Meta-heurísticas aplicadas ao problema de sequenciamento de tarefas com prazos de entrega distintos em múltiplas máquinas - Piic/UFES

Edital:	**Edital Piic 2023/2024.**
Grande Área do Conhecimento (CNPq):	Ciências Exatas e da Terra.
Área do Conhecimento (CNPq):	Ciência da Computação.
Título do Projeto:	Métodos Computacionais em Otimização.
Título do Subprojeto:	Meta-heurísticas aplicadas ao problema de sequenciamento de tarefas com prazos de entrega distintos em múltiplas máquinas
Professor Orientador:	Oberlan Christo Romão.
Estudante:	Marcos Vinícius Vargas Mello.

Resumo

Neste trabalho, heurísticas e meta-heurísticas são aplicadas ao problema de sequenciamento de tarefas com prazos de entrega distintos em múltiplas máquinas. Neste tema, o foco está em encontrar uma sequência de tarefas adequadas que vise minimizar os atrasos e adiantamentos das tarefas desempenhadas nestes ambientes. Porém, por esse ser um problema pertencente à classe NP-Difícil, existe uma dificuldade em encontrar soluções ótimas por meio de modelos matemáticos. Em consequência disso, métodos heurísticos e meta-heurísticos baseados nos algoritmos NEH, genético e de busca tabu serão apresentados como alternativa. Por fim, esses foram testados e comparados com as soluções apresentadas pelo modelo matemático desenvolvido e apresentaram resultados positivos e competitivos.

Palavras-chave: Otimização, sequenciamento de tarefas, prazos de entrega distintos, meta-heurísticas, algoritmo genético, busca local.

1 Introdução

Um dos grandes desafios das indústrias na atualidade encontra-se na minimização das perdas que ocorrem quando se há atraso ou adiantamento no término da fabricação de produtos em relação a sua data de entrega. Essas perdas se apresentam, quando em atrasos, através de penalidades contratuais, perda de clientes e credibilidade (Sen & Gupta, 1984; S. Sakuraba et al., 2009). Por sua vez, quando em adiantamentos, as perdas surgem por meio dos custos com áreas como galpões, almoxarifados e salas de estoque que não são necessárias nos casos onde não há a estocagem de produtos (Brown & Rieche, 2006).

Diante dessa situação, se originou, na década de 1950, a abordagem de gestão *Just In Time* (JIT), que busca, entre outros objetivos, "tornar disponível a peça certa, no momento certo e na quantidade certa

para entrar na montagem" (Ohno, 1982). Sendo assim, uma filosofia caracterizada pela busca contínua por maneiras de aumentar a eficiência dos processos e redução dos desperdícios.

