OTIMIZAÇÃO DE PROBLEMAS DE PROGRAMAÇÃO DE PRODUÇÃO UTILIZANDO MÉTODOS DE PESQUISA OPERACIONAL

Bruna Bernardes de Aguiar (UFU/FACES) brunabernardes123@hotmail.com
Jorge Von Atzingen dos Reis (UFU/FACES) jorgereis@ufu.br

Resumo:

O planejamento e controle da produção (PCP) em uma empresa têm como objetivo reduzir os custos relacionados a produção. Um das tarefas do PCP é determinar a melhor sequência de produção, problema conhecido na literatura como *Job Shop Problem* (JSP). A determinação do sequenciamento é uma tarefa complexa devido às diversas restrições que necessitam ser consideradas. Este trabalho tem como objetivo principal testar o comportamento de um modelo matemático para gerar uma sequência de produção para uma linha de produção de modo a minimizar o *makespan*. Este modelo foi resolvido através do software GUROBI© e foram incluídos testes relacionados à modificações nos dados de entrada e na utilização de uma heurística generalizada para auxiliar a resolução matemática do modelo matemático testado, partindo-se então para a utilização de outros métodos de Pesquisa Operacional em busca do resultado esperado. Os testes computacionais foram realizados com dados *benchmark* da literatura.

Palavras-Chaves: Problema de *Flow Shop*, Problema de *Job Shop*, Modelagem Matemática, Heurística.

Introdução

Tubino (2009) define como Planejamento e Controle da Produção (PCP) a atividade responsável por organizar os dados e as tomadas de decisões considerando todos os fatores do processo produtivo. O PCP precisa definir a melhor maneira de coordenar e aplicar os recursos da empresa de modo a atender os planos estabelecidos nos níveis estratégico, tático e operacional.

Algumas das atividades mais importantes do PCP são as previsões da demanda, o planejamento estratégico, o planejamento mestre, a programação da produção e o acompanhamento e controle da produção (TUBINO, 2009).

Dentro da programação da produção, um dos principais problemas é a alocação dos recursos para assegurar a data de conclusão dos projetos. Esse é um problema de sequenciamento da produção, conhecido na literatura como *Job Shop Problem* ou *Job Scheduling Problem* (JSP). É parte importante do Planejamento e Controle da Produção, devido às empresas necessitarem de um plano eficiente a fim de elevar sua produtividade utilizando recursos restritos, como

máquina e rotinas de trabalho. Muitas empresas aplicam multas e restrições em caso de atrasos, logo, o planejamento eficiente é um fator determinante para a empresa (TUBINO, 2009).

As tarefas de um processo produtivo são independentes entre si, o que significa que cada uma possui seu próprio fluxo nas máquinas. Comum a todos os processamentos é que cada máquina pode processar uma tarefa por vez e cada tarefa só pode ser processada por uma máquina por vez. O tempo de processamento de todas as tarefas é conhecido como *makespan* (*Cmax*) e objetivo de solucionar um JSP são tornar este tempo o menor possível (JAIN & MEERAN, 1998).

Os métodos de Pesquisa Operacional (PO) podem ser utilizados para solucionar problemas de programação como o JSP. A PO é formada por um aglomerado de técnicas de solução para os diversos tipos de problemas, a determinação da solução é baseada na complexidade do problema (TAHA, 2008).

Algumas dos métodos de PO são a programação linear, a programação inteira, a programação dinâmica, otimização em redes, a programação não linear e as heurísticas e meta-heurísticas. As heurísticas e meta-heurísticas diferentemente dos outros métodos podem não apresentar uma solução ótima, mas sim uma solução boa ou melhorada, dado que elas são utilizadas na solução de problemas que possuem modelos matemáticos muito complexos, como o caso do JSP (TAHA, 2008).

2. Revisão Bibliográfica

Nos processos produtivos há grandes gargalos que decorre da organização e planejamento dos processos de fabricação com um todo, sendo necessário um planejamento eficiente que consiga alcançar seu objetivo final em menor tempo, com menor custo, no prazo e parâmetros de qualidade estabelecidos inicialmente. Tanto os administradores de organizações quanto clientes estão sempre mais exigentes, sempre em busca de novidades e benefícios.

