	Instância	MOVNS	PILS
20_10	0,06	0,39	50_10	3,95	11,71
20_15	0,03	0,23	50_15	3,34	15,10
20_20	0,03	0,34	50_20	1,02	11,62
25_10	0,16	0,96	60_10	4,48	40,37
25_15	0,13	0,53	60_15	6,59	36,31
25_20	0,07	0,82	60_20	4,09	22,14
30_10	0,34	2,27	80_10	4,59	82,75
30_15	0,23	1,70	80_15	9,51	70,16
30_20	0,14	0,83	80_20	9,91	90,47
40_10	0,89	6,21	100_10	1,03	40,50
40_15	1,36	7,03	100_15	1,20	36,64
40_20	0,87	8,27	100_20	8,05	19,99

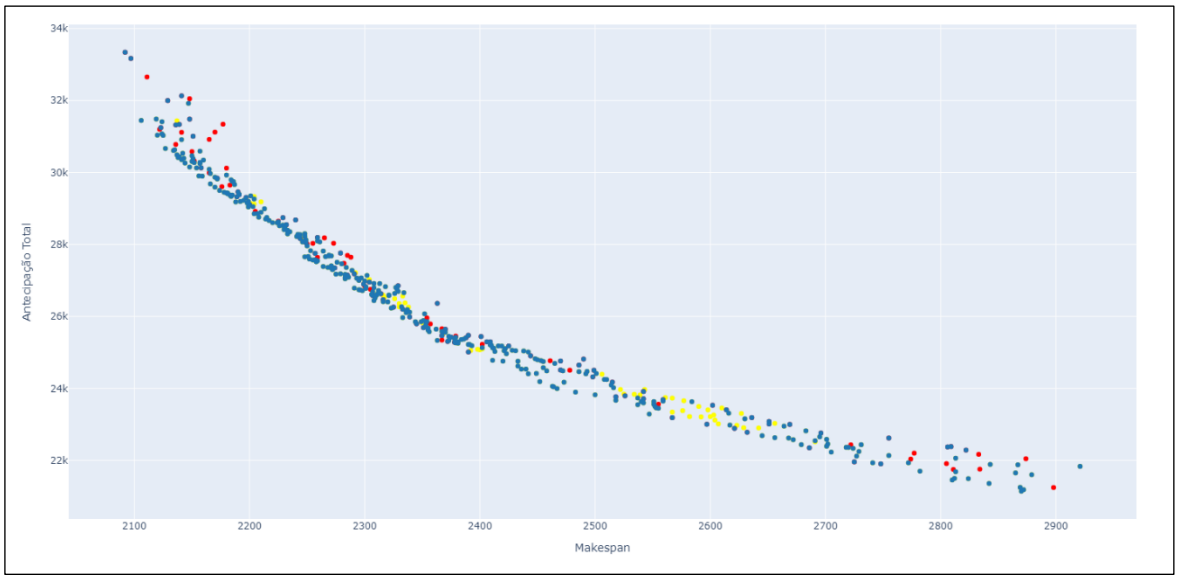
Fonte: Autora (2022)

Nas Figuras 4.14 (a) e (b) são representadas de forma bidimensional a FP do MOVNS, do PILS e o conjunto *Ref*, pontos amarelos, vermelhos e azuis, respectivamente de uma

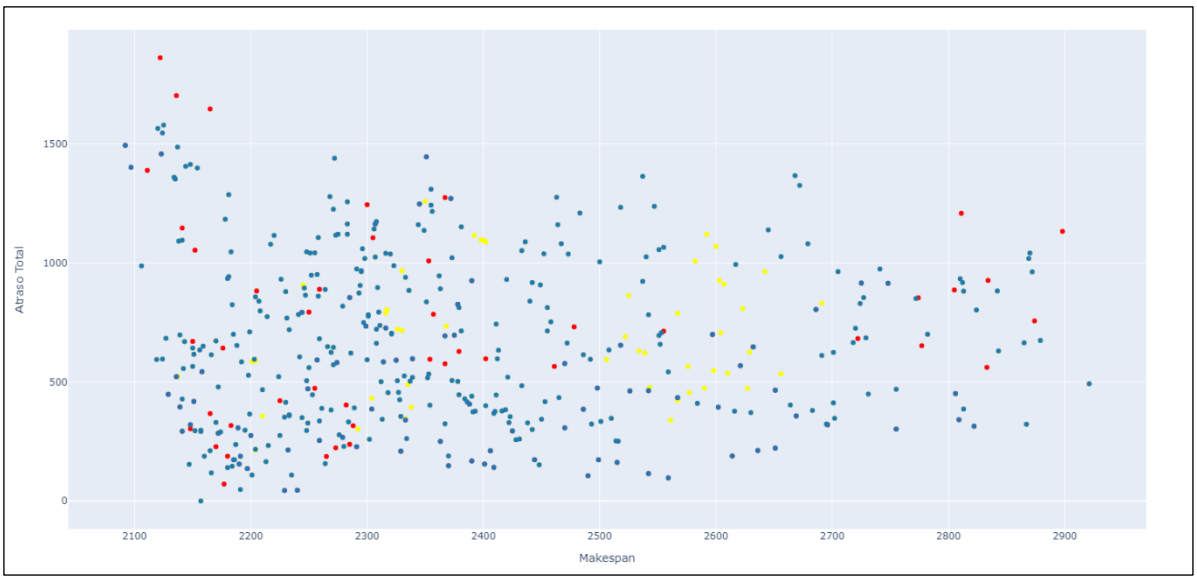
instância com 25 *jobs* e 10 máquinas. Analisando as Figuras 4.14 (a) e (b) é possível perceber como *Ref* abrange grande parte do espaço de soluções, ou seja, possui o maior volume de espaço quando comparado aos métodos.

