XV SIMPÓSIO
Programas de Mestrado Profissional
11 a 12 de novembro de 2020

*Desafios de uma sociedade digital nos Sistemas Produtivos e na Educação*

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

# Avaliação do desempenho do algoritmo genético aplicado no problema de *flow shop* com bloqueio

Cassiano da Silva Tavares[1]

**Resumo -** Este trabalho avalia o desempenho da meta-heurística Algoritmo Genético, aplicada em um contexto de programação de operações com a característica *Flow Shop* com Bloqueio. O objetivo deste trabalho é realizar uma avaliação da meta-heurística para o conhecimento de seus ganhos computacionais obtidos em tempos de processamento dos exemplares, bem como, a qualidade de suas soluções. Os resultados apontam que para tamanhos de instâncias relativamente menores, o desempenho da meta-heurística não foi tão satisfatório, porém quando o tamanho das instâncias aumenta, a situação se inverte e o Algoritmo Genético fornece soluções satisfatórias em um tempo computacional muito abaixo do modelo determinístico.

**Palavras-chave**: *Flow Shop* com Bloqueio, Meta-Heurística, Algoritmo Genético.

**Abstract -** *This work evaluates the performance of the Genetic Algorithm meta-heuristic, applied in an operation programming context with the Flow Shop with blocking feature. The objective of this work is to carry out a evaluation of the meta-heuristic to know its computational gains obtained in times of processing the samples, as well as the quality of its solutions. The results indicate that for relatively smaller instance sizes, the performance of the meta-heuristic was not so satisfactory, however when the size of the instances increases, the situation is reversed and the Genetic Algorithm provides satisfactory solutions in a computational time lower than the deterministic model. .*

**Keywords:** *Flow Shop with Blocking, Meta-Heuristic, Genetic Algorithm.*

[1] UFSCar – cassiano_engenharia@yahoo.com.br

## 1. Introdução

A programação das operações ou *scheduling* é uma questão problemática vivida diariamente pelos profissionais que atuam nas áreas de Planejamento e Controle da Produção (PCP). O desafio desta atividade é definido pela busca em encontrar a melhor sequência de *n* tarefas que serão processadas em *m* recursos disponíveis (máquinas, colaboradores, servidores, entre outros) com o objetivo de atender uma medida de desempenho adotada como mais relevante naquele caso investigado (Pinedo,2008).

Os problemas de programações de operações podem ser classificados de acordo com suas respectivas características que são baseadas em restrições tecnológicas em função do fluxo produtivo. Sendo assim, MacCarthy e Liu (1993) apresentam a seguinte classificação: *job shop, flow shop, open shop,* máquina única e máquinas paralelas. Existem ainda meios de programações híbridos, misturando dois casos dos cinco já citados, devido a limitação de espaço, as definições de cada classificação não serão tratadas neste trabalho, para maiores detalhes, consultar MacCarthy e Liu (1993).

Diversos trabalhos na literatura abordam variações de cada classificação por meio de restrições adicionais para aproximar mais o problema da realidade. Uma classe especial, que vem sendo amplamente estudada desde a década de 50, é o *Flow Shop* Permutacional (FSP) que está presente em diversos segmentos como: indústrias metalúrgicas, indústrias de bebidas, indústrias farmacêuticas e indústrias de produção contínua em geral. Algumas variações do FSP que valem ser ressaltadas sãs as formulações que possuem restrições adicionais de: *setup*, bloqueio e manutenção preventiva (Miyata & Nagano, 2019) (Kometsu, 2015).

Como o problema de FSP com mais de três recursos é considerado NP-Completo pelo estado-da-arte na atualidade (Pinedo, 2005), diversos métodos heurísticos e meta-heurísticos veem sendo amplamente empregados ao longo dos anos, para promover soluções mais atraentes aos modelos, em um tempo computacional razoável (Kometsu, 2015).

Diante disso, o foco do presente trabalho é avaliar o desempenho de um método de solução meta-heurístico Algoritmo Genético aplicado em um ambiente FSP com a restrição adicional de bloqueio (PBFSP) com base no trabalho de Carrafa et al., 2001 visando oferecer uma nova perspectiva de solução para o desafio vivido diariamente pelos profissionais da área de PCP .