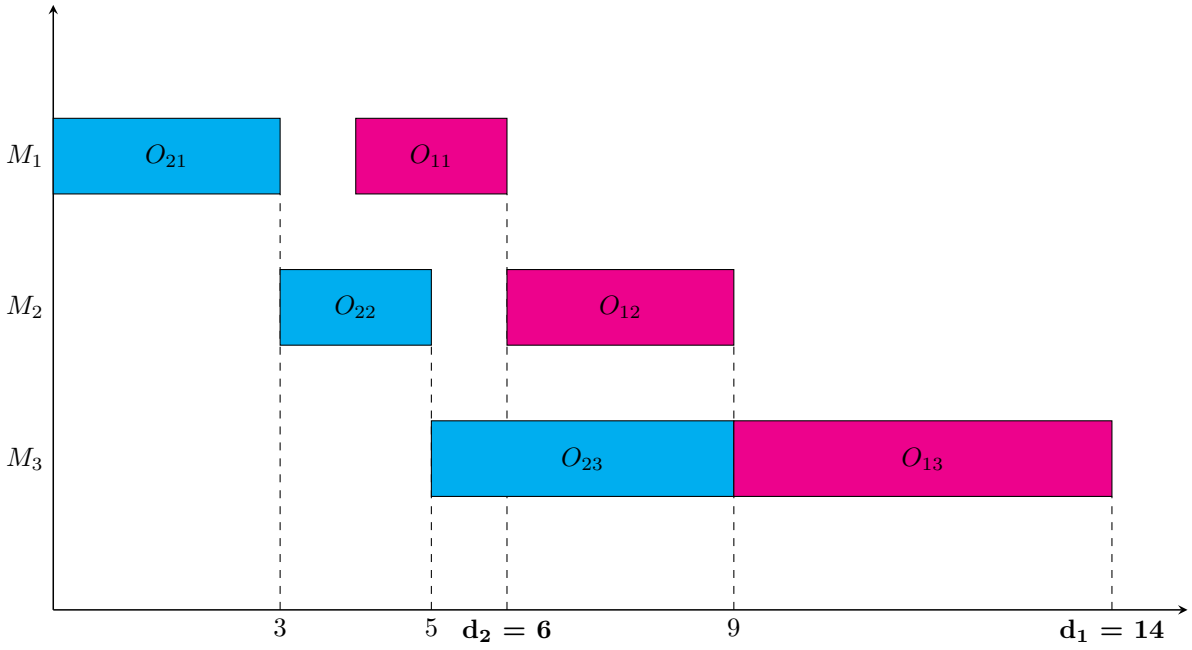
A partir disso, tem-se que esse desafio e filosofia podem ser observados em ambientes *flow shop*, uma vez que esses têm a capacidade de serem modelados como linhas de produção, que consistem em um conjunto de m máquinas em série que processam uma quantidade j de produtos (referidos também como tarefas, ou pelo termo *jobs* da língua inglesa) em uma sequência fixa a fim de serem entregues o mais próximo possível de suas respectivas datas de entrega. Para isso, por conta de possibilitar a procura por cenários de datas de entrega compatíveis com o problema proposto neste trabalho, será utilizado o critério de otimalidade, advindo da metodologia JIT, da *soma de desvios absolutos* (SDA), ou *soma de adiantamentos e atrasos total* (Vasquez & Birgin, 2017).

Consoante a isso, de maneira formal, esse ambiente pode ser descrito por meio de uma sequência fixa de m máquinas, representada por $M = (M_1, M_2, ..., M_m)$, e um conjunto de n tarefas que precisam ser processadas nessas m máquinas, representada por $J = (J_1, J_2, ..., J_n)$. Além disso, tem-se que cada tarefa J_i possui uma data de entrega d_i e é composta por m operações $(O_{i1}, O_{i2}, ..., O_{im})$, onde a k-ésima entrada dessa, representada por O_{ik}, denota, obrigatoriamente, que a tarefa está sendo processada, durante um período de tempo p_{ik}, pela máquina M_k. Ademais, se uma tarefa i é finalizada antes da data de entrega d_i, essa é considerada adiantada, e, caso finalizada após d_i, ela é considerada atrasada. Outrossim, o objetivo nesse ambiente é a minimização da SDA. Finalmente, é importante salientar que as operações de cada tarefa seguem uma ordem específica, passando pelas mesmas máquinas na mesma ordem.

Posto isso, este trabalho possui algumas restrições que precisam ser consideradas na solução do problema proposto: A operação O_{ik+1} só pode ser iniciada após a operação O_{ik} ser finalizada; Cada máquina pode processar apenas uma operação de cada vez, da mesma forma que uma tarefa só pode ser processada por uma máquina por vez; Todas as máquinas estão sempre em pleno funcionamento sem necessidade de interrupções ou tempo de preparo; Todas as informações do problema são conhecidas de forma determinística, ou seja, são previamente estabelecidas.

Como exemplo, tem-se a Figura (1) que ilustra uma possível solução, demonstrando a distribuição das tarefas em cada máquina com relação ao tempo. Neste exemplo, tem-se que $m = 3$ máquinas e $M = (M_1, M_2, M_3)$, $n = 2$ tarefas e $J = (J_1, J_2)$, $J_1 = (O_{11}, O_{12}, O_{13})$, $J_2 = (O_{21}, O_{22}, O_{23})$, $p_1 = (p_{11}, p_{12}, p_{13}) = (2, 3, 5)$, $p_2 = (p_{21}, p_{22}, p_{23}) = (3, 2, 4)$ e com $d_1 = 14$ e $d_2 = 6$. Com isso, é possível notar que a SDA desta instancia é 3, uma vez que a tarefa J_2 foi concluída no instante 9, estando 3 unidades de tempo em atraso, enquanto J_1 foi concluída no instante 14, finalizada assim em sua data de entrega.