Por se tratar de uma representação bidimensional das soluções não dominadas de um problema com três objetivos de otimização, é possível haver soluções aparentemente dominadas, quando analisadas por apenas dois objetivos. No entanto, todas as soluções representadas na Figura 4.14 são não dominadas quando analisados simultaneamente os três objetivos de otimização.

Figura 4.14: Espaço de soluções 2D – instância 25_10_6



(a)



(b)

Fonte: Autora (2022)

É perceptível, na Figura 4.14(a), o comportamento conflitante entre o *makespan* e a antecipação total. A melhora de um critério deteriora o outro. Com relação ao atraso total e o *makespan*, Figura 4.14(b), este comportamento não é tão perceptível, apesar de haver soluções que apresentam bons valores para *makespan* e alto atraso total.

Ao analisar a FP formada pelos métodos é notória a existência de diferentes valores para os objetivos. De forma a facilitar a análise destes valores pelo tomador de decisões, por exemplo, é interessante mapear quais são os melhores cenários possíveis para cada uma das instâncias. Vale ressaltar que o melhor cenário é aquele que apresenta menores valores para os objetivos, pois busca-se a minimização deles. Nas Tabelas 4.7 e 4.8 são apresentados os melhores valores apurados para o MOVNS e PILS, respectivamente, em cada um dos objetivos de otimização abordados, onde T representa o mínimo atraso total, C o mínimo *makespan* e E a mínima antecipação total. Ressalta-se que estes valores não são apresentados por uma única solução, mas sim por soluções diferentes.

Tabela 4.7: Melhores cenários para MOVNS

Instância	T	C	E	Instância	T	C	E	Instância	T	C	E
20_10_2	0	1707	27117	20_15_1	133	2208	59529	20_20_2	0	2408	173596
20_10_5	2	1811	31477	20_15_3	498	2012	103252	20_20_3	0	2520	164433
20_10_7	0	1762	23772	20_15_6	0	2192	95892	20_20_6	0	2555	154977
20_10_8	148	1742	22855	20_15_8	0	2019	84199	20_20_8	0	2399	192468
20_10_9	302	1738	20623	20_15_9	166	2171	68720	20_20_10	0	2473	155049
25_10_2	216	2084	14830	25_15_3	447	2265	78394	25_20_1	0	2730	209262
25_10_3	365	1999	24854	25_15_5	0	2401	87521	25_20_3	10	2824	229915
25_10_4	7	1998	30491	25_15_6	0	2386	96625	25_20_5	0	2777	219398
25_10_6	0	2106	21144	25_15_9	422	2315	96706	25_20_8	0	2860	189250
25_10_8	223	1904	17158	25_15_10	0	2485	106463	25_20_10	38	2908	183268
30_10_1	3083	2234	14554	30_15_1	239	2770	94127	30_20_2	0	3216	213542
30_10_5	527	2329	21867	30_15_2	0	2687	93256	30_20_4	0	3061	206144
30_10_7	116	2286	32938	30_15_4	0	2775	140908	30_20_6	0	3121	315328
30_10_9	135	2178	29059	30_15_7	9	2651	123541	30_20_7	0	3107	220842
30_10_10	89	2228	21171	30_15_8	0	2681	132920	30_20_9	69	3160	246162
40_10_1	348	2927	22058	40_15_2	2	3250	130605	40_20_1	635	3814	298683
40_10_3	334	2789	21806	40_15_3	14	3385	154065	40_20_4	113	3766	273694
40_10_4	394	2837	30316	40_15_6	0	3220	137182	40_20_5	24	3574	258956
40_10_6	577	2946	21576	40_15_7	46	3300	124716	40_20_7	127	3774	210300
40_10_8	1168	2866	16005	40_15_9	556	3183	163782	40_20_8	0	3745	283875
50_10_2	1682	3560	9245	50_15_1	269	3888	144809	50_20_1	0	4334	330804
50_10_4	1555	3563	11848	50_15_3	864	3924	136608	50_20_4	30	4442	405206
50_10_7	1049	3369	17565	50_15_6	205	3921	145829	50_20_5	0	4347	359011

Continuação Tabela 4.7

Instância	T	C	E
50_10_9	4430	3449	9836
50_10_10	814	3566	18015
60_10_2	7290	4275	8818
60_10_3	18214	4079	4510
60_10_7	2719	4264	15207
60_10_9	2766	4147	8282
60_10_10	3556	4051	14864
80_10_3	38721	5192	453
80_10_4	11132	5084	3731
80_10_7	39511	5481	604
80_10_9	24285	5334	650
80_10_10	22115	5091	1031
100_10_1	81335	6295	0
100_10_2	68930	6392	0
100_10_5	55046	6277	0
100_10_6	74803	6395	1
100_10_8	64377	6154	202