O trabalho está organizado da seguinte forma: a Seção 2 apresenta uma brevemente os métodos empregados no trabalho; a Seção 3 apresenta o método de resolução proposto em detalhes; os resultados são apresentados na Seção 4 e, por fim, a Seção 5 apresenta as conclusões e direções futuras.

## 2. Referencial Teórico

Esta seção apresenta uma breve revisão literária sobre os dois métodos empregados neste trabalho para a resolução do PBFSP.

## 2.1 Modelo determinístico

O modelo determinístico para o PBFSP com $m$ máquinas e minimização da duração total da programação $(C_{max})$ – com $n$ tarefas que são processadas em $m$ máquinas dispostas em série. Neste problema, todas as tarefas possuem o mesmo roteiro de produção. O termo permutacional indica que a sequência de tarefas estabelecida na primeira máquina é a mesma para todas as outras máquinas. Cada tarefa $(i = 1,2, \dots, n)$ possui uma operação na máquina $(j = 1,2, \dots, m)$ com um tempo de processamento denotado como $p_{ij}$.

A condição de bloqueio implica na ausência ou limite de estoque intermediário entre as máquinas. Logo, a tarefa $i$ finalizada na máquina $j$ deve aguardar na referida máquina até que a tarefa anterior $i - 1$ tenha sido finalizada e liberada da máquina $j + 1$. Esta condição causa um "bloqueio" na máquina $j$ ou seja, a impede de processar outras tarefas até que a tarefa $i$ seja liberada da máquina $j$ O tipo de bloqueio é denominado por *Release when started bloking* (RSb) (Miyata & Nagano, 2019).

O objetivo do problema é encontrar uma sequência de tarefas que minimiza a duração total da programação $(C_{max})$. A inclusão do termo permutacional reduz o total de sequências de $(n!)^m$para $n!$. Técnicas de solução exatas, tais como modelos matemáticos solucionados por meio de *solvers* (CPLEX, Gurobi,etc.), *branch-and-bound* e programação dinâmica, podem ser propostas para instâncias em que o número de tarefas é relativamente reduzido.

Uma forma de modelar matematicamente o problema, por meio de um modelo de programação linear inteira mista (MILP) é descrita a seguir. Considere as seguintes notações (com base na terminologia apresentada em Pinedo (2005)):

| | |
|---|---|
| $i, k, n$ | Índices auxiliares que representam as tarefas, sendo $i, k = (1, ., n)$; |
| $m, j$ | Índices auxiliares que representam as máquinas, sendo j= $(1, ., m)$; |
| $D_{[i]j}$ | *Departure time* (tempo de partida) da tarefa $i$ na máquina $j$; |
| $X_{[i]k}$ | 1 se a tarefa $k$ é alocada na máquina $j$. O caso contrário, e |
| $p_{kj}$ | Tempo de processamento da tarefa $k$ na máquina $j$. |

A variável $D_{[i]j}$representa o instante em que a tarefa $i$ é liberada da máquina $j$ para ser processada na máquina $j + 1$. E a variável binária $X_{[i]k}$ é uma variável de atribuição, indicando a posição da sequência $i$ onde a tarefa $k$ foi alocada. Com todas estas definições, o modelo MILP de PBFSP pode ser descrito como:

$$\min C_{max} = D_{[n]m} \tag{1}$$

sa.

$$\sum_{i=1}^{n} X_{[i]k} = 1 \qquad k = 1,2, \dots ., n \tag{2}$$

$$\sum_{k=1}^{n} X_{[i]k} = 1 \qquad i = 1,2, \dots ., m \tag{3}$$

$$D_{[i]j} = \sum_{k=1}^{n} X_{[i]k} p_{kj} \quad (4)$$

$$D_{[1]j} = D_{[1]j} + \sum_{k=1}^{n} X_{[i]k} p_{kj} \qquad j = 2, \ldots, m \quad (5)$$

$$D_{[i]1} \geq D_{[i-1]2} \qquad i = 2, \ldots, n \quad (6)$$

$$D_{[i]1} \geq D_{[i-1]1} + \sum_{k=1}^{n} X_{[i]k} p_{kj} \qquad i = 2, \ldots, n \quad (7)$$

