Uma abordagem simheuristic para resolver o problema de flowshop permutacional multiobjetivo

An simheuristic approach to solve the multi-objective permutation flowshop problem

DOI:10.34117/bjdv8n1-374

Recebimento dos originais: 07/12/2021
Aceitação para publicação: 20/01/2022

Fernanda dos Reis Cota
Universidade Federal de Ouro Preto – UFOP
E-mail: fernanda.cota@aluna.ufop.edu.br

Naiara Helena Vieira
Universidade Federal de Ouro Preto – UFOP
E-mail: naiara.helena@aluno.ufop.edu.br

Aloisio de Castro Gomes Júnior
Universidade Federal de Ouro Preto – UFOP
E-mail: aloisio.junior@ufop.edu.br

Helton Cristiano Gomes
Universidade Federal de Ouro Preto – UFOP
E-mail: helton.gomes@ufop.edu.br

Allexandre Fortes
Universidade Federal de São João Del-Rei – UFSJ
E-mail: afortes@ufsj.edu.br

Área: 3 – Pesquisa Operacional
Sub-Área: 3.1 – Modelagem, Simulação e Otimização

RESUMO
O problema de flowshop permutacional (pfsp) é amplamente estudado por ter vasta aplicação em problemas teóricos e reais. Neste artigo foi abordado a variante estocástica e multiobjetivo do pfsp, sendo os possíveis atrasos ocorridos durante o processo produtivo. Para tanto, é proposta uma abordagem híbrida que combina a versão multiobjetivo da meta-heurística variable neighborhood search (movns) e simulação, a fim de minimizar os valores esperados para makespan e atraso total. Um conjunto de soluções iniciais é gerado pela heurística neh e novas soluções não-dominadas são produzidas a partir de três estruturas de vizinhança: 1-point-move, 2-point-move e 3-point-move, e armazenadas em uma fronteira pareto. Após a aplicação do movns, um número de replicações é realizado modificando os parâmetros estocásticos obtendo as esperanças dos valores do makespan e atraso total. Os resultados computacionais mostram que o movns precisa de ajustes para permitir a obtenção mais consistente de soluções não-dominadas.

Palavras-chaves: Flowshop, Simheuristicas, Multiobjetivo, Variable Neighborhood Search, Simulação.

ABSTRACT
The permutational flowshop problem (pfsp) is widely studied because it has wide application in theoretical and real problems. In this article, the stochastic and multi-objective variant of pfsp was approached, with possible delays occurring during the production process. To solve, a hybrid approach is proposed that combines the multi-objective version of the variable neighborhood seach (movns) metaheuristic and simulation to minimize the expected values for makespan and total delay. A set of initial solutions is generated by the neh heuristic and new non-dominated solutions are produced from three neighborhood structures: 1-point-move, 2-point-move and 3-point-move and stored in a pareto frontier. After the application of movns, several replications are carried out by modifying the stochastic parameters obtaining the expected values of the makespan values and total delay. The computational results show that movns needs adjustments to allow more consistent obtaining of non-dominated solutions.

Keywords: Flowshop, Simheuristics, Multi-Objective, Variable Neighborhood Search, Simulation.

1 INTRODUÇÃO

O problema de sequenciamento visa determinar a sequência ideal, em um determinado período, para processamento de um conjunto de atividades em um conjunto de recursos limitados a fim de otimizar um ou mais objetivos previamente estabelecidos (GONZÁLEZ-NEIRA et al., 2017a). O problema de sequenciamento pode ser classificado de acordo com a configuração do ambiente produtivo como problema de máquina única, máquinas paralelas, flowshop, job shop e open shop (PINEDO, 2016).

Um dos problemas de sequenciamento que tem sido bastante estudado nas últimas décadas e podem ser encontrados em diferentes setores industriais é o problema flowshop (PFS) (GONZÁLEZ-NEIRA et al., 2017b). O problema flowshop consiste no processamento de n tarefas em todas as m máquinas dispostas no ambiente produtivo. Ao assumir que as tarefas devem ser processadas na mesma sequência em todas as máquinas, surge uma variante do PFS conhecida como problema flowshop permutacional (PFSP) (GONZÁLEZ-NEIRA et al., 2017a).

A maioria das contribuições literárias ao PFSP prevê a otimização de um único objetivo. Os principais objetivos que podem ser verificados são: (i) minimização do instante de conclusão da última atividade na última máquina (também conhecido como makespan) (VALLADA et al., 2015), (ii) minimização do tempo de processamento total (PAN et al., 2019) e (iii) minimização do atraso total (TA et al., 2018). No entanto, os tomadores de decisão lidam diariamente não apenas com um objetivo, mas com vários que devem ser otimizados simultaneamente, sendo estes objetivos usualmente conflitantes (ARROYO et al., 2011a).