Figura 1: Distribuição de tarefas em um ambiente *flow shop* com 2 tarefas, 3 máquinas e data de entrega d1 = 14 e d2 = 6.



Fonte: Produção do próprio autor.

Por fim, esse desafio é classificado como um problema do tipo de sequenciamento de tarefas, que, apesar de aparentar simplicidade, segundo Hall et al. (1991), o ambiente *flow shop* permutacional com SDA é provado ser NP-difícil. Em decorrência disto, obter soluções exatas, em tempo razoável, para instâncias medianas ou grandes é pouco provável. Logo, faz-se necessário propor métodos alternativos aos modelos matemáticos para que essa limitação possa ser superada.

2 Objetivos

Diante deste cenário, o principal objetivo deste trabalho é propor heurísticas e meta-heurísticas para o problema de sequenciamento de tarefas com datas de entrega distintas em múltiplas máquinas a fim de que seja possível encontrar, em tempo razoável, soluções que sejam de ótima ou boa qualidade. Neste problema, o desafio está em encontrar uma sequência ótima de execução de tarefas, que visa cumprir com o maior número de prazos de entrega possíveis, otimizando assim a eficiência do processo. Para isso, será utilizada uma abordagem exata, a fim de possibilitar a comparação de resultados, e uma algorítmica que buscará sequências adequadas por meio da exploração de soluções evolutivas.

3 Embasamento Teórico

Embora a literatura sobre programação de tarefas em ambientes *flow shop* permutacional seja bastante extensa, os trabalhos que abordam múltiplas máquinas com datas de entrega distintas são limitados, apesar de ser uma prática comum na indústria. O trabalho de Vasquez & Birgin (2017) aborda o

problema programação de tarefas com mais de duas máquinas, cujo objetivo minimizar a soma total dos adiantamentos e atrasos das tarefas, por meio do critério SDA, em relação a uma data de entrega em comum. Além disso, Vasquez & Birgin (2017) também propõe adaptações na heurística proposta em Nawaz et al. (1983) para poder utilizá-la com múltiplas máquinas.

Nos estudos de Reeves (1995) e de Murata et al. (1996) são propostos algoritmos genéticos, para resolver o problema de programação *flow shop* com múltiplas máquinas e objetivo de minimizar o tempo de conclusão total, conhecido como *makespan*. Além disso, Murata et al. (1996) também propõe o uso de algoritmos genéticos híbridos e realiza simulações a fim de evidenciar os operadores de *crossover* e mutação que melhor se encaixam para a resolução desses problemas. Esses diferem-se deste trabalho pois não possuem restrições de datas de entrega e, como consequência, utilizam do critério de otimalidade makespan.

4 Metodologia

Uma vez que o foco deste trabalho está em sequenciamento de tarefas com datas de entrega distintas, houveram adaptações e a adição de novas restrições ao modelo proposto por Vasquez & Birgin (2017). A partir destas adaptações, tem-se a implementação deste modelo por meio do *solver* IBM ILOG CPLEX tanto em linguagem OPL, quanto em C++. Sobre a implementação em C++, essa ocorreu a fim de possibilitar testes mais eficientes, facilitar a comparação de resultados e possibilitar a geração de tabelas de forma automatizada. Também, possibilitou a padronização do formato de entrada dos casos de testes, tornando mais fácil a criação de novas instancias de testes.

4.1 Modelo Matemático

Após adaptações do modelo apresentado por Vasquez & Birgin (2017), houve a implementação deste modelo matemático para solucionar o problema. Perante a isso, tem-se na Tabela 1 os parâmetros de entrada, enquanto na Tabela 2, tem-se as variáveis de decisão.

Tabela 1: Parâmetros de entrada do modelo matemático proposto.

Parâmetro	**Descrição**
n	Número de tarefas.
m	Número de máquinas.
p_{ik}	Tempo de processamento da tarefa J_i $(1 \leq i \leq n)$ na máquina M_k $(1 \leq k \leq m)$.
d_i	Data de entrega da tarefa J_i $(1 \leq i \leq n)$.