Com um bom sequenciamento de máquinas é possível gera inúmeros benefícios à empresa, minimizando o atraso, otimizando o tempo gasto para produção e assim atender mais pedidos, o que leva a aumentar a receita da empresa e sua credibilidade no mercado, podendo elevar a sua participação no mercado (*market share).*

O problema de planejamento de produção, tende a se realizar a entrega do produto no prazo estabelecido, otimizando o tempo de produção e a utilização de insumos. A medida de desempenho principal da programação de produção é a referente ao tempo de processamento e atraso.

O planejamento da produção pode ser classificado como estático ou dinâmico, em ambos se considera o hoje, porém a diferenciação está que no planejamento estático as mudanças

ocorridas durante o processo não são consideradas, já no planejamento dinâmico são considerados os imprevisto e mudanças de ordens de serviço (TUBINO, 2009).

Tendo em vista a otimização deste processo um dos problemas mais relevantes vem a ser a alocação dos recursos para que haja garantia da conclusão dos projetos estabelecidos. O problema de sequenciamento *Job Shop* ou *Job Scheduling* e o *Flow Shop* (problema de sequenciamento de tarefas em máquinas de produção) são partes importantes do Planejamento e Controle da Produção, pois sempre é buscado se realizar as tarefas com o mínimo de custo possível sem ter que repassar este valor aos clientes ou afetar a produção.

Atualmente as empresas estão buscando ainda mais diminuir o custo gasto na produção de seus produtos finais, sem que haja perda nos seus padrões estabelecidos para está produção, portanto as pesquisas nas áreas de otimização e sequenciamento de máquinas passam a ter grande relevância, como no caso do problema de *Job Shop* e *Flow Shop*.

Segundo Pacheco & Santoro (1999) quando se existe uma série de elementos que envolvam número de máquinas, roteiro de ordens, regime de chegada de ordens, variabilidade dos tempos de processamento, entre outras características, classifica-se como sendo um problema de *Job Shop Scheduling* e para tentar resolver esse problema uma abordagem interessante é tratá-lo como um problema de otimização (Barbara, 2000), pois neste se tenta organizar as tarefas buscando a melhor eficiente possível.

Segundo Arenales et al. (2007), um *Job Shop* Clássico é um ambiente de produção com *n* tarefas e *m* máquinas, em que cada tarefa é processada nas *m* máquinas, de acordo com um roteiro preestabelecido. Considere, por exemplo, 5 tarefas e 3 máquinas, denotadas por 1, 2 e 3. As matrizes *O* e *P* a seguir, representam, respectivamente, a matriz de operações, e a matriz de tempos de processamento nessas máquinas. Assim, por exemplo, a primeira linha da matriz *O* indica que a tarefa 1 é processada nas máquinas 2, 1 e 3, nesta ordem, com tempos de processamento de 5, 7 e 10 unidades de tempo, respectivamente, correspondentes aos elementos da primeira linha da matriz *P*.

Figura 1 – Exemplo do problema de *Job Shop*

$$\boldsymbol{O} = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 2 \end{bmatrix} \quad \boldsymbol{P} = \begin{bmatrix} 5 & 7 & 10 \\ 9 & 5 & 3 \\ 5 & 8 & 2 \\ 2 & 7 & 4 \\ 8 & 8 & 8 \end{bmatrix}$$

Fonte: Adaptado de Arenales et al. (2007)

Admita que as *n* tarefas estão disponíveis para processamento inicialmente e que não é permitida a interrupção do processamento de nenhuma tarefa. Sejam os parâmetros:

p_{ik} : Tempo de processamento da tarefa *i* na máquina *k*;

maq_i : Máquina que processa a *i-ésima* tarefa;

M : Número suficientemente grande.

E as seguintes variáveis de decisão:

C_{ik} : Instante de término do processamento da tarefa *i* na máquina *k*;

x_{ijk} : Variável binária que assume valor 1 se a tarefa *i* precede a tarefa *j* na máquina *k* e 0, caso contrário.