Então, devido a necessidade de buscar melhores resultados em mais de uma métrica

simultaneamente, surge o problema do flowshop permutacional multiobjetivo (PFSP-MO). Em sua maioria, o PFSP-MO visa a otimização de duas ou mais funções objetivo (KOMAKI et al., 2018). Os objetivos de minimização podem ser compostos por variadas métricas. Por exemplo, Arroyo et al. (2011b) 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. Já Xu et al. (2017) buscam a minimização do makespan e atraso total ponderado. Gomes et al. (2014) por sua vez objetivam a minimização do makespan e tempo total ponderado de início das atividades. 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 e minimização do consumo de energia total, combinação rara na literatura.

O PFSP-MO pode ser caracterizado também como modelo determinístico ou estocástico. Ao assumir que todas as informações referentes ao PFSP-MO, como tempos de processamento, data de entrega, tempo de setup, e outras, são previamente conhecidas e não possuem variação, o problema é dito determinístico (PINEDO, 2016). No entanto, em problemas reais estas informações estão sujeitas a incertezas, tais como quebra ou indisponibilidade de máquina, falta de energia, atraso de matéria prima e outras, que são incontroláveis e podem tornar ineficaz a decisão determinística (GONZÁLEZ-NEIRA et al., 2017b).

Diante disto, parâmetros estocásticos têm sido cada vez mais utilizados para solucionar o PFSP-MO. González-Neira et al. (2017a), Hatami et al. (2018) e Juan et al. (2014), por exemplo, utilizam tempos de processamento estocásticos. Já Allahverdi e Allahverdi (2018) representam os tempos de setup como parâmetro incerto e Liu et al. (2017) utilizam as possíveis quebras de máquina. Mesmo diante do aumento da pesquisa abordando parâmetros estocásticos nos últimos anos, o acervo literário da versão determinística do problema ainda é mais extenso quantitativamente (GONZÁLEZ-NEIRA et al., 2017a).

O PFSP-MO com dados estocásticos pode ser entendido como uma generalização do problema, em que determinada variável não possui valor constante, mas sim um valor aleatório não-negativo usualmente gerado a partir de uma distribuição de probabilidade, por exemplo, Normal, Log-Normal, Exponencial, Gamma, etc. (JUAN et al., 2014). Dessa forma, as incertezas do cenário real são melhores representadas.

A aleatoriedade é introduzida no PFSP-MO com o intuito de descrever problemas mais realistas, em que os dados não são previamente conhecidos. Para solução destes problemas estocásticos, foram desenvolvidas, nas últimas décadas, métodos eficientes que combinam simulação e otimização, as chamadas Simheurísticas (Juan et al., 2014). Os autores afirmam

que a hibridização entre simulação e otimização, incluindo utilização de meta-heurísticas, é uma abordagem promissora para solução de problemas de otimização combinatória estocásticos, como mostrado em González-Neira et al. (2017a), González-Neira e Montoya-Torres (2019), Hatami et al. (2018) e Juan et al. (2014).

As meta-heurísticas são métodos aplicados originalmente em problemas de otimização com apenas uma função objetivo, mas devido à sua eficácia estas têm sido utilizadas também para problemas multiobjetivo (ARROYO et al., 2011a). Recorrentemente, abordagens simheurísticas aplicam as meta-heurísticas GRASP (GONZÁLEZ-NEIRA; MONTOYA-TORRES, 2019) e Iterated Local Search (ILS) (HATAMI et al., 2018) (JUAN et al., 2014) para solução do problema PFSP-MO.

Dentre estes métodos meta-heurísticos a abordagem Variable Neighborhood Search (VNS), proposta por Mladenovic e Hansen (1997) para problemas combinatórios mono-objetivo tem recebido destaque, sendo o trabalho de Geiger (2004) o primeiro a utilizar o VNS multiobjetivo para o problema de flowshop permutacional. O VNS tem como princípio encontrar soluções melhores por meio de trocas sistemáticas baseadas em estruturas de vizinhança, partindo de uma solução inicial.

Desta forma, este artigo tem como objetivo apresentar uma abordagem simheurística combinando a meta-heurística VNS com simulação para resolver o problema de flowshop permutacional multiobjetivo de maneira a contribuir com a literatura. As funções de avaliação, conflitantes entre si, adotadas são a minimização do makespan e a minimização do atraso total. O restante deste artigo é organizado como se segue. A Seção 2 apresenta a descrição do problema PFSP-MO abordado neste estudo. A Seção 3 apresenta o algoritmo VNS proposto. Na Seção 4 os experimentos computacionais são descritos e apresentados. Conclusões e oportunidades para futuras pesquisas são expressas na Seção 5.