Fonte: Vasquez & Birgin (2017) - Adaptado

Tabela 2: Variáveis de decisão do modelo matemático proposto.

Variável	**Descrição**
X_{ij}	Variável binária cujo valor é 1 se a tarefa J_i é atribuída à j-ésima posição da sequência, 0 em caso contrário.
E_j, T_j	Variáveis inteiras indicando, respectivamente, o adiantamento e o atraso da j-ésima tarefa.
C_j	Variável inteira representando o tempo de conclusão da j-ésima tarefa na última máquina.
W_{jk}	Variável inteira indicando o tempo de espera (no *buffer*) da j-ésima tarefa entre as máquinas M_k e M_{k+1}.
I_{jk}	Variável inteira indicando o tempo de inatividade entre as tarefas j e a $(j+1)$-ésima na máquina M_k.
I_0	Variável inteira representando o tempo de inatividade da primeira máquina até processar a primeira tarefa.

Fonte: Vasquez & Birgin (2017)

Assim, o modelo matemático para o problema tratado neste trabalho consiste em:

$$\text{Minimize } \sum_{i=1}^{n} (E_i + T_i) \tag{1}$$

sujeito à

$$C_1 = I_0 + \sum_{k=1}^{m} \left(\sum_{i=1}^{n} X_{i1} p_{ik} \right) + \sum_{i=1}^{m-1} W_{1i} \tag{2}$$

$$C_j = C_{j-1} + I_{j-1,m} + \sum_{i=1}^{n} X_{ij} p_{im} \qquad j = 2, \ldots, n \tag{3}$$

$$I_{jk} + \sum_{i=1}^{n} X_{i,j+1} p_{ik} + W_{j+1,k} = W_{jk} + \sum_{i=1}^{n} X_{ij} p_{i,k+1} + I_{j,k+1} \qquad j = 1, \ldots, n-1; k = 1, \ldots, m-1 \tag{4}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad j = 1, \ldots, n \tag{5}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad i = 1, \ldots, n \tag{6}$$

$$E_i \geq C_i - \sum_{j=1}^{n} X_{ji} d_j \qquad i = 1, \ldots, n \tag{7}$$

$$T_i \geq \sum_{j=1}^{n} X_{ji} d_j - C_i \qquad i = 1, \ldots, n \tag{8}$$

A função objetivo (1) busca minimizar a soma dos adiantamentos e atrasos de todas as tarefas, fazendo com que cada uma termine o mais próximo da sua respectiva data de entrega. As restrições (2) e (3) garantem que o tempo em que cada tarefa termina é igual ao somatório de seus respectivos tempos em cada máquina junto dos tempos de espera no *buffer* de cada máquina. As restrições (4) garantem que a soma dos tempos de inatividade, processamento e espera para cada par de tarefas consecutivas deve ser a mesma. As restrições (5) e (6) asseguram que uma tarefa só pode estar em uma posição da ordem de processamento e só pode haver uma tarefa em cada posição da ordem de processamento. Por fim, as restrições (7) e (8) são responsáveis por garantir que o tempo de atraso e adiantamento irá ser maior ou igual que a diferença entre a data de entrega e o tempo de conclusão da tarefa.

4.2 Heurística

Considerando a dificuldade de se encontrar soluções ótimas pelo modelo matemático, como mostrado na seção de Resultados, é proposta uma heurística baseada na heurística NEH (Nawaz et al., 1983). A heurística é iniciada com dois dos melhores *jobs* que são selecionados utilizando critérios de ordenação próprios, Tabela 3, a fim de serem permutados e analisados, por meio de uma função de avaliação de custo, para que possam ter suas posições relativas fixadas nos passos subsequentes até o fim do processo. Após isso, esse continuará selecionando o próximo melhor *job* e inserindo esse nas lacunas disponíveis dos *jobs* fixados, até que finalize-se o número de *jobs* disponíveis (Nawaz et al., 1983). Nos casos de empate entre sequências parciais, apenas a primeira sequência será considerada.