Assim, o modelo de programação matemática relativo ao *Job Shop* pode ser representado por:

Figura 2 – Modelo Matemático do *Job Shop Problem*

$$\min \sum_{i=1}^{n} C_{i\,m}$$

$$\begin{aligned}
C_{i,maq_1} &\geqslant p_{i,maq_1} && \forall\ i=1,\ldots,n \\
C_{i,maq_{k+1}} &\geqslant C_{i,maq_k} + p_{i,maq_{k+1}} && \forall\ i=1,\ldots,n \quad \forall\ k=1,\ldots,m-1 \\
C_{jk} &\geqslant C_{ik} + p_{jk} - M(1-x_{ijk}) && i\neq j \quad \forall\ i,j=1,\ldots,n \quad \forall\ k=1,\ldots,m \\
C_{ik} &\geqslant C_{jk} + p_{ik} - M\,x_{ijk} && i\neq j \quad \forall\ i,j=1,\ldots,n \quad \forall\ k=1,\ldots,m \\
C_{ik} &\geqslant 0 && \forall\ i=1,\ldots,n \quad \forall\ k=1,\ldots,m \\
x_{ijk} &\in [0,1] && \forall\ i,j=1,\ldots,n \quad \forall\ k=1,\ldots,m
\end{aligned}$$

Fonte: Adaptado de Arenales et al. (2007)

A função objetivo expressa a minimização do tempo de fluxo total das tarefas. O primeiro conjunto de restrições garante que a primeira operação de cada tarefa *i* é completada após o respectivo tempo de processamento. O segundo conjunto de restrições impõe que a operação *k+1* seja concluída depois do término da operação *k* e do tempo de processamento da operação *k+1*. O terceiro e o quarto conjunto de restrições são restrições disjuntivas que indicam respectivamente se, na máquina *k*, a tarefa *i* precede a tarefa *j*, ou se a tarefa *j* precede a tarefa *i*.

Caso $x_{ijk}=1$ então se tem que: $C_{jk} \geqslant C_{ik} + p_{jk}$ e $C_{ik} - C_{jk} \geqslant p_{ik} - M$, isto é, o quarto conjunto de restrições é desativado. De modo análogo, se $x_{ijk}=0$ então $C_{jk} - C_{ik} \geqslant p_{jk} - M$ e $C_{ik} \geqslant C_{jk} + p_{ik}$, isto é, o terceiro conjunto de restrições é desativado. Os dois últimos conjuntos de restrições estabelecem o tipo das variáveis.

Já no problema *Flow Shop* segundo Nagano, Moccellin & Lorena (s/d) todas as tarefas possuem a mesma sequência de processamento no conjunto de máquinas. Em seu trabalho, os autores especificaram no *Flow Shop Permutacional*, que adiciona à característica do problema

que para cada máquina existe sempre a mesma sequência de tarefa, assim diferente do *Job Shop* que para cada tarefa se tem sua própria sequência de processamento no conjunto de máquinas. E ainda será utilizado o mesmo modelo matemático aplicado ao *Job Shop*.

O algoritmo de ordenação *Quicksort* é utilizado para rearranjar conjunto de valores, como vetores e tabelas, em ordem crescente ou decrescente (FARIAS,2012). Este foi o método escolhido para ordenar os dados em ordem crescente e também em ordem decrescente para se realizar os testes e comparações.

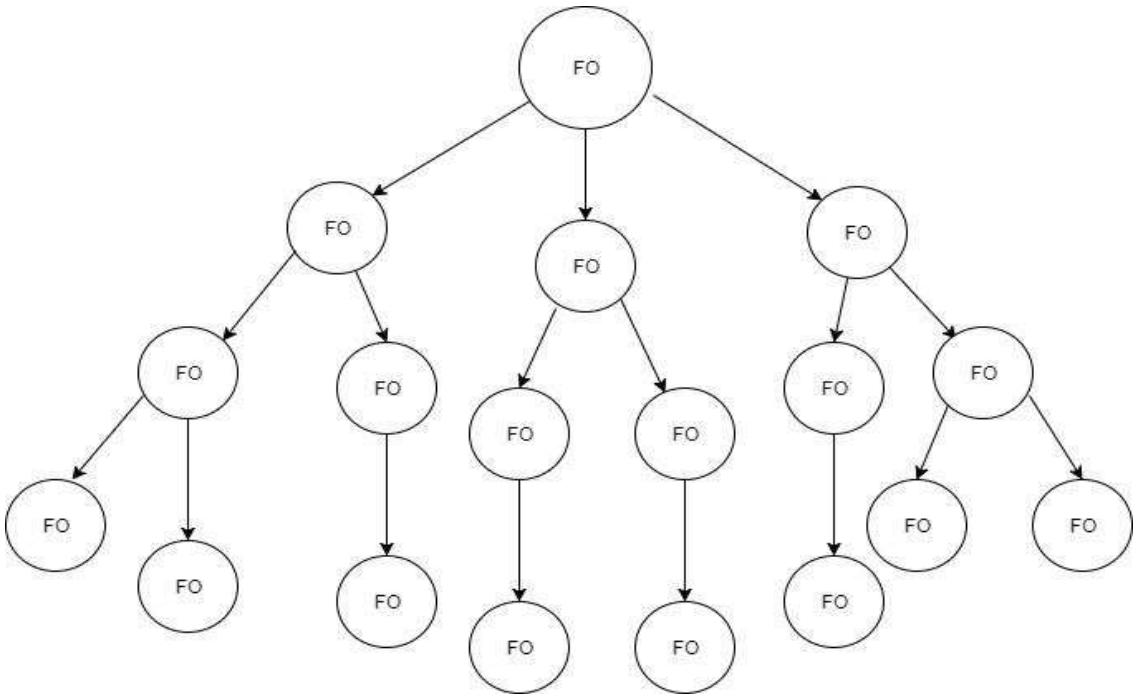
2. Heurística Generalizada & o Software GUROBI©