2 DESCRIÇÃO DO PROBLEMA

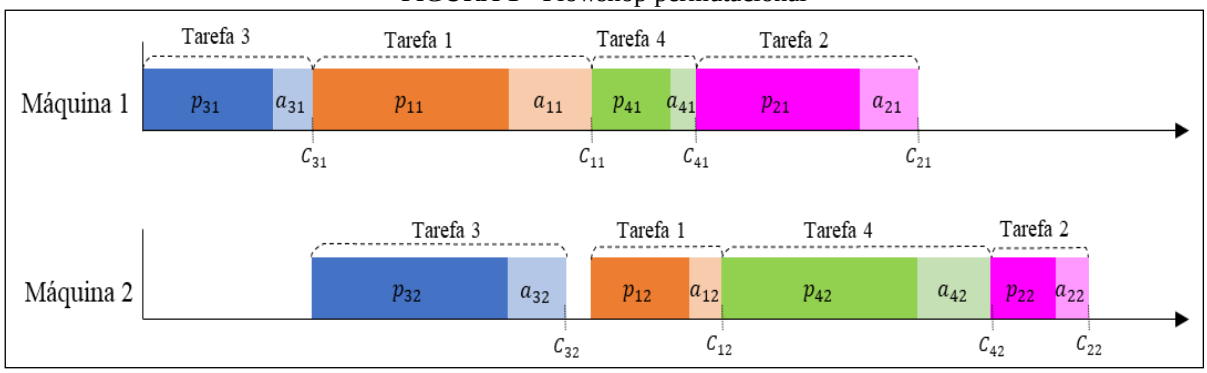
O problema flowshop permutacional multiobjetivo abordado possui um conjunto de jobs J={1, 2, ..., n} os quais devem ser processados em um conjunto de máquinas M={1, 2, ..., m}. Sendo estes jobs processados na mesma ordem em todas as máquinas. Cada job $j \in J$ possui um tempo de processamento p_{ji} previamente conhecido para cada máquina $i \in M$. Assume-se que todas os jobs e máquinas estão disponíveis no instante zero e quando iniciam seus respectivos processamentos, estes não podem ser interrompidos.

Uma tarefa não pode ser processada por mais de uma máquina simultaneamente, da mesma forma que cada máquina deve processar apenas uma tarefa por vez. Um novo

processamento ocorre em uma máquina i quando: (i) a atividade j tenha seu processamento finalizado na máquina i-1 e (ii) a máquina i esteja disponível.

Para representar possíveis atrasos ocorridos durante o processamento das tarefas um parâmetro estocástico a_{ji} é gerado e adicionado ao tempo de processamento p_{ji}, compondo assim o fluxo total. A Figura 1 ilustra as informações apresentadas com um exemplo contendo duas máquinas e quatro tarefas.

FIGURA 1 - Flowshop permutacional



Fonte: Elaborado pelos autores (2021)

Cada tarefa $j \in J$ possui uma data entrega $_{vj}$. Se o instante do término de processamento da tarefa j na última máquina (C_j) for menor que a data de entrega v_j, ou seja, $C_j < v_j$, não há atraso na tarefa j ($T_j = 0$). O atraso T_j é calculado pela diferença entre o instante de conclusão da tarefa e sua respectiva data de entrega: $T_j = max\{0, C_j - v_j\}$.

O objetivo do problema é determinar sequenciamentos de tarefas viáveis com mínimo makespan (f_1) e mínimo atraso total (f_2) (ARROYO et al., 2011a). As funções f_1 e f_2 são então calculadas pelas equações (1) e (2).

$$f_1 = min\ C_j \tag{1}$$

$$f_2 = min \sum_{j=1}^{n} T_j \tag{2}$$

A função f_1 tem como objetivo a minimização do makespan da tarefa j, portanto, para chegar-se a este resultado o instante de finalização do processamento da tarefa j em cada máquina deve ser calculado. Conforme apresentado na Figura 1, períodos de ociosidade nas máquinas podem ser verificados durante o processamento. Esta situação ocorre devido a possibilidade de a máquina i + 1 estar livre para processar a tarefa j, mas esta ainda não tenha finalizado seu processamento pela máquina i. De forma similar, a tarefa j pode ter finalizado seu processamento na máquina i, estando disponível para ser processada na máquina i + 1,

porém esta pode estar ainda processando a tarefa j - 1. Dessa forma, pode haver situações em que a tarefa aguardará para ser processada ou a máquina aguardará para processar.