Algoritmo 1: Algoritmo NEH

Saída: Uma sequência de tarefas J

1: **início**
2: Ordenar as tarefas conforme as regras de sequenciamento ordenadas da Tabela (3)
3: Selecionar as duas primeiras e aplicar a função de avaliação de custo às duas possíveis sequências de tarefas e salvar a melhor delas
4: **para** $k \leftarrow 3$ ***até*** n **faça**
5: Inserir a k-ésima tarefa na posição que minimize a função objetivo parcial dentre as k sequências possíveis
6: **fim**
7: **fim**

Tabela 3: Descrição das regras de sequenciamento para o procedimento NEH.

Regra	**Descrição**
1	Menor data de entrega
2	Maior tempo de processamento
3	Menor identificador

Fonte: Produção do próprio autor.

4.3 Meta-Heurística

Apesar da heurística NEH encontrar soluções de boa qualidade em instâncias pequenas e médias, percebeu-se que para instâncias de grande porte, ela não conseguia um bom resultado. Por isso, também é proposto um algoritmo genético (AG) que, em contextos de otimização de problemas de combinatória, é um algoritmo de seleção da melhor solução, neste caso, a melhor ordem de execução para minimizar a SDA.

Conforme (Reeves, 1995), o AG se inspira em conceitos da evolução biológica e seleção natural, que ocorre por meio da manutenção de uma população de s soluções, que possuem seus valores de aptidão calculados a fim de possibilitar a classificação destas em etapas posteriores. A partir disso, tem-se as

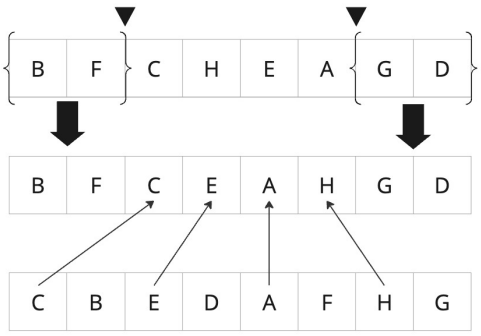
etapas que compõem o AG, sendo geralmente divididas em geração da população, seleção dos pais da próxima geração, cruzamento, mutação e repetição dos passos anteriores até que seja encontrada uma condição de parada.

Neste trabalho a geração da população é feita em primeiro momento com o uso do melhor indivíduo gerado pela heurística NEH e do preenchimento do restante da população de maneira aleatória e após esse, utilizando-se da seleção dos pais e dos operadores de cruzamento e mutação. Por sua vez, a seleção dos pais baseia-se em ranqueamento e nas probabilidades geradas por meio do modelo proposto por Reeves (1995), conforme na equação (9). Também, têm-se os operadores de cruzamento e mutação propostos por Murata et al. (1996), respectivamente, *Two-Point* (Figura 2) e *Arbitrary Two Job Change* (Figura 3).

Por fim, segundo Murata et al. (1996), o AG também pode ser modificado, por meio da adição de heurísticas em seus passos, para possibilitar um aumento em sua performance. Em concordância com isso, tem-se a Busca Tabu (BT), implementada segundo Murata et al. (1996), que é um tipo de busca local cujo papel é procurar por soluções próximas (vizinhas) através do melhor individuo do AG em momentos onde esse pode se encontrar em mínimos locais, a fim de conseguir resultados melhores que ainda não foram explorados.

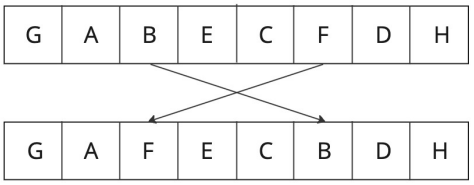
$$probabilidade(i) = \frac{2 \times (tamanhoPopulacao - i)}{tamanhoPopulacao \times (tamanhoPopulacao + 1)} \tag{9}$$

Figura 2: Representação do operador de cruzamento *Two Point*



Fonte: Produção do próprio autor.

Figura 3: Representação do operador de mutação *Arbitrary Two Job Change*.



Fonte: Produção do próprio autor.