O software GUROBI© é um pacote de otimização disponível na universidade, utiliza o método simplex para resolver os modelos matemáticos, este método é utilizado para determinar algebricamente a solução ótima de um problema de programação linear. Sabendo-se que a solução ótima é um ponto extremo gerado pelas restrições.

Sendo realizada no mesmo a utilização de um comando para controlar a porcentagem de tempo computacional que utiliza heurística durante o processo, uma heurística generalizada que pode ser aplicada a qualquer problema, seu funcionamento no software GUROBI© é a utilização de um pré-processamento em tudo, sendo a ideia básica a utilização relaxação, como por exemplo, uma relaxação lagrangeana onde se fornece limites, que em conjunto com outras soluções factíveis aos problemas, fornecendo as possíveis soluções viáveis ao problema, sendo possível sua participação também no desenvolvimento de heurísticas.

Seguindo essa linha o GUROBI© então seleciona determinada variável e vai tornar ela mais fácil de ser resolvida, ou seja, fazer uma relaxação no problema. Sendo análoga a uma árvore de ramificações onde ele terá que explorar todos os nós nela presentes, como exemplo o *branch and bound,* que é um método que tem a ideia baseada enumerar as possíveis soluções que são as candidatas a soluções ótimas do problema em questão que está sendo tratado. *Branch* é o termo que tem referência a efetuar as partições das soluções no espaço, e o termo *Bound* tem ênfase na busca por essas soluções viáveis utilizando os limites que são calculados ao longo das enumerações.

Figura 3 – Árvore ramificada (*Branch and bound*)



Fonte: Autoria própria (2018)

Sendo feita essa relação com uma árvore ramificada, percebe-se que ela tem que explorar todas as soluções possíveis, ou seja, os nós. Quando utiliza tal heurística o Software GUROBI© segue o caminho de alguns nós, desconsiderando outros que estão mais distantes da atual solução, utilizando aqueles que estão o mais próximo possível. Então quando for feita a heurística, ele seleciona os pontos já escolhidos, elimina os arcos de maior comprimento e tenta alocar a solução para esses arcos menores minimizando o tamanho total e dando uma solução local.

Após para a heurística, o GUROBI© passa a utilizar para resolução o simplex e o mesmo continua otimizando a partir daquele ponto, sendo um método que tem como ideia básica resolver repetidas vezes um sistema de equações lineares para explorar melhor o espaço das soluções factíveis interrompendo o processo de refinamento da solução quando for obtida uma solução ótima. Desta forma a heurística implementada no GUROBI© visa reduzir o tempo de processamento computacional garantindo a otimalidade da solução obtida.

2.2. Heurística

Heurísticas segundo Glover & Kochenberg (2003) são estratégias de soluções que combinam qualidade de solução com esforço computacional necessário para obtê-la, ou seja, buscam uma solução satisfatória para os problemas com tempo e esforço computacional aceitável. Já as meta-heurísticas são consideradas superiores, também solucionando problemas de otimização, dado que elas controlam as heurísticas da busca local com o objetivo de melhorar o modo como determinar as soluções, explorando o espaço da vizinhança para evitar os ótimos locais (Cunha, 2006).

Segundo Li & Chen (2010), o *Job Scheduling* é um problema de otimização combinatória, e pode ser classificado como *NP-hard*, isso significa que não existe um algoritmo que encontre a solução ideal em tempo polinomial, assim como o *Flow Shop*. Ainda sim sendo bastante utilizados pelas empresas.