Em problemas de otimização onde há mais de uma função objetivo geralmente não há uma única solução que otimize ambos os objetivos simultaneamente. Dessa forma, em problemas multiobjetivo a otimalidade das soluções encontradas é verificada através da dominância de Pareto (ARROYO *et al*., 2011a). Os autores acrescentam que a dominância de Pareto consiste na construção do conjunto de soluções Pareto-ótimo, as quais compõem a Fronteira Pareto (FP), sendo esta composta por todos os vetores de solução que são não-dominados por qualquer outro vetor solução.

A dominância de Pareto compara duas soluções e verifica se estas são dominadas entre si ou não. Xu *et al*. (2017) afirmam que uma solução X domina uma solução Y se e somente se $f_i(X) \leq f_i(Y)$ para todo objetivo i, havendo ao menos um objetivo i com $f_i(X) < f_i(Y)$. Assim, a solução X é inserida na FP se e somente se X não é dominado por qualquer outra solução pertencente à FP.

3 ALGORITMO *SIMHEURISTICA*

Com o propósito de desenvolver uma abordagem simheurística para o problema flowshop permutacional, este trabalho apresenta um algoritmo baseado na meta-heurística VNS que considera simultaneamente o caráter multiobjetivo e de incerteza do problema. O método é descrito no Algoritmo 1 que inicia com a leitura das características das instâncias, linha 1, identificando a quantidade de atividades (n), a quantidade de máquinas (m) e seus respectivos tempos de processamento (p_{ji}). Na linha 2 é realizada uma pequena simulação, em que são gerados valores estocásticos para as datas de entrega (v_j) e possíveis atrasos de cada atividade em cada máquina (a_{ji}). Em seguida, o algoritmo NEH (Seção 3.1), utilizando estas informações constrói duas soluções iniciais. Estas soluções são adicionadas a fronteira Pareto (linha 4). O laço entre as linhas 5 e 9 é executado por um número iterMax vezes e onde o algoritmo MOVNS (Seção 3.2) encontra novas soluções não-dominadas que serão adicionadas a fronteira Pareto. Por fim, uma grande simulação (linha 10) é realizada por um número definido de replicações em que novos valores para v e a são gerados e aplicados às soluções Pareto-ótima, assim encontrando novos valores esperados para as funções objetivo do problema.

ALGORITMO 1 - Sim-MOVNS

```
   Entrada: instância
1  (n, m, p) ← leitura(instância);
2  (v, a) ← Estocastico();
3  {s1, s2} ← NEH(n, m, p, v, a,);
4  FP ← {s1, s2};
5  para iter = 1, ..., iterMax faça
6  |   enquanto Critério de parada não é satisfeito faça
7  |   |   FP ← FP ∪ MOVNS(n, m, p, v, a, FP);
8  |   fim
9  fim
10 E(Makespan), E(Atraso) ← Simulação(FP);
11 retorna FP, E(Makespan), E(Atraso);
```

Fonte: Elaborado pelos autores (2021)

3.1 OBTENÇÃO DAS SOLUÇÕES INICIAIS

As soluções iniciais para o PFSP-MO implementado neste artigo foram geradas a partir da heurística NEH, apresentada por Nawaz *et al*., 1983. Esta heurística é conhecida por sua simplicidade e eficácia na construção de soluções iniciais para o PFSP (XU *et al*., 2017).

Neste trabalho a heurística NEH gera duas soluções iniciais sendo a primeira utilizando o critério de ordem decrescente da soma entre tempo de processamento p_{ji} e parâmetro de atraso estocástico a_{ji}, e a segunda partindo do critério de ordem crescente da data de entrega v_j. Ambas as soluções são adicionadas ao conjunto de soluções iniciais não dominadas entre si.

3.2 VNS MULTIOBJETIVO

Este artigo propõe um método VNS multiobjetivo (MOVNS) para resolver o PFSP-MO. Seu pseudocódigo é descrito no Algoritmo 2 e executa as seguintes atividades enquanto o critério de parada não é satisfeito, sendo que para este trabalho foi adotado o tempo limite de um minuto. Na linha 2 uma solução s ainda não explorada é selecionada dentre as pertencentes a fronteira Pareto (FP). Esta solução é então marcada como explorada e um vizinho s' é gerado a partir de s de acordo com uma estrutura de vizinhança V_k aleatoriamente sorteada (linhas 3 a 5). Se s' não é dominada por nenhuma das soluções que pertencem a FP, então s' é adicionada ao conjunto de soluções não dominadas.

Em seguida, uma outra estrutura V_l $(l \neq k)$ é sorteada e uma nova solução s'' é gerada a partir de s' e se s'' não é dominada por nenhuma das soluções que pertencem a FP, então s'' é adicionada ao conjunto de soluções não dominadas (linhas 6 a 9). Por fim, se todas as soluções já foram exploradas, elas são desmarcadas para que possam ser exploradas nas iterações futuras (linha 10).