5 Resultados e Discussão

Todos experimentos foram realizados utilizando o servidor de otimização da Universidade Federal do Espírito Santo configurado com processador Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz 10 núcleos (20 *threads*), 160Gb RAM, rodando em Sistema GNU/Linux Ubuntu Server. Adicionalmente, os algoritmos foram implementados em C++17 e compilados com o GNU g++ versão 11.4.0. Para o modelo

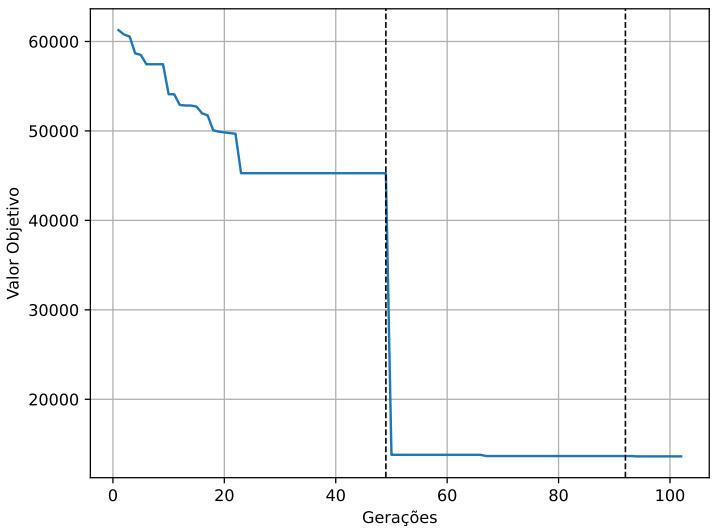
matemático, foi utilizado o IBM ILOG CPLEX versão 20.1, sendo implementado por meio do ILOG Concert Technology.

Para avaliar os modelos, as instâncias disponibilizadas por Vasquez & Birgin (2017) em `https://www.ime.usp.br/~jdelgado/flowshop/resultados/` serão utilizadas como base para a Tabela 4. Para a Tabela 5, uma vez que as instâncias de Vasquez & Birgin (2017) são limitadas a datas de entrega comum, essas foram geradas de maneira aleatória e podem ser acessadas em `https://github.com/MarcosMello/flowshop-ic`.

Outro ponto importante, é que o solver CPLEX foi configurado para entregar a melhor solução possível em um intervalo de, no máximo, 21.600 segundos (6 horas) utilizando-se de apenas uma *thread*, portanto execução sequencial. Além disso, empiricamente, o Algoritmo Genético (AG) foi configurado com uma população de $20 + \lfloor\sqrt{n}\rfloor$ indivíduos com limite de, no máximo, 100 gerações e, caso não ocorra nenhuma melhora entre elas, 50 gerações. Já a Busca Tabu (BT) foi configurada com tamanho máximo de 10 itens para a lista tabu e com a mesma quantidade máxima de gerações que o AG. Por fim, a composição entre NEH, AG e BT representa o algoritmo proposto (**AP**). Para esse, os resultados foram obtidos por meio de 30 execuções para cada instância a fim de possibilitar maior precisão estatística e confiabilidade nos dados.

A Busca Tabu foi implementada com a finalidade de melhorar a solução encontrada pelo AG. Este comportamento pode ser observado na Figura 4, onde as linhas verticais em preto indicam os pontos onde a busca tabu foi utilizada. Consoante a isso, é possível notar que no inicio deste gráfico, o AG desempenhou de forma satisfatória, porém a partir de sua 23ª geração, esse se manteve com resposta constante em um mínimo local por 25 gerações. Com isso, após a aplicação da técnica de BT, houve uma queda no valor da função objetivo e o AG voltou a desempenhar seu papel e conseguiu, na sua 66ª geração, encontrar mais uma solução possível para a instância antes de ocorrer o próximo uso de BT.

Figura 4: Representação da evolução dos resultados do algoritmo genético em suas gerações.



Fonte: Produção do próprio autor.