Instância	T	C	E
50_15_8	60	3992	157157
50_15_10	134	4014	158373
60_15_2	563	4618	120706
60_15_4	342	4524	164353
60_15_5	345	4544	172859
60_15_8	564	4558	130222
60_15_9	472	4477	143891
80_15_2	283	5956	205004
80_15_3	1645	5877	162812
80_15_5	441	5790	158861
80_15_7	689	5810	212052
80_15_9	3403	5798	122009
100_15_1	1209	6986	153331
100_15_4	3611	6832	149501
100_15_6	2854	7032	115244
100_15_8	4793	6980	139689
100_15_10	1875	7055	150275

Instância	T	C	E
50_20_7	0	4371	364836
50_20_8	561	4418	399550
60_20_1	33	4965	374892
60_20_3	588	5184	351886
60_20_6	33	4931	410562
60_20_7	501	5146	340901
60_20_9	202	4963	315479
80_20_1	192	6174	505740
80_20_4	890	6453	500540
80_20_6	53	6428	520076
80_20_8	736	6411	521511
80_20_9	0	6220	514655
100_20_2	1595	7717	485206
100_20_3	295	7473	471832
100_20_5	1124	7611	595848
100_20_7	328	7541	494255
100_20_10	343	7638	563124

Fonte: Autora (2022)

Tabela 4.8: Melhores cenários para PILS

Instância	T	C	E
20_10_2	0	1733	27744
20_10_5	0	1831	31744
20_10_7	0	1780	24213
20_10_8	148	1763	22954
20_10_9	167	1741	20772
25_10_2	213	2092	15214
25_10_3	321	2042	25508
25_10_4	0	2019	32015
25_10_6	44	2092	21247
25_10_8	168	1897	17524
30_10_1	2915	2255	14623
30_10_5	389	2337	22683
30_10_7	32	2286	32982
30_10_9	95	2225	30221
30_10_10	61	2269	23059
40_10_1	203	2904	22787
40_10_3	293	2766	24052
40_10_4	265	2859	34029
40_10_6	577	2943	23986
40_10_8	1187	2894	16020

Instância	T	C	E
20_15_1	65	2216	60315
20_15_3	498	2053	103419
20_15_6	0	2257	96065
20_15_8	0	2096	84686
20_15_9	149	2229	69038
25_15_3	446	2317	80233
25_15_5	0	2402	88112
25_15_6	14	2453	97236
25_15_9	422	2381	97005
25_15_10	0	2463	106529
30_15_1	239	2789	94268
30_15_2	27	2717	95130
30_15_4	0	2817	141853
30_15_7	9	2778	124823
30_15_8	0	2768	135339
40_15_2	0	3294	134261
40_15_3	0	3386	157231
40_15_6	0	3285	143180
40_15_7	0	3380	128164
40_15_9	556	3243	167526

Instância	T	C	E
20_20_2	0	2429	174010
20_20_3	0	2568	164795
20_20_6	0	2614	155686
20_20_8	0	2451	193187
20_20_10	0	2490	155514
25_20_1	0	2721	210791
25_20_3	0	2859	231388
25_20_5	0	2825	220122
25_20_8	0	2815	189738
25_20_10	39	2883	184832
30_20_2	0	3258	217050
30_20_4	0	3047	208674
30_20_6	0	3120	317756
30_20_7	0	3167	221916
30_20_9	68	3244	249111
40_20_1	561	3903	305169
40_20_4	30	3820	277583
40_20_5	0	3730	261623
40_20_7	0	3831	214045
40_20_8	0	3858	286365

Continuação Tabela 4.8

Instância	T	C	E
50_10_2	1857	3525	9421
50_10_4	1438	3541	12651
50_10_7	964	3297	17032
50_10_9	4486	3481	12431
50_10_10	524	3522	19556
60_10_2	7924	4242	9265
60_10_3	18288	4030	4723
60_10_7	2660	4239	16147
60_10_9	3233	4137	8949
60_10_10	4115	4135	13887
80_10_3	39530	5139	1197
80_10_4	12520	5074	7311
80_10_7	42117	5459	760
80_10_9	27354	5319	2444
80_10_10	23170	5080	1675
100_10_1	85000	6273	0
100_10_2	70837	6508	392
100_10_5	62661	6335	181
100_10_6	81590	6392	22
100_10_8	69483	6144	361

Instância	T	C	E
50_15_1	108	3937	150823
50_15_3	844	3992	135454
50_15_6	0	4001	148981
50_15_8	0	4032	165099
50_15_10	135	4078	163164
60_15_2	191	4612	124907
60_15_4	123	4558	173182
60_15_5	102	4613	176776
60_15_8	646	4469	134471
60_15_9	202	4488	152684
80_15_2	250	5921	216914
80_15_3	748	5886	171318
80_15_5	451	5827	162118
80_15_7	700	5811	224869
80_15_9	5082	5836	127372
100_15_1	3421	6955	157643
100_15_4	5622	6961	149947
100_15_6	4131	7207	130716
100_15_8	9900	6966	152616
100_15_10	2871	7051	164670