$$D_{[i]1} \geq D_{[i-1]j+1} \qquad i = 2, \ldots, n; j = 2, \ldots, m-1 \quad (8)$$

$$D_{[i]j} \geq D_{[i]j-1} + \sum_{k=1}^{n} X_{[i]k} p_{kj} \qquad i = 2, \ldots, n; j = 2, \ldots, m \quad (9)$$

$$D_{[i]j} \in R^{+} \quad i = 1, \ldots, n; j = 1, \ldots, m \quad (10)$$

$$X_{[i]k} \in B \quad i = 1, \ldots, n; j = 1, \ldots, m \quad (11)$$

A equação (1) representa a função objetivo do problema que é a minimização da duração total da programação ($C_{max}$), representada pela data de partida da tarefa que ocupa a última posição da sequência na última máquina.

O conjunto de restrições (2) asseguram que somente uma tarefa $k$ pode ser programada na posição $i$ da sequência. Já o conjunto de restrições (3) asseguram que somente a tarefa $k$ pode ser programada em apenas uma posição $i$ da sequência. Os conjuntos de restrições (4) e (5) representam a data de partida da tarefa que ocupa a primeira posição. Por não existir tarefa posterior a primeira, a data de conclusão (*completion time*) da tarefa que ocupa a primeira posição da sequência em todas as máquinas é igual a sua data de partida nas $m$ máquinas.

Os conjuntos de restrições (6) e (7) determinam a data de partida da tarefa na primeira máquina. As restrições (8) e (9) definem a data de partida da tarefa no restante das máquinas. Observe que se os valores de $D_{[i]j}$ obtido pelas restrições (6) e (8) forem maiores que os das restrições (7) e (9) um tempo de bloqueio maior que zero unidades de tempo (u.t.) ocorre entre as máquinas $j$ e $j+1$. Por outro lado, se os valores de $D_{[i]j}$ forem maiores em (7) e (9) é provável que um tempo de ociosidade de no mínimo zero (u.t.) ocorre entre as máquinas $j$ e $j+1$. Por fim, (10) e (11) apresentam os domínios das variáveis.

## 2.2 Algoritmo Genético

Alguns problemas na literatura de Pesquisa Operacional são classificados como NP-difíceis quando os métodos de solução exatos não conseguem atingir bons resultados em um tempo computacional aceitável. O PBFSP é um caso especial desta classe de problemas, devido ao espaço amostral de soluções ser $n!$ (Pinedo, 2005). Para apoiar este tipo de problema, existem métodos de soluções alternativos como as Heurísticas e as Meta-Heurísticas que oferecem boas soluções em um tempo computacional aceitável.

Como Heurísticas se pode destacar na literatura do tema: Busca Local, Algoritmo NEH e Busca Tabu. Já nas Meta-Heurísticas, se pode destacar:

Algoritmo Genético (do inglês *Genectic Algorithmic* (GA)), Colônia de Formigas e *Simulated Annealing* (Carrafa, et al.,1999).

Para o presente trabalho, a Meta-Heurística GA foi selecionada para a avaliação do seu desempenho, perante a formulação MILP. A GA foi desenvolvida por Holland in 1975 e realiza uma analogia das sequências de processamento das tarefas com uma sequência de ***cromossomos***, onde um agrupamento de cromossomos constitui uma ***população***. (Alharkan, 2010)( Holland, 1992).

Cada sequência possui o seu valor respectivo de *makespan* que caracteriza uma medida de desempenho chamada ***fitness***. Esta medida de desempenho, associa a sequência com o valor da função objetivo. Então o GA realiza uma busca iterativa, percorrendo pelos membros da população em busca do menor valor do *fitness*. Esta busca é composta por três fases: reprodução, seleção e critério de parada.

Neste contexto, as *gerações de cromossomos* são definidas pelas suas ***reproduções***, dadas por uma operação de *crossover* e *mutação* entre as cadeias genéticas. O operador ***crossover*** é definido pelo cruzamento das cadeias genéticas de mesmo tamanho, dado pela seleção em um gene específico de ambas as cadeias. Após a seleção deste gene, as cadeias são subdivididas em cadeias menores, definidas até o gene selecionado e, após isso, é realizada uma junção entre as cadeias, avaliando qual gene já foi inserido na sequência para evitar problemas de infactibilidade.