Para medir a eficiência, foi computado a variação percentual da solução obtida pelos algoritmos apresentados com relação ao valor de referência, que é o melhor resultado, sendo esse a solução ótima do CPLEX, quando não ultrapassado o limite de tempo, ou a solução do AP, quando contrário (Vasquez & Birgin, 2017). Esta, encontra-se representada pela equação (10).

$$\text{Variação Percentual (V\%)} = 100 \times \left(\frac{f_{metodo} - f_{referencia}}{f_{referencia}} \right) \tag{10}$$

A Tabela 4 e Tabela 5 apresentam os resultados obtidos com os experimentos numéricos, respectivamente, das instâncias com datas de entrega em comum e distintas. Esses resultados estão divididos em 4 subgrupos de instâncias baseado na quantidade de tarefas (indicada por n) e na quantidade de máquinas (indicada por m). Adicionalmente, esses grupos estão também subdivididos pela coluna **Inst.** que indica quais dados foram utilizados para a obtenção dos resultados.

Nestas tabelas, as duas primeiras colunas se referem a instância de teste. As duas próximas da Tabela 4 se referem ao valor da função objetivo e tempo (em segundos) obtidos pelo CPLEX. As quatro colunas restantes estão associadas ao algoritmo proposto: solução mediana dos resultados após as 30 execuções do AP; desvio padrão das soluções; tempo médio (em segundos) das execuções e; variação percentual. Na Tabela 5, além das colunas da Tabela 4, foi incluído, na solução do CPLEX, as colunas GAP% e V%, representando, respectivamente, a diferença percentual entre a melhor solução inteira encontrada até o momento e a solução relaxada do problema e a variação percentual.

Conforme Tabela 4, note que o CPLEX encontrou a solução ótima para todas instâncias com data de entrega em comum, uma vez que retornou resultados sem ter atingido seu tempo limite. Isso também pode ser observado através das medianas apresentadas como soluções pelo AP. Porém, é notável que, no geral, o tempo de resposta do CPLEX foi consideravelmente menor que o tempo de AP.

A Tabela 5 resume os resultados para as instâncias com data de entrega distintas, foco deste trabalho. Observe que o CPLEX encontrou as as soluções ótimas para todas instancias de n5m10 e n10m20, uma vez que retornou resultados sem ter atingido seu tempo limite. De modo semelhante, é possível perceber que o AP também entregou as soluções ótimas, com exceção da instancia n10m20_3, na qual obteve uma variação percentual de menos de 1%. Porém, a partir das instâncias médias (n50m20) e grandes (n80m20) presentes nesta tabela, o CPLEX já não foi capaz de entregar a solução ótima dentro de um período de 6 horas, tendo GAPs% variando entre 86% a 99%, indicando assim que muitas soluções ainda não haviam sido exploradas. Em contrapartida, o AP conseguiu fornecer soluções melhores, que não necessariamente são ótimas, para todos esses casos gastando menos de 3 minutos. Com isso, justifica-se a seleção destes algoritmos, assim como afirmado em Murata et al. (1996). Além disso, com base nesses exemplos, é possível afirmar que em situações reais que, geralmente, possuem instâncias ainda maiores e com mais restrições, é impraticável tentar resolvê-las por meio do modelo matemático.

Tabela 4: Resultados obtidos para as instâncias com data de entrega comum.

Nome	Inst.	CPLEX		AP			
		Solução	Tempo (seg)	Solução	SD	Tempo Médio (seg)	V%
n5m10	1	173	0.01	**173**	-	0.02	-
	2	302	0.03	**302**	-	0.01	-
	3	245	0.01	**245**	-	0.02	-
n10m20	1	1472	0.10	**1472**	-	0.11	-
	2	636	0.12	**636**	-	0.25	-
	3	599	0.09	**599**	-	0.25	-
n50m20	1	22500	8.25	**22500**	0.61	68.90	-
	2	18523	4.52	**18523**	3.32	77.83	-
	3	18684	5.96	**18684**	-	87.49	-
n80m20	1	55492	14.73	**55492**	-	413.23	-
	2	48399	15.42	**48399**	0.17	442.22	-
	3	51951	17.85	**51951**	0.49	393.25	-

Fonte: Produção do próprio autor.

Tabela 5: Resultados obtidos para as instâncias com data de entrega distintas.