Instância	T	C	E
50_20_1	16	4432	340926
50_20_4	30	4474	409357
50_20_5	0	4376	363247
50_20_7	1	4455	372915
50_20_8	562	4477	405071
60_20_1	0	5020	378502
60_20_3	390	5231	352376
60_20_6	0	5094	428679
60_20_7	450	5181	349142
60_20_9	0	4987	321743
80_20_1	146	6195	514317
80_20_4	810	6400	518555
80_20_6	132	6629	530025
80_20_8	270	6512	529676
80_20_9	66	6358	538903
100_20_2	1452	7781	500990
100_20_3	121	7548	492534
100_20_5	1043	7722	616418
100_20_7	510	7669	531143
100_20_10	1567	7675	583727

Fonte: Autora (2022)

Analisando as Tabelas 4.7 e 4.8 é possível visualizar com clareza a característica conflitante entre o atraso total e antecipação total. Por exemplo, os valores apurados para o atraso total pelo método MOVNS nas instâncias 30_15_4 e 20_20_10 são nulos, no entanto os valores para antecipação são elevados.

Com o auxílio das Tabelas 4.7 e 4.8 o tomador de decisões pode perceber qual o valor mínimo que pode alcançar para o atraso total, *makespan* e antecipação total durante a execução dos *jobs*. Supondo que, a minimização máxima do *makespan*, aplicando o PILS, da instância 60_15_4 seja o objetivo do tomador de decisão em detrimento dos demais objetivos, este poderá saber então que o valor mínimo que pode alcançar para o *makespan* é 4558 unidades de tempo. Caso o objetivo principal seja minimizar ao máximo a antecipação total da instância 80_10_9, pelo método MOVNS, pode-se verificar que o valor mínimo apurado para esta instância é de 650 unidades de tempo. Esta análise permite então ter uma ampla visão dos valores mínimos que podem ser verificadas nas instâncias em cada um dos objetivos de otimização indicados.

4.8.5 Análise estatística dos resultados

A análise estatística de resultados tem como objetivo apurar a existência de diferença estatística significante entre os algoritmos propostos, investigando os valores retornados pelas métricas de avaliação de desempenho utilizadas. A análise estatística aplicada nesta pesquisa foi executada utilizando a versão 21 do *software* Minitab®.

A análise estatística permite realizar inferências em toda a população de instâncias, 120 ao todo, já que as observações nas duas populações de interesse são coletadas em pares. Montgomery e Runger (2021) afirmam que cada par de observações, (X_{1j}, X_{2j}), é verificado sob homogêneas condições, mesmo as amostras sendo independentes. Sendo assim, se não houver diferença significativa entre estes dados a média de suas diferenças deverá ser zero. Esse procedimento é nomeado de teste *t* pareado.

Assumindo $(X_{11}, X_{21}), (X_{12}, X_{22}), \dots, (X_{1n}, X_{2n})$ como um conjunto de n observações pareadas, considera-se que a média da população representada por X_1 seja μ_1 e a média da população representada por X_2 seja μ_2. É definido por Montgomery e Runger (2021) a diferença entre cada par de observações como:

$$D_j = X_{1j} - X_{2j} \qquad j = 1,2, \dots, n \tag{4.4}$$

sendo D_j distribuído normalmente com média:

$$\mu_D = \mu_1 - \mu_2 \tag{4.5}$$

Dessa forma, considerando uma amostra, é possível testar hipóteses sobre a diferença entre μ_1 e μ_2 através do teste *t*. É testado então a hipótese nula, H_0: $\mu_D = 0$, contra H_1: $\mu_D \neq 0$. A hipótese nula H_0 declara que não há diferença~~s~~ significativa~~s~~ entre a médias dos valores das métricas obtidas pelos algoritmos. Já a hipótese alternativa H_1 anuncia a existência de diferença estatística entre os resultados. Dessa forma, aplicando o teste *t* é possível aceitar ou rejeitar H_0. Em consequência desta aceitação ou rejeição é possível concluir se os resultados obtidos pelas métricas efetivamente afirmam se algum método apresenta maior eficiência em solucionar o PFSP-MO.

No teste *t*, é aplicado a estatística de teste por *p-value* e assume-se um valor tolerável para erros, denominado nível de significância, α. O valor encontrado para *p-value* é então comparado com a tolerância α, sendo que se: $\alpha \geq$ *p-value*, H_0 é rejeitado.

Montgomery e Runger (2021) afirmam que a estatística de teste *t* é dado pela Equação (4.6).

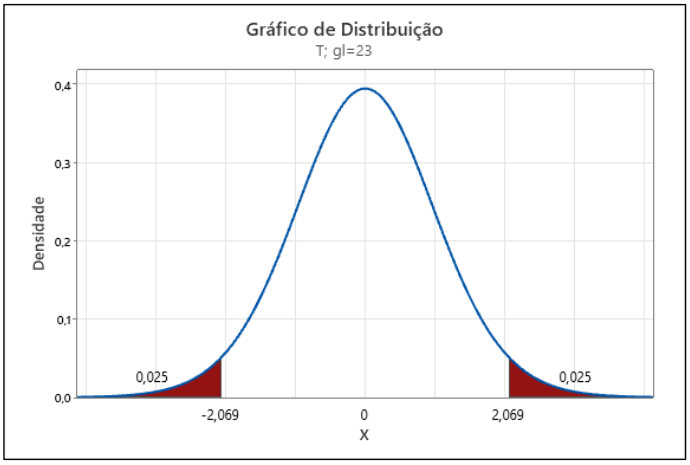
$$t_0 = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{4.6}$$

em que $\bar{d}$ é a média das diferenças entre as amostras, s_d se refere ao desvio padrão e n é a quantidade de amostras.