Já o operador ***mutação*** é definido pelas combinações aleatórias possíveis e factíveis das novas sequências obtidas da fase de *crossover* dentro de uma mesma geração. Por fim, o operador ***seleção*** seleciona os melhores indivíduos de cada geração, através da medida de desempenho *fitness* (quanto menor, melhor), carregando estas informações genéticas para a geração posterior.

Todo o processo do GA ocorre de forma iterativa, percorrendo as três fases, buscando a melhor sequência genética da população. Então estas reproduções geram sequências filhas, netas, bisnetas, e assim, sucessivamente até que o **critério de parada** pré estabelecido pelo algoritmo seja atendido.

## 3. Método

Esta seção apresenta o método de resolução proposto para a avaliação do desempenho do GA em relação ao modelo exato MILP. Em todas as análises a codificação do cromossomo foi estruturada como uma dada sequência de tarefas a serem programadas, então cada gene da cadeia equivale a uma tarefa a ser programada. Mediante isso, a Subseção 3.1 apresenta os critérios adotados para a fase de Reprodução, enquanto a Subseção 3.2 apresenta as decisões adotadas para a fase de Seleção.

### 3.1. Fase de Reprodução

Esta fase possuí duas decisões muito importantes para a avaliação do desempenho do GA que são: as decisões de estruturação dos operadores

*crossover* e a de mutação. Para o operador *crossover* foi adotada a seguinte estratégia: -mediante a inicialização de duas sequências de tarefas, um número aleatório pertencente ao domínio de ambas as sequências foi sorteado para se traçar um **ponto de *crossover***.

Após isso, o cruzamento das sequências foi iniciado, obtendo duas novas subsequências filhas iniciadas das tarefas mais à esquerda das sequências pais (primeira tarefa a ser processada). para a direita, de forma sequencial, até o ponto sorteado. Então, foram analisadas todas as tarefas já pertencentes as cada subsequência respectivamente e, então as tarefas novas foram inseridas de forma crescente em relação ao número ordinal de cada tarefa: $(job_1, job_2, job_3, \ldots., job_n)$.

Para o operador *mutação* foram adotadas as seguintes decisões: para cada filho gerado da fase de Reprodução, foram geradas todas as permutações possíveis da dada sequência obtida e, após isso, para cada sequência permutada a medida de desempenho *fitness* foi calculada para a seleção da sequência mais atraente (menor valor de *fitness*).

Esta seleção ocorreu analisando se o valor do *fitness* atual era menor do que o valor obtido no cálculo da sequência anterior. Sendo assim, quando esta condição foi verdadeira, o valor da sequência foi subscrito e a sequência com o menor *fitness* foi descartada (processo de morte do indivíduo).

### 3.2. Fase de Seleção

Nesta fase ocorreram as seleções dos indivíduos que perpetuariam para a nova geração propagando a herança genética mais atraente do ponto de vista do problema. A seleção ocorreu por meio de um *operador de seleção* que avaliava os melhores indivíduos da população naquela respectiva geração (menor *fitness*). Este operador recebia quatro sequências com suas respectivas soluções candidatas, a saber, seq1 (pai), seq2 (pai), seq3(filho após mutação) e seq4(filho após mutação).

Após isso, o operador realizava o processo de seleção por meio da medida de desempenho *fitness* realizando um *ranking* de soluções, classificadas em ordem crescente. Por fim, as duas melhores soluções perpetuavam para a próxima geração e as sequências da população inicial (seq1 e seq2) eram sobrescritas, retroalimentando o algoritmo.

## 4. Resultados e Discussão

Esta seção apresenta os experimentos computacionais do método de solução proposto. A Subseção 4.1 apresenta uma visão da dimensionalidade do problema, com um resumo dos dados. A Subseção 4.2 apresenta os resultados numéricos do MILP e os resultados do GA. Os dois métodos de solução propostos foram implementados na linguagem Python na versão 3.6 e usando o *software* de otimização IBM ILOG CPLEX 12.9. Todos os experimentos dos modelos foram realizados em um computador Intel® Core TM i7-3537U 2.00

GHz com 16 GB de memória RAM usando o sistema operacional Ubuntu 18.04.3 LTS.