Nome	Inst.	CPLEX				AP			
		Solução	Tempo (seg)	GAP%	V%	Solução	SD	Tempo Médio (seg)	V%
n5m10	1	1312	0.06	-	-	**1312**	-	0.01	-
	2	1213	0.06	-	-	**1213**	-	0.01	-
	3	1788	0.03	-	-	**1788**	-	0.01	-
n10m20	1	7922	15.64	-	-	**7922**	-	0.06	-
	2	6975	3.07	-	-	**6975**	-	0.04	-
	3	7408	22.48	-	-	**7478**	21.00	0.06	0.94
n50m20	1	27780	21600.00	89.84	106.67	**13442**	101.92	12.97	-
	2	17840	21600.00	86.83	89.87	**9396**	66.97	27.10	-
	3	34084	21600.00	97.01	64.10	**20770**	410.19	8.90	-
n80m20	1	62862	21600.00	97.22	256.30	**17643**	149.28	66.98	-
	2	83998	21600.00	99.47	155.61	**32862**	486.98	60.39	-
	3	46651	21600.00	97.17	213.94	**14860**	251.81	142.17	-

Fonte: Produção do próprio autor.

Tabela 6: Resultados do algoritmo genético obtidos para as instâncias com data de entrega distintas.

Nome	Inst.	AP			
		Melhor Solução	Pior Solução	Mediana	SD
n5m10	1	1312	1312	1312	-
	2	1213	1213	1213	-
	3	1788	1788	1788	-
n10m20	1	7922	7922	7922	-
	2	6975	6975	6975	-
	3	7408	7478	7478	21.00
n50m20	1	12946	13442	13442	101.92
	2	9194	9439	9396	66.97
	3	19890	21430	20770	410.19
n80m20	1	17284	18224	17643	149.28
	2	32123	34333	32862	486.98
	3	14775	15583	14860	251.81

Fonte: Produção do próprio autor.

6 Conclusões

Neste trabalho, foi estudado heurísticas e meta-heurísticas aplicadas ao problema de sequenciamento de tarefas com datas de entrega distintas em múltiplas máquinas com objetivo de minimizar a soma total dos adiantamentos e atrasos das tarefas. Para isso, em primeiro lugar foi apresentado um modelo matemático utilizado como referência para os algoritmos propostos posteriormente. Porém, por conta da dificuldade de resolver este problema, pertencente à classe NP-difícil, para casos médios ou grandes, faz-se necessário de outras alternativas de resolução que não apenas por meio do modelo matemático. Para isso, foi proposto um algoritmo que faz uso de heurísticas (NEH) e meta-heurísticas (algoritmo genético e busca tabu). Esse foi implementado e testado, apresentando resultados positivos e demonstrando competitividade na busca de soluções de boa qualidade em tempo reduzido quando comparado ao modelo matemático. Por fim, como trabalhos futuros, pode-se haver novos estudos sobre outros métodos heurísticos e meta-heurísticos, assim como a inclusão de novas restrições ao problema.

7 Agradecimentos

O presente projeto foi desenvolvido com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) através da concessão de bolsa de Iniciação Científica - IC, nº de processo 143015/2023-0. Outrossim, este também contou com o apoio da Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) através da utilização do servidor de otimização adquirido por meio do processo FAPES 116/2019.

Referências Bibliográficas

Brown, S. & Rieche, A. (2006). Administração da produção e operações: um enfoque estratégico na manufatura e nos serviços. *Elsevier*.

Hall, N., Kubiak, W., & Sethi, S. (1991). Earliness-tardiness scheduling problems, ii: Deviation of completion times about a restrictive common due date. *Operations Research*, 39.

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, 30(4):1061–1071.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

Ohno, T. (1982). How the toyota production system was created. *Japanese Economy*, 10(4):83–101.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, 22(1):5–13. Genetic Algorithms.

S. Sakuraba, C., Ronconi, D. P., & Sourd, F. (2009). Scheduling in a two-machine flowshop for the minimization of the mean absolute deviation from a common due date. *Computers & Operations Research*, 36(1):60–72. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

Sen, T. & Gupta, S. K. (1984). A state-of-art survey of static scheduling research involving due dates. *Omega*, 12(1):63–76.

Vasquez, J. C. D. & Birgin, E. J. G. (2017). Programação de tarefas em um ambiente flow shop com m máquinas para a minimização do desvio absoluto total de uma data de entrega comum. Master's thesis, Universidade de São Paulo.