Para cada uma das métricas é realizada uma análise estatística, onde cada uma das populações, MOVNS e PILS, possuem um conjunto amostral contabilizado em 24, já que as métricas de desempenho analisaram os resultados por dimensão de instância. Em todos os experimentos estatísticos considera-se nível de significância $\alpha = 0{,}05$. Tendo então nível de confiança de 95%.

Ao aplicar a análise estatística em cada uma das métrica, tem-se como parâmetro de interesse apurar a diferença média em relação à métrica ora analisada, ou seja, $\mu_D = \mu_1 - \mu_2 = 0$. Onde, a hipótese nula é definida como $H_0: \mu_D = 0$, já a hipótese afirmativa é descrita como $H_1: \mu_D \neq 0$. Para verificar se H_0 deve ser aceita ou rejeitada é realizado o cálculo do *p-value*, utilizando o Minitab®, e este é comparado ao α. Outra forma de analisar a aceitação de H_0 é verificando em qual área do gráfico de distribuição *t* o valor da estatística de teste t_0, Equação (4.6), está inserido. A Figura 4.15 apresenta o gráfico de distribuição *t* com 23 graus de liberdade, $g = n - 1$. Caso o valor de t_0 pertença a área crítica, área sombreada de vermelho, H_0 é rejeitada. Mas se $-2{,}069 \leq t_0 \leq 2{,}069$, H_0 é então aceita.

Figura 4.15: Gráfico de distribuição *t*



Fonte: Minitab®

Na Tabela 4.9 é apresentado os resultados apurados através da análise estatística para cada uma das métricas aplicadas. Sendo apresentado: a média das diferenças $\bar{d}$, o desvio padrão s_d, o valor de *p-value*, t_0 e a conclusão de aceitação ou rejeição de H_0.

Tabela 4.9: Resultado da análise estatística

Métrica	$\bar{d}$	s_d	*p-value*	t_0	Aceitar ou Rejeitar H_0
Número de soluções na FP	182	37,44	0,00	23,81	Rejeitar
Medida de cardinalidade	273,7	80,7	0,00	16,63	
Distância média	-7,216	2,941	0,00	-12,02	
Distância máxima	-21,39	11,68	0,00	-8,97	
Taxa de erro	-62,02	20,49	0,00	-14,83	
Diferença de hipervolume	$-3{,}98.10^{13}$	$8{,}15.10^{13}$	0,025	-2,39	

Fonte: Minitab@

Ao aplicar o experimento estatístico é possível verificar que para todas as métricas de avalição de desempenho é verificado que *p-value* < α que todos os valores calculados para t_0 pertencem à área crítica na Figura 4.15. Por consequência, em todas as métricas analisadas, os experimentos estatísticos indicam que a hipótese nula H_0 deve ser rejeitada. Havendo assim, evidências suficientes para afirmar a existência de diferença média entre o MOVNS e PILS em todas as métricas. À vista disso, é comprovado tanto pelas métricas quanto pela análise estatística destas que o método MOVNS se apresenta mais eficiente em resolver o PFSP-MO quando comparado ao PILS.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho trata do problema *flow shop* permutacional multiobjetivo e estocástico (PFSP-MO), variante pouco encontrada na literatura. O objetivo deste problema é determinar um conjunto de sequenciamentos de *jobs* que minimizem simultaneamente os valores esperados para o atraso total, o *makespan* e a antecipação. Devido ao fato de tratar-se de um problema de otimização multiobjetivo, com objetivos conflitantes, não é verificado apenas uma solução, mas sim um conjunto de boas soluções. A qualidade das soluções encontradas foi avaliada segundo o critério de Pareto, onde as melhores soluções compõem a Fronteira Pareto e são ditas não dominadas ou soluções Pareto-ótimas.

Após a realização de estudo da literatura é identificada uma abordagem, desenvolvida nos últimos anos, capaz de resolver problemas multiobjetivos e estocásticos, denominada simheurística. Abordagem essa, resultado da hibridização entre simulação e técnicas de otimização.

Inicialmente, para melhor compreender o PFSP-MO, um modelo matemático foi adaptado e testado em um conjunto de 80 instâncias adaptadas da literatura. As instâncias contam com dados determinísticos e dimensão máxima de 25 *jobs* e 20 máquinas. Para resolução do modelo foi aplicado o método clássico de solução para problemas de otimização multiobjetivo denominado ponderação dos objetivos. 30 combinações de pesos foram definidas, buscando desde soluções extremas a soluções medianas.

Na fase de modelagem, foi verificado que o tempo de execução da instância está diretamente associado ao número de *jobs*. Em instâncias com maior quantidade de *jobs* o otimizador tem grande dificuldade em apresentar as soluções em baixo tempo computacional, tornando inviável a solução do PFSP-MO por este método.