### 4.1 Criação dos dados

Para a criação das instâncias foram definidas diferentes classes que indicam o tamanho das instâncias em função da cardinalidade de seus conjuntos. As instâncias são constituídas por variações dentro das classes obedecendo a distribuição de probabilidade uniforme e as cardinalidades dos conjuntos.

Para isso, foi criado um padrão de classe indicando $aN - bM$ onde significa $a$ número de tarefas a serem sequenciadas e $b$ número de máquinas disponíveis. Então, foram criadas as seguintes classes: N5M4, N10M8, N20M16, 40N32M e 80N64M. Todas as classes possuem cinco exemplares de instâncias e são compostas por apenas o parâmetro $p_{[k]j}$ que representa o tempo de processamento da tarefa $k$ na máquina $j$. Para todos os exemplares, o parâmetro foi gerado de forma aleatória seguindo a distribuição uniforme no intervalo [1,11], dado em minutos.

### 4.2 Resultados computacionais

O modelo determinístico foi resolvido com o objetivo de analisar seu desempenho e identifica o custo computacional e o nível de qualidade das soluções. Para os testes computacionais foram utilizados todos os vinte e cinco exemplares descritos na Subseção 4, como critérios de parada foram preestabelecidos o limite de tempo computacional de 3600 segundos e/ou o alcance de 0% de GAP. Por fim, foram utilizadas as configurações padrões do *solver* CPLEX.

Para a avaliação de desempenho do GA e para que a meta-heurística obtivesse o comportamento próximo ao modelo MILP, foi necessário criar uma função *fitness* que tivesse o maior número de características(restrições) do modelo MILP para a representação do bloqueio Rbs. Mediante isso, foi realizada uma análise do modelo e foi observado que o conjunto de restrições (4)-(9) caracterizavam o comportamento do bloqueio para todas as máquinas disponíveis. Porém, todo este conjunto de restrições possuí uma variável binária $X_{[i]k}$ associada que deveria ser considerada. A alternativa encontrada foi a considerar o comportamento da variável em função de dados conhecidos.

Então, para os termos das restrições que necessitavam de algum tipo de somatório, envolvendo a variável $X_{[i]k}$ obteve-se as seguintes aproximações:

$$\sum_{i=1}^{n} X_{[i]k} \approx \alpha N \tag{12}$$

$$\sum_{k=1}^{n} X_{[i]k} \approx \alpha M \tag{13}$$

Em (12) e (13) o parâmetro $N$ representa o tamanho do conjunto de tarefas e o parâmetro $M$ representa o tamanho do conjunto de máquinas disponíveis. Assim, quando em um restrição, houvesse um somatório com a variável $X_{[i]k}$ como é o caso da (5) e, como todas as máquinas são utilizadas sempre, em toda a sequência, podemos inferir que o valor máximo deste somatório é o número total de máquinas $M$.

Então se pode adotar esta substituição sem perda de generalidade para $N$ seguindo a mesma lógica, uma vez que, todas as tarefas devem ser programadas. Já $\alpha$ é um parâmetro artificial formado por números reais inteiros utilizado para o ajuste da curva de aderência. Por fim, após estas aproximações foi possível definir a função objetivo para o cálculo da medida de desempenho *fitness* através do modelo apresentado abaixo.

$$fitness = \min C_{max} = D_{[n]m} \tag{14}$$

sa.

$$D_{[i]j} = \beta M p_{kj} \tag{15}$$

$$D_{[1]j} = D_{[1]j} + \beta M p_{kj} \quad j = 2, \ldots., m \tag{16}$$

$$D_{[i]1} \geq D_{[i-1]1} + \alpha N p_{kj} \qquad i = 2, \ldots., n \tag{17}$$

$$D_{[i]j} \geq D_{[i]j-1} + \alpha N p_{kj} \qquad i = 2, \ldots., n; j = 2, \ldots., m \tag{18}$$

$$restrições\ (6), (8)\ e\ (10).$$

Após a definição da modelagem da função *fitness* foram realizados testes de aderência em alguns cenários, para a escolha do melhor cenário em função de comportamento da função criada em relação ao modelo MILP. Para os testes computacionais o critério de parada estabelecido foi o limite de 1000 iterações do algoritmo para cada exemplar, levando aproximadamente oito horas para o processamento de cada cenário (25.000 iterações).