ALGORITMO 2 - VNS multiobjetivo (MOVNS)

Entrada: $n, m, \mathbf{p}, \mathbf{v}, \mathbf{a}, FP, \mathcal{V}$
1 **enquanto** *Critério de parada não é satisfeito* **faça**
2 Seleciona aleatoriamente uma solução não explorada $s \in FP$;
3 Marque s como explorada;
4 Selecione aleatoriamente uma vizinhança $V_k, k \in \mathcal{V}$;
5 Gere um vizinho qualquer $s' \leftarrow V_k(s, n, m, \mathbf{p}, \mathbf{v}, \mathbf{a})$;
6 **se** s' é não-dominada **então** $FP \leftarrow FP \cup \{s'\}$;
7 Selecione aleatoriamente uma vizinhança $V_l, l \in \mathcal{V}, l \neq k$;
8 Gere um vizinho qualquer $s'' \leftarrow V_l(s', n, m, \mathbf{p}, \mathbf{v}, \mathbf{a})$;
9 **se** s'' é não-dominada **então** $FP \leftarrow FP \cup \{s''\}$;
10 **se** toda solução $s \in FP$ foi explorada **então** marque todas como não explorada;
11 **fim**
12 **retorna** FP;

Fonte: Elaborado pelos autores (2021)

O PFSP-MO abordado neste artigo utiliza três estruturas de vizinhança: 1-point-move, 2-point-move e 3-point-move (GROËR et al., 2010). O movimento 1-point-move, também conhecido como inserção, seleciona uma tarefa j e realiza sua inserção em uma nova posição no vetor solução. A estrutura de vizinhança 2-point-move, ou troca, realiza de forma simultânea a troca de posição entre duas tarefas na sequência. Por fim, o movimento 3-point-move cria uma solução vizinha a partir da troca de posição entre dois tarefas consecutivos com a posição de um terceiro tarefa. Quando alguma destas vizinhanças são selecionadas, elas realizam n/s movimentos sobre a solução que está sendo atualmente explorada.

3.3 SIMULAÇÃO

A última etapa do algoritmo proposto é a realização de uma grande simulação. Nela as soluções que participam da fronteira Pareto são analisadas considerando um determinado número de replicações (N_{rep}). A cada replicação, linha 3, novos valores estocásticos para as datas de entrega (v_j) e possíveis atrasos de cada atividade em cada máquina (a_{ji}) são gerados. Estes novos valores são aplicados a cada uma das soluções da fronteira Pareto (linhas 4 a 7) de maneira em que se determine o valor esperado do makespan e do atraso total (linha 9).

ALGORITMO 3 – Simulação

Entrada: Fronteira Pareto (FP)
1 $FP' \leftarrow \emptyset$;
2 **para** $r = 1, \ldots, N_{rep}$ **faça**
3 $(\mathbf{v}, \mathbf{a}) \leftarrow$ Estocastico();
4 **para todo** $s \in FP$ **faça**
5 $s' \leftarrow s(\mathbf{v}, \mathbf{a})$;
6 $FP' \leftarrow FP' \cup \{s'\}$;
7 **fim**
8 **fim**
9 $E(Makespan), E(Atraso) \leftarrow$ Esperança(FP');
10 **retorna** $E(Makespan), E(Atraso)$;

Fonte: Elaborado pelos autores (2021)

4. INSTÂNCIAS E AMBIENTE COMPUTACIONAL

4.1 INSTÂNCIAS TESTE

Para verificar o desempenho do algoritmo proposto, foram testadas um conjunto de 20 instâncias (disponíveis para consulta em http://soa.iti.es/problem-instances), apresentadas por Taillard (1993). Cada instância é composta por problemas com 10, 20, 30, 40, 50 tarefas e 5, 10, 15 e 20 máquinas, além dos tempos de processamento de cada tarefa em cada máquina.

Para este problema foram gerados os seguintes parâmetros estocásticos: a_{ji} representando o possível atraso ao se processar cada tarefa *j* em cada máquina *i* seguindo uma distribuição normal de média 3 e desvio padrão de 0,5. Este parâmetro é contabilizado em conjunto com os tempos de processamento para determinação do *makespan* de cada tarefa.

A data de entrega v_j é calculada a partir da equação $v_j = (P_j + S_j) * (1 + random * 3)$ (XU *et al.*, 2017), onde P_j representa a soma dos tempos de processamento da tarefa *j* em todas as máquinas $P_j = \sum_{i=1}^{m} \quad p_{ji}$; S_j representa a soma de todos os possíveis atrasos $S_j = \sum_{i=1}^{m} \quad a_{ji}$, e *random* é um valor uniformemente distribuído no intervalo [0, 1].