Buscando resolver o PFSP-MO em menor tempo computacional e utilizando parâmetros estocásticos foram desenvolvidas duas abordagens simheurísticas: MOVNS e PILS. Ambos os algoritmos foram desenvolvidos a fim de buscar soluções não-dominadas para o PFSP-MO estocástico. Tanto no MOVNS quanto PILS, a solução inicial é gerada pelo método de solução do GRASP.

Os algoritmos MOVNS e PILS foram executados em um conjunto de 120 instâncias, adaptadas da literatura, com dimensão mínima de 20 *jobs* e 10 máquinas e dimensão máxima de 100 *jobs* e 20 máquinas. Instâncias com dimensão inferior a 20 *jobs* e 10 máquinas não foram

testadas pois o método de ponderação dos objetivos é capaz de encontrar soluções não-dominadas para elas em um tempo aceitável. Em todas as instâncias, os métodos propostos foram capazes de gerar um conjunto de soluções não-dominadas em um tempo computacional aceitável.

A comparação entre os resultados obtidos pelo MOVNS e pelo PILS foi feita através da aplicação de cinco métricas de avaliação de desempenho: número de soluções na FP, medida de cardinalidade, medidas de distância média e máxima, taxa de erro e diferença de hipervolume. Com base nos resultados apurados pelas métricas, percebeu-se que o algoritmo MOVNS apresenta os melhores valores em todas as instâncias, possuindo então desempenho superior ao PILS.

Experimentos estatísticos, baseados no teste *t*, foram realizados utilizando os resultados obtidos pelas métricas de avaliação. A análise estatística foi aplicada para cada uma das métricas de avaliação, onde pôde-se concluir que há significativa diferença entre o MOVNS e o PILS em relação a todas as métricas de avaliação.

Para o desenvolvimento deste trabalho, a ausência de instâncias estocásticas disponíveis na literatura é apontada como principal fator limitante. Haja visto que, devido a essa carência, foi necessário adaptar instâncias determinísticas para fins de utilização. Em consequência, não há outros trabalhos que abordam o problema em estudo, impossibilitando a comparação dos métodos propostos com outros da literatura.

Como propostas para trabalhos futuros, sugere a implementação de simheurística que combinem simulação e metaheurísticas de busca populacional utilizando-se algoritmos Genéticos, por exemplo, para resolver este mesmo problema. Outras sugestões, são aplicação dos métodos MOVNS e PILS propostos em outros tipos de problemas de sequenciamento descritos na seção 2.4, e inclusão de tempos de *setup* estocásticos no problema. Outros métodos e critérios também podem ser aplicados para gerar uma solução inicial mais diversificada, além da possibilidade de proporcionar maior tempo para execução dos métodos.

REFERÊNCIAS

Arenales, M.; Armentano, V.; Morabito, R.; Yanasse, H. (2015) *Pesquisa Operacional*. 2. ed. Rio de Janeiro: Elsevier: ABEPRO.

Arroyo, J. E. C. (2002) *Heurísticas e meta-heurísticas para otimização combinatória multiobjetivo*. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Estadual de Campinas. Campinas, SP. p. 227.

Arroyo, J. E. C., Ottoni, R. S., Oliveira, A. P. (2011) Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Eletronic Notes in Theoretical Computer Science*, 281: 5-19. DOI: 10.1016/j.entcs.2011.11.022

Arroyo, J. E. C., Pereira, A. A. S. (2011) A GRASP heuristic for the multi-objective permutacion flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 55: 741-753. DOI: 10.1007/s00170-010-3100-x

Arroyo, J. E. C.; Armentano, V. A. (2005) Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, vol. 167, 717-738. DOI: 10.1016/j.ejor.2004.07.017.

Baesler, F. F.; Sepulveda, J. A. (2000) Multi-response simulation optimization using stochastic genetic search within a goal programming framework. *In: Proceedings of the 2000 winter simulation conference*, 788–4. DOI: 10.1109/WSC.2000.899865.

Byrne, M. D.; Hossain, M. M. (2005) Production planning: An improved hybrid approach. *International Journal Production Economics*, 93, 225–229. DOI: 10.1016/j.ijpe.2004.06.021

Calvet, L.; Wang, D.; Juan, A.; Bové, L. (2019) Solving the multidepot vehicle routing problem with limited depot capacity and stochastic demands. *International Transactions in Operational Research*, 26(2), 458-484. DOI:10.1111/itor.12560

Chankong, V. e Haimes, Y. (1983) *Multiobjective Decision Making -Theory and Methodology*. North-Holland, New York: North-Holland.

Chica, M., Juan Pérez, A. A., Cordon, O., e Kelton, D. (2017) *Why simheuristics? benefits, limitations, and best practices when combining metaheuristics with simulation*. Disponível em SSRN: https://ssrn.com/abstract=2919208 ou doi:10.2139/ssrn.2919208

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4), 568-581.

Deb, K. (2008) Introduction to evolutionary multiobjective optimization. *In:* J. Branke, K Deb, K. Miettinen, R. Słowiński (Eds.), *Multiobjective Optimization*. *Lecture Notes in Computer Science*, 5252. Berlin: Springer.

Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A. M. T. (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, April 2002, DOI: 10.1109/4235.996017.