Os cenários criados tiveram como objetivo selecionar os melhores valores $\alpha$ que variou em cada cenário no intervalo [1,2,3]. E o desempenho de cada cenário é apresentado pelo Tabela 1.

Na primeira coluna da Tabela 1 é apresentado o exemplar avaliado, da segunda à quarta colunas são apresentados os resultados computacionais do modelo determinístico, sendo apresentados os valores da função objetivo, tempos computacionais e os valores de GAP, respectivamente. Da quinta à décima terceira coluna, os resultados são apresentados em três blocos, onde cada bloco representa um cenário.

A primeira coluna de cada bloco apresenta o valor obtido da função *fitness* na milésima iteração do GA. Para a análise correta do resultado desta coluna, foi necessário realizar avaliação da solução do modelo MILP antes do cálculo da medida de desempenho ganho que será abordada à frente. Então, quando o modelo MILP encontro uma solução ótima para o dado exemplar e o GA obteve uma solução menos custosa, a solução do GA não foi considerada, pois seria infactível para o modelo MILP e foi representada por "--------".

Já a segunda coluna de cada bloco, apresenta o tempo computacional gasto no processamento do exemplar na medida de tempo em segundos, e por fim, a última coluna de cada bloco é apresentado o valor relativo de ganho que a solução do modelo GA obteve em relação a solução do modelo determinístico, processando a mesma instância, que foi calculado da seguinte maneira:

$$ganho = \left(\frac{z_{determinístico} - z_{GA}}{z_{GA}}\right) - 100\% \tag{19}$$

Analisando a Tabela 1 se pode observar que o desempenho do GA não foi satisfatório para duas primeiras classes de instâncias (5M4M e 10N8M) em todos os cenários, não havendo nenhum ganho no processamento de nenhum exemplar. Porém, nas classes de instâncias maiores, a situação se inverte. Para melhor compreensão, uma análise por cenários é apresentada.

Cenário 1: neste cenário o GA conseguiu obter sete soluções menos custosas que o CLPEX em um tempo muito menor, porém devido ao valor do GAP obtido pelo CPLEX no processamento da instância N20M16-1 ser muito baixo (4,98%), não se pode afirmar com certeza que a solução do MILP não é ótima para este exemplar. Em dez exemplares o GA encontrou soluções infactíveis e em sete exemplares, o GA obteve uma solução pior do que a do modelo MILP. Então, neste cenário, o GA apresentou 28% das soluções melhores que o MILP, 28% das soluções piores que o MILP, 40% das soluções infactíveis e em 4% dos exemplares, o GA apresentou a mesma solução que o CPLEX.