4.2 CONFIGURAÇÃO DOS PARÂMETROS

O algoritmo foi desenvolvido e executado usando a linguagem C/C++ em um computador com sistema operacional Windows 10 equipado com um processador Intel Core i5 e 8 GB de memória RAM. Foram executadas 5 rodadas independentes, utilizando-se o tempo como semente para o gerador de números aleatórios. O parâmetro *iterMax* do Algoritmo 1 (linha 6) foi igual a 5. O critério de parada da linha 1 do Algoritmo 2 foi 60 segundos de execução e o número de replicações (N_{rep}) do Algoritmo 3 foi igual a 100. Com isso, uma pequena simulação e uma grande simulação foram realizadas compostas por 5 e 100 execuções. Ao final dos experimentos foram armazenadas as respectivas soluções que pertencem a fronteira Pareto e os valores esperados do *makespan* e atraso total, e as análises são discutidas na próxima seção.

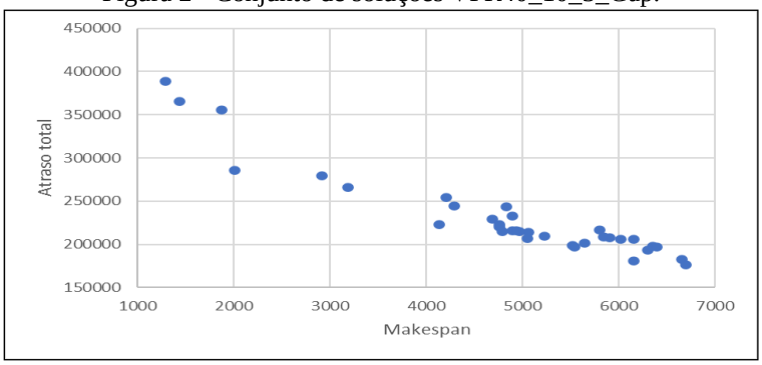
4.3 ANÁLISE DOS RESULTADOS

O algoritmo proposto apresentou uma combinação entre a meta-heurística VNS e simulação para resolver o PFSP-MO. O propósito seria oferecer diferentes cenários (soluções) para o tomador de decisão considerando objetivos conflitantes, de forma que ele possa escolher o que lhe for mais conveniente, além da expectativa (esperança) do valor das funções objetivos de minimizar o *makespan* e atraso total.

Para verificar a qualidade das soluções obtidas de cada instância foi realizada uma

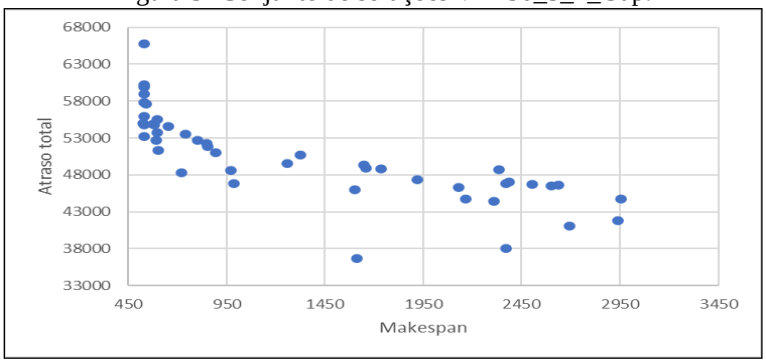
análise avaliando se o método foi capaz de obter um bom conjunto de soluções não dominadas entre si. De maneira geral, o método Sim-MOVNS teve dificuldades em formar a fronteira de soluções eficientes (Pareto-ótima) para as instâncias analisadas. Como exemplos, os resultados para as os problemas VFR40_10_3_Gap e VFR30_5_1_Gap foram ilustrados nas Figura 2 e Figura 3, respectivamente. Isto sugere que melhorias devem ser realizadas no algoritmo para que ele alcance uma maior aderência aos objetivos do problema.

Figura 2 - Conjunto de soluções VFR40_10_3_Gap.



Fonte: Elaborado pelos autores (2021)

Figura 3- Conjunto de soluções VFR30_5_1_Gap.