Dengiz, B.; Alabas, C. (2000) Simulation optimization using tabu search. *In: Proceedings of the 2000 winter simulation conference*, 805–810. DOI: 10.1109/WSC.2000.899877

Denton, B. T.; Rahman, A. S.; Nelson, H.; Bailey, A. C. (2006) Simulation of a multiple operating room surgical suite. *In: Proceedings of the 2000 winter simulation conference*, 414–424.

Gansterer, M.; Almeder, C.; Hartl, RF. (2014) Simulation-based optimization methods for setting production planning parameters. *International Journal Production Economics*, 151, 206–213. DOI: 10.1016/j.ijpe.2013.10.016.

Geiger, M. J. (2004) Randomized variable neighborhood search for multi objective optimization, 4th EU/ME*: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, 34-42.

Geiger, M. J. (2006) Foundations of the Pareto iterated local search metaheuristic. In: *Proceedings of the 18th MCDM*. Chania, Greece; June.

Ghorpade, T.; Corlu, C. G. (2020) Selective Pick-up and Delivery Problem: A Simheuristic Approach. *Winter Simulation Conference*, 1468-1479. DOI: 10.1109/WSC48552.2020.9384060.

Gomes Jr, A. C.; Carvalho, C. R. V. de; Munhoz P. L. A.; Souza, M. J. F. (2007) Um método híbrido para a resolução do problema de sequenciamento de uma máquina com penalidade por antecipação e atraso da produção. Anais... *XXXIX Simpósio Brasileiro de Pesquisa Operacional*, Fortaleza, Ceará.

Gomes, H. C.; Neves, A. N.; Souza, M. J. F. (2014) Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. *Computers & Operations Research*, 44: 92-104. DOI: 10.1016/j.cor.2013.11.002

González-Neira, E. M.; Ferone, D.; Hatami, S.; Juan, A. A. (2017a) A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 79: 23-36. DOI: 10.1016/j.simpat.2017.09.001

González-Neira, E. M.; Montoya-Torres, J. R. (2019a) A simheuristic for bi-objective stochastic permutation flow shop scheduling problem. *Journal os Project Management*, 4: 57-80. DOI: 10.5267/j.jpm.2019.1.003

González-Neira, E. M.; Montoya-Torres, J. R.; Barrera, D. (2017b) Flow-shop scheduling problem under uncertainties: review and trens. *International Journal of Industrial Engineering Computations*, 8: 399-426. DOI: 10.5267/j.ijiec.2017.2.001

González-Neira, E. M.; Urrego-Torres, A. M.; Cruz-Riveros, A. M.; Henao-Gracía, C.; Montoya-Torres, J. R.; Molina-Sánchez, L. P.; Jiménez, J-F. (2019b) Robust solutions in multi-objective stochastic permutation flow shop problem. *Computers & Industrial Engineering*, 137: 106026. DOI: 10.1016/j.cie.2019.106026

Groër, C., Golden, B., & Wasil, E. (2010) A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2), 79-101. DOI: 10.1007/s12532-010-0013-5

Hatami, S.; Calvet, L.; Fernández-Viagas, V.; Framiñán, J. M.; Juan, A. A. (2018) A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simulation Modelling Practice and Theory*, 86: 55-71.

Horn, J. (1997) *Handbook of Evolutionary Computation*, volume 1. Oxford University Press, Oxford, England.

Juan, A. A.; Barrios, B. B.; Vallada, E.; Riera, D.; Jorba, J. (2014) A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 46: 101-117. DOI: 10.1016/j.simpat.2014.02.005

Juan, A. A.; Faulin, J.; Grasman, S. E.; Rabe, M.; Figueira, G. (2015) A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives* 2, 62-72. DOI: 10.1016/j.orp.2015.03.001

Komaki, G. M.; Sheikh, S.; Malakooti, B. (2018) Flow shop scheduling problems with assembly operations: a review and new trends. *International Journal of Production Research*. DOI: 10.1080/00207543.2018.1550269

Latorre-Biel, J. I.; Ferone, D.; Juan, A. A.; Faulin, J. (2021) Combining simheuristics with petri nets for solving the stochastic vehicle routing problem with correlated demands. *Expert Systems with Applications*, 168. DOI:10.1016/j.eswa.2020.114240

Liefooghe, A.; Humeau, J.; Mesmoudi, S.; Jourdan, L.; Talbi, E-G. (2012) On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics* 18, 317–352. DOI: 10.1007/s10732-011-9181-3

Liu, F.; Wang, S.; Hong, Y.; Yue, X. (2017) On the Robust and Stable Flowshop Scheduling Under Stochastic and Dynamic Disruptions. *IEEE Transactions on Engineering Management*, 64(4), 539-553. DOI: 10.1109/TEM.2017.2712611

Lourenço H.R., Martin O.C., Stützle T. (2003) Iterated Local Search. In: Glover F., Kochenberger G.A. (eds) *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol 57. Springer, Boston, MA. DOI: 10.1007/0-306-48056-5_11

Lustosa, L. J.; Mesquita, M. A.; Quelhas, O. L. G.; Oliveira, R. J. (2008) *Planejamento e controle da produção.* Rio de Janeiro: Elsevier.