Tabela 1 - Avaliação de desempenho do GA

| | Determinístico | | | Cenário 01 – α = 1 | | | Cenário 02 – α = 2 | | | Cenário 03 – α = 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instância | *Fitness* | Tempo | GAP | *Fitness* | Tempo | Ganho | *Fitness* | Tempo | Ganho | *Fitness* | Tempo | Ganho |
| N5M4-1 | 49,99 | 0,04 | 0,00% | 45,00 | 13,82 | -------- | 48,00 | 12,70 | -------- | 62,00 | 19,67 | -------- |
| N5M4-2 | 47,00 | 0,04 | 0,00% | 47,00 | 14,06 | 0,00% | 47,00 | 12,49 | 0,00% | 47,00 | 27,82 | 0,00% |
| N5M4-3 | 44,99 | 0,04 | 0,00% | 43,00 | 13,79 | -------- | 43,00 | 12,26 | -------- | 47,00 | 21,57 | -------- |
| N5M4-4 | 55,00 | 0,03 | 0,00% | 48,00 | 13,81 | -------- | 53,00 | 12,49 | -------- | 54,00 | 21,65 | -------- |
| N5M4-5 | 55,00 | 0,03 | 0,00% | 50,00 | 14,01 | -------- | 65,00 | 12,69 | -------- | 80,00 | 20,68 | -------- |
| N10M8-1 | 102,00 | 0,19 | 0,00% | 105,00 | 20,55 | -------- | 103,00 | 18,69 | -------- | 112,00 | 31,88 | -------- |
| N10M8-2 | 108,00 | 0,23 | 0,00% | 106,00 | 21,20 | -------- | 118,00 | 18,31 | -------- | 114,00 | 33,26 | -------- |
| N10M8-3 | 110,00 | 0,60 | 0,00% | 113,00 | 20,53 | -------- | 115,00 | 20,20 | -------- | 116,00 | 32,12 | -------- |
| N10M8-4 | 112,00 | 0,21 | 0,00% | 108,00 | 19,41 | -------- | 108,00 | 18,71 | -------- | 112,00 | 33,37 | 0,00% |
| N10M8-5 | 101,00 | 0,47 | 0,00% | 99,00 | 21,15 | -------- | 103,00 | 18,78 | -------- | 102,00 | 34,38 | -------- |
| N20M16-1 | 240,00 | 3600,00 | 4,98% | 230,00 | 67,47 | 4,35% | 242,00 | 69,05 | -0,83% | 247,00 | 103,22 | -2,83% |
| N20M16-2 | 236,99 | 3600,00 | 4,96% | 242,00 | 68,42 | -2,07% | 247,00 | 68,50 | -4,22% | 230,00 | 109,47 | 3,04% |
| N20M16-3 | 225,00 | 3600,00 | 4,68% | 245,00 | 61,18 | -8,16% | 238,00 | 59,80 | -5,78% | 248,00 | 106,56 | -9,27% |
| N20M16-4 | 235,00 | 1786,41 | 0,00% | 251,00 | 62,55 | -------- | 247,00 | 64,33 | -------- | 265,00 | 105,97 | -------- |
| N20M16-5 | 215,00 | 3600,00 | 2,30% | 230,00 | 62,04 | -6,52% | 244,00 | 68,66 | -13,49% | 242,00 | 65,90 | -11,16% |
| N40M32-1 | 504,00 | 3600,00 | 22,35% | 507,00 | 405,65 | -0,59% | 516,00 | 417,96 | -2,38% | 534,00 | 411,83 | -5,62% |
| N40M32-2 | 497,99 | 3600,00 | 20,68% | 513,00 | 397,15 | -2,93% | 513,00 | 396,15 | -3,01% | 516,00 | 384,11 | -3,49% |
| N40M32-3 | 524,00 | 3600,00 | 22,31% | 512,00 | 380,75 | 2,34% | 527,00 | 425,33 | -0,57% | 529,00 | 395,45 | -0,95% |
| N40M32-4 | 498,99 | 3600,00 | 17,64% | 501,00 | 387,50 | -0,40% | 515,00 | 414,74 | -3,21% | 519,00 | 441,98 | -3,86% |
| N40M32-5 | 502,99 | 3600,00 | 18,69% | 515,00 | 393,39 | -2,33% | 515,00 | 421,19 | -2,39% | 518,00 | 382,33 | -2,90% |
| N80M64-1 | 1113,00 | 3600,00 | 29,95% | 1059,00 | 3083,03 | 5,10% | 1081,00 | 3049,58 | 2,88% | 1095,00 | 2897,53 | 1,64% |
| N80M64-2 | 1122,00 | 3600,00 | 29,97% | 1061,00 | 3141,14 | 5,75% | 1074,00 | 3215,69 | 4,28% | 1062,00 | 2919,03 | 5,65% |
| N80M64-3 | 19566,00 | 3600,00 | 95,99% | 1066,00 | 3065,05 | 1735,46% | 1064,00 | 4257,96 | 94,56% | 1091,00 | 3014,32 | 1693,40% |
| N80M64-4 | 1116,00 | 3600,00 | 29,26% | 1058,00 | 3348,35 | 5,48% | 1058,00 | 3906,06 | 5,20% | 1075,00 | 2865,82 | 3,81% |
| N80M64-5 | 206363,00 | 3600,00 | 96,28% | 1047,00 | 3544,41 | 19609,93% | 1056,00 | 4462,27 | 99,49% | 1066,00 | 3661,70 | 19258,63% |