Fonte: Elaborado pelos autores (2021)

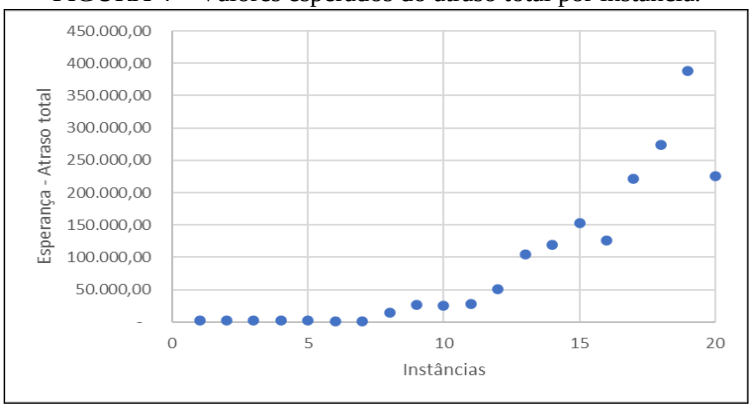
A Tabela 1 mostra os valores esperados médios do makespan e do atraso total após realizadas cinco rodadas independentes. É possível notar que para alguns problemas com cinco máquinas, os valores médios obtidos para o makespan foram discrepantes das demais instâncias. Isso sugere que ao se considerar a minimização do makespan, o problema se torna mais difícil de se sequenciar com a diminuição do número de máquinas. Ao observar os valores esperados para o atraso total, conforme mostrado na Figura 4, é possível notar que com o aumento do número de tarefas ocorre uma tendência exponencial (linha pontilhada) do atraso total, o que evidencia o aumento do grau de dificuldade de resolução do problema. Dessa maneira, se o tomador de decisão necessitasse de priorizar um dos objetivos, ele poderia considerar o makespan, pois este apresenta uma menor média, se comparado ao atraso total.

Tabela 1 – Valores esperados do makespan e do atraso total

	Esperança	
Instância	**Makespan**	**Atraso Total**
VFR10_5_1_Gap	569.369,00	2.475,51
VFR10_5_2_Gap	705,89	2.520,08
VFR10_5_3_Gap	284.007,42	2.626,50
VFR10_5_4_Gap	614,29	1.701,58
VFR10_10_5_Gap	1.305,41	2.525,09
VFR10_15_6_Gap	1.616,06	949,23
VFR10_20_8_Gap	1.943,25	758,49
VFR20_5_4_Gap	543.488,51	14.902,76
VFR20_10_7_Gap	2.025,70	26.632,96
VFR20_15_3_Gap	3.069,75	25.404,38
VFR20_20_10_Gap	4.008,93	28.125,72
VFR30_5_1_Gap	177.415,61	50.572,56
VFR30_10_9_Gap	2.534,98	104.348,10
VFR30_15_1_Gap	5.294,54	118.710,20
VFR30_20_9_Gap	7.598,72	153.113,80
VFR40_5_7_Gap	1.785,84	126.188,60
VFR40_10_3_Gap	5.112,89	220.940,40
VFR40_15_6_Gap	8.555,49	273.913,40
VFR40_20_5_Gap	11.930,70	388.166,20
VFR50_5_8_Gap	1.906,28	225.135,20
Média	81.714,46	88.485,54

Fonte: Elaborado pelos autores (2021)

FIGURA 4 – Valores esperados do atraso total por instância.



Fonte: Elaborado pelos autores (2021)

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho abordou o problema de flowshop permutacional multiobjetivo (PFSP-MO), o qual oferece enormes desafios teóricos e computacionais dado a sua ampla aplicabilidade a diversos problemas reais, como exemplo o sequenciamento de atividades e projetos ou o planejamento da produção. Dessa forma, diferentes abordagens ainda podem oferecer grandes contribuições para resolução do problema.

Como forma de contribuir com este tema, o presente estudo abordou o uso de um

algoritmo híbrido, classificado como simheurística, que combina otimização multiobjetivo utilizando uma meta-heurística e técnicas de simulação. Este algoritmo tratou dois objetivos, a minimização do makespan e do atraso total. Para isto, a meta-heurística aplicada foi a Variable Neighborhood Search (VNS) que é baseada em trocas sistemáticas de estruturas de vizinhança. Diferentes soluções foram obtidas através das estruturas de vizinhança escolhidas (1-point-move, 2-point-move e 3-point-move) e estas soluções foram comparadas aplicando-se o conceito de Pareto a fim de formar a fronteira Pareto. Após cada execução da meta-heurística, uma simulação é aplicada a fim de apurar as esperanças dos valores das funções objetivo escolhidas. Os resultados mostraram que melhorias devem ser realizadas no algoritmo MOVNS com o objetivo de se obter um conjunto de soluções Pareto-ótimo mais consistentes e bem definido.

Como propostas para estudos futuros fica a implementação de uma fase de intensificação (JUAN et al., 2014) e de uma busca em profundidade variável baseada em Pareto (Pareto-based variable depth search) (XU et al., 2017), além de outro método multiobjetivo para permitir a utilização de métricas de avaliação e comparação. Para se obter o melhor desempenho dos algoritmos que serão desenvolvidos, e para calibrar os parâmetros do problema utilizar a ferramenta irace (LÓPEZ-IBÁNEZ et al., 2016).