Marković, D.; Petrović, G.; Ćojbašić, Ž.; Stanković, A. (2020) The vehicle routing problem with stochastic demands in an urban area – a case study. *Mechanical Engineering* 18, 107-120. DOI: 10.22190/FUME190318021M

Miettinen, K. (2008) Introduction to evolutionary multiobjective optimization. *In:* J. Branke, K Deb, K. Miettinen, R. Słowiński (Eds.), *Multiobjective Optimization*. *Lecture Notes in Computer Science*, 5252. Berlin: Springer.

Mishra, A. K., Shrivastava D., Bundela B., Sircar S. (2020) An Efficient Jaya Algorithm for Multi-objective Permutation Flow Shop Scheduling Problem. In: Venkata Rao R., Taler J. (eds) Advanced Engineering Optimization Through Intelligent Techniques. *Advances in Intelligent Systems and Computing*, 949: 113-125. Springer, Singapore. DOI: 10.1007/978-981-13-8196-6_11

Montgomery, D. C.; Runger, G. C. (2021) *Estatística Aplicada e Probabilidade para Engenheiros*. Rio de Janeiro: Grupo GEN: LTC.

Nouri, N.; Ladhari, T. (2018) Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking. *Annals of Operations Research* 267, 413–430. DOI: 10.1007/s10479-017-2465-8

Pan, Q-K.; Gao, L.; Wang, L.; Liang, J.; Li, X-Y. (2019) Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert systems with applications*, 124: 309-324. DOI: 10.1016/j.eswa.2019.01.062.

Pinedo, M. L. (2016) *Scheduling: theory, algorithms, and systems*. 5 ed. New York: Springer. doi:10.1007/978-3-319-26580-3_9

Ramesh, N.; Dickerson, T. (2022) "Just-in-time to Just-in-case,". IEEE Engineering Management Review. DOI: 10.1109/EMR.2022.3147594.

Su L, Qi Y, Jin L-L and Zhang G-L. (2016) Integrated batch planning optimization based on fuzzy genetic and constraint satisfaction for steel production. *International Journal of Simulation Modelling*, vol. 15(1), pp. 133-143. doi: 10.2507/IJSIMM15(1)CO1

Sun, Y.; Zhang, C.; Gao, L.; Wang, X. (2011) Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. *The International Journal of Advanced Manufacturing Technology* 55, 723–739. DOI: 10.1007/s00170-010-3094-4

Ta, Q.; Billaut, J-C.; Bouquard, J-L. (2018) Matheuristic algorithms for minimizing total tardiness in the m-machines flow-shop scheduling problem. *Journal os Inteligente Manufacturing*, 29:617-628. DOI: 10.1007/s10845-015-1046-4

Ta, Q.; Billaut, J-C.; Bouquard, J-L. (2018) Metaheuristic algorithms for minimizing total tardiness in the m-machines flow-shop scheduling problem. *Journal of Inteligente Manufacturin*g, 29:617-628. DOI: 10.1007/s10845-015-1046-4

Taillard, E. (1993) Benchmarks for basic scheduling problems. *European Journal Operational Research*. 64, 278–285. DOI: 10.1016/0377-2217(93)90182-M

Tasgetiren, M. F., Oztop, H., Pan, Q. -K., Ornek, M. A., Temizceri, T. (2021) A Variable Block Insertion Heuristic for the Energy-Efficient Permutation Flowshop Scheduling with Makespan Criterion. In: Yalaoui F., Amodeo L., Talbi EG. (eds) Heuristics for Optimization and Learning. *Studies in Computational Intelligence*, 906: 33-49. Springer, Cham. DOI: 10.1007/978-3-030-58930-1_3

Vallada, E.; Ruiz, R.; Framinan, J. M. (2015) New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal Operational Research*, 240: 666-677. DOI: 10.1016/j.ejor.2014.07.033

Veldhuizen, D. (1999) *Multi-objective Evolutionary Algorithms: Classifications, Analyses and New Innovations*. Tese de Doutorado, Escola de Pós-Graduação em Engenharia do Instituto de Tecnologia da Força Aérea.

Villarinho, P.; Pinto, D.; Lila, M.; Panadero, J.; Cyrino, F.; Pessoa, L. (2019) Proposta de uma abordagem híbrida de simulação e heurística para um problema de *flowshop* estocástico. *LI Simpósio Brasileiro de Pesquisa Operacional*, Limeira. *Anais...* Campinas, Galoá. https://proceedings.science/p/106899?lang=pt-br

Xu, J., Chin-Chia, W., Yin, Y., Lin, W-C. (2017) An iterated local Search for the multi-objective permutation flowshop scheduling problem with sequence-dependet setup times. *Applied Soft Computing*, 52: 39-47. DOI: 10.1016/j.asoc.2016.11.031

Zitzler, E.; L. Thiele. (1998) Multi-objective Optimization Using Evolutionary Algorithms – a Comparative Case Study. In: Eiben A.E.; Bäck T.; Schoenauer M.; Schwefel HP. (eds) *Parallel*

Problem Solving from Nature — PPSN V. PPSN 1998. Lecture Notes in Computer Science, vol 1498. Springer, Berlin, Heidelberg. DOI: 10.1007/BFb0056872