Cenário 2: neste cenário o GA conseguiu obter cinco soluções menos custosas que o CLPEX em um tempo muito menor. Novamente, foram encontrados dez exemplares com soluções infactíveis e o cenário de soluções piores que o modelo MILP aumentou para nove exemplares. Então, neste cenário, o GA apresentou 20% das soluções melhores que o MILP, 36% das soluções piores que o MILP, 40% das soluções infactíveis e, novamente, em 4% dos exemplares, o GA apresentou a mesma solução que o CPLEX.

Cenário 3: neste cenário, o GA conseguiu obter seis soluções menos custosas que o CLPEX em um tempo muito menor. Os cenários de exemplares infactíveis diminuíram para nove e de soluções piores também diminuiu para oito exemplares. Então, neste cenário, o GA apresentou 24% das soluções melhores que o MILP, 32% das soluções piores que o MILP, 36% das soluções infactíveis e, novamente, em 8% dos exemplares, o GA apresentou a mesma solução que o CPLEX.

## 5. Considerações finais

Neste trabalho foi realizada uma avaliação de desempenho da meta-heurística GA, aplicada em um contexto de programação de operações com a característica PBFSP através de 25 exemplares gerados aleatoriamente subdivididos em cinco classes. O objetivo didático deste trabalho foi alcançado, sendo que, o desempenho da meta-heurística foi avaliado para o conhecimento de seus ganhos computacionais obtidos em tempos de processamento dos exemplares, bem como, a qualidade de suas soluções. Também se pode afirmar que o objetivo prático também foi alcançado, uma vez que, é apresentada uma nova perspectiva de solução para o desafio dos profissionais da área de PCP no PBFSP que está presente em diversos segmentos industriais.

Por meio dos experimentos computacionais realizados, conclui-se que: 1) o número de exemplares é limitado e não pode ser generalizado e 2) o desempenho do GA não foi satisfatório para duas primeiras classes de instâncias (5M4M e 10N8M) em todos os cenários, não havendo nenhum ganho no processamento de nenhum exemplar. Porém, nas classes de instâncias maiores, a situação melhora, e o primeiro cenário possuiu maior aderência aos dados pelos quais os exemplares foram avaliados.

Os resultados mais satisfatórios foram obtidos no Cenário 1, onde o GA apresentou 28% das soluções melhores que o MILP, 28% das soluções piores que o MILP, 40% das soluções infactíveis e, em 4% dos exemplares, o GA apresentou a mesma solução que o CPLEX. Isso ocorreu, devido à complexidade computacional de gerar uma boa programação de operações, considerando a dimensão das cardinalidades dos conjuntos de máquinas e tarefas. Com isso, o autor considera que esse estudo deve ser aprofundado, para conseguir resultados melhores de desempenho, através da comparação de resolução de outros métodos, a saber, Colônia de Formigas e *Simulated Annealing.*

## Referências


Alharkan, I. M. (2010). Algorithms for sequencing and scheduling. Disponível em: http://ikucukkoc.baun.edu.tr/lectures/EMM4129/AlgorithmsforSequencingandScheduling. Acesso em 14/07/2020 às 14:00h.

Caraffa, V.; et al. (2001) Minimizing makespan in a blocking flowshop using genetic algorithms. *International Journal of Production Economics.* v.70, p.101-115, Elsevier.

Holland, J. H. (1992). Adaptation in Natural and Artificial Systems.

Pinedo, M. L. (2005). Planning and Scheduling in Manufacturing and Services. *Springer*.

Kometsu., A. S. (2015). Heurística Evolutiva para a minimização do atraso total em ambiente de produção flow shop com buffer zero. *Dissertação*. Universidade de São Paulo.

Maccarthy, B. L.; Liu, J. Y. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*., v.31, p.:59-79, Elsevier.

Miyata, H. H.; Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Systems with Applications*. V.137, p.130-156, Elsevier.