REFERÊNCIAS

ALLAHVERDI, A.; ALLAHVERDI, M. **Two-machine no-wait flowshop scheduling problem with uncertain setup times to minimiza maximum lateness**. Computational and Applied Mathematics, 37: 6774-6794, 2018. DOI: 10.1007/s40314-018-0694-3

ARROYO, J. E. C., OTTONI, R. S., OLIVEIRA, A. P. **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, 2011a. DOI: 10.1016/j.entcs.2011.11.022

ARROYO, J. E. C., PEREIRA, A. A. S. **A GRASP heuristic for the multi-objective permutacion flowshop scheduling problem**. Int J Adv Manuf Technol, 55: 741-753, 2011b. DOI: 10.1007/s00170-010-3100-x

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

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

GONZÁLEZ-NEIRA, E. M.; FERONE, D.; HATAMI, S.; JUAN, A. A. **A baised-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times**. Simulation Modelling Practice and Theory, 79: 23-36, 2017a. DOI: 10.1016/j.simpat.2017.09.001

GONZÁLEZ-NEIRA, E. M.; MONTOYA-TORRES, J. R. **A simheuristic for bi-objective stochastic permutation flowshop scheduling problem**. Journal os Project Management, 4: 57-80, 2019. DOI: 10.5267/j.jpm.2019.1.003

GONZÁLEZ-NEIRA, E. M.; MONTOYA-TORRES, J. R.; BARRERA, D. **Flow-shop scheduling problem under uncertainties: review and trens**. International Journal of Industrial Engineering Computations, 8: 399-426, 2017b. DOI: 10.5267/j.ijiec.2017.2.001

GROËR, C., GOLDEN, B., & WASIL, E.**A library of local search heuristics for the vehicle routing problem**. Mathematical Programming Computation, 2(2), 79-101, 2010.

HATAMI, S.; CALVET, L.; FERNÁNDEZ-VIAGAS, V.; FRAMIÑÁN, J. M.; JUAN, A. A. **A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem.** Simulation Modelling Practice and Theory, 86: 55-71, 2018.

JUAN, A. A.; BARRIOS, B. B.; VALLADA, E.; RIERA, D.; JORBA, J. **A simheuristic algorithm for solving the permutation flowshop problem with stochastic processing times**. Simulation Modelling Practice and Theory, 46: 101-117, 2014. DOI: 10.1016/j.simpat.2014.02.005

KOMAKI, G. M.; SHEIKH, S.; MALAKOOTI, B. **Flow shop scheduling problems with assembly operations: a review and new trends**. International Journal of Production Research, 2018. DOI: 10.1080/00207543.2018.1550269

LIU, F.; WANG, S.; HONG, Y.; YUE, X. **On the robust and stable flowshop scheduling under stochastic and dynamic disruptions**. IEEE transactions on engineering management, 64: 539-553, 2017. DOI: 10.1109/TEM.2017.2712611

LÓPEZ-IBÁNEZ, M., DUBOIS-LACOSTE, J., CÁCERES, L. P., BIRATTARI, M., & STÜTZLE, T. **The irace package: Iterated racing for automatic algorithm configuration**. Operations Research Perspectives, 3, 43-58, 2016.

MISHRA, A. K., SHRIVASTAVA D., BUNDELA B., SIRCAR S. **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, 2020. DOI: 10.1007/978-981-13-8196-6_11

MLADENOVIĆ, N., & HANSEN, P. **Variable neighborhood search**. Computers & operations research, 24(11), 1097-1100, 1997.

NAWAZ, M.; ENSCORE, E. E., HAM I. **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**. Omega, 11: 91-95, 1983. DOI: 10.1016/0305-0483(83)90088-9

PAN, Q-K.; GAO, L.; WANG, L.; LIANG, J.; LI, X-Y. **Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem.** Expert systems with applications, 124: 309-324, 2019. DOI: 10.1016/j.eswa.2019.01.062

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

TA, Q.; BILLAUT, J-C.; BOUQUARD, J-L. **Metaheuristic algorithms for minimizing total tardiness in the m-machines flow-shop scheduling problem**. Journal os Inteligente Manufacturing, 29:617-628, 2018. DOI: 10.1007/s10845-015-1046-4

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

TASGETIREN, M. F., OZTOP, H., PAN, Q. -K., ORNEK, M. A., TEMIZCERI, T. **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, 2021. DOI: 10.1007/978-3-030-58930-1_3

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

XU, J., CHIN-CHIA, W., YIN, Y., LIN, W-C. **An iterated local Search for the multi-objective permutation flowshop scheduling problem with sequence-dependet setup times**. Applied Soft Computing, 52: 39-47, 2017. DOI: 10.1016/j.asoc.2016.11.031