Em geral as empresas não podem esperar muito tempo por soluções, por isto que existe a necessidade da utilização de meta-heurísticas para encontrar soluções de otimalidade em tempos aceitáveis gerando sempre assim melhorias contínuas. Os algoritmos agem guiando o planejador nas suas tomadas de decisões, pois estes tentam encontrar graus de correspondência entre os produtos e as máquinas e assim, otimizar os tempos de produção.

3. Resultados Obtidos

Inicialmente foi realizada a implementação do algoritmo para a resolução dos problemas propostos, registrando seus respectivos tempo de resolução e os valores da função objetivo, que são seus tempos de produção. Sendo realizado para 16 instâncias de teste, sendo 8 *Job Shop* e as de mais de *Flow Shop*, ambas disponíveis em OR-LIBRARY (2018) que é uma coleção de conjuntos de dados de teste para uma variedade de problemas de Pesquisa Operacional.

Em seguida se iniciou a ordenação destas instâncias de teste, com o objetivo de diminuir o seu tempo computacional necessário para resolução. Vale ressaltar que foi estabelecido que o programa seria interrompido com 3.600 segundos, sendo a nossa função objetivo o melhor valor encontrado durante este período.

Tabela 1 – Ordenação das Instâncias de Teste

	Job Shop	**FO**	**Tempo [s]**		***Flow shop***	**FO**	**Tempo [s]**
Normal		265	1	Normal’		47875	3600
Decrescente	6X6	270	0	Decrescente	11x5	45895	3601
Crescente		265	0	Crescente		48679	3601
Normal		4832	1100	Normal		54687	3600
Decrescente	10x5	4712	2000	Decrescente	12x5	56381	3600
Crescente		4777	2685	Crescente		58470	3600
Normal		10602	3600	Normal		66330	3600
Decrescente	10x10	10881	3600	Decrescente	14x4	69683	3600
Crescente		10770	3600	Crescente		66598	3600

Normal		9006	3600	Normal		15841	3600
Decrescente	15x5	8681	3600	Decrescente	20x5	16975	3600
Crescente		8897	3600	Crescente		15821	3600
Normal		16651	3600	Normal		59988	3600
Decrescente	20x5	16174	3532	Decrescente	30x10	56623	3600
Crescente		15576	3600	Crescente		77321	3600
Normal		13367	3600	Normal		inf	3593
Decrescente	20x15	13675	3600	Decrescente	50x10	217400	3585
Crescente		13825	3600	Crescente		inf	3591
Normal		48266	3600	Normal		inf	3600
Decrescente	30x10	47470	3600	Decrescente	75x20	inf	3600
Crescente		45492	3592	Crescente		inf	3600
Normal		136381	3592	Normal		inf	3600
Decrescente	50x10	140239	3522	Decrescente	100x10	inf	3600
Crescente		Inf	3584	Crescente		inf	3600

Fonte: Autoria própria (2018)

Como se pode observar por se tratar de problemas com grande magnitudes, poucos apresentaram o tempo computacional menor do que 3600 segundos, entretanto, os que apresentaram e foi possível a comparação em relação ordenação crescente e decrescente, foi notado que em alguns casos a FO (função objetivo) era melhorada, porém em outros a mesma não apresenta melhorias, o mesmo se observou para o tempo de resolução que em certos casos foi melhorado com a ordenação crescente, em outros com a decrescente, e ainda alguns sem nenhuma alteração.

O próximo passo foi à utilização da heurística generalizada, para teste foi utilizado uma das instâncias de teste do OR-LIBRARY do *Job Shop* e outra do *Flow Shop*, para que fosse identificada a melhor porcentagem dessa heurística na resolução do problema, lembrando que o GUROBI© utiliza normalmente 5% de heurística, então foram realizados os testes a cada aumento gradual de 10% na utilização da heurística generalizada.

Tabela 2 – Teste com o aumento da Heurística Generalizada

	Flow Shop	**FO**	**Tempo [s]**		***Job shop***	**FO**	**Tempo [s]**
Normal		47875	3600	Normal		4832	1100
10%		47400	3600	10%		4832	3346
20%		47400	3600	20%		4832	1324
30%		47400	3600	30%		4832	2556
40%		47400	3600	40%		4832	3600
50%	11x5	47400	3600	50%	10x5	4832	3600
60%		47400	3600	60%		4832	3600
70%		47400	3600	70%		4832	3600
80%		47400	3600	80%		4832	3600
90%		47400	3600	90%		4832	3600
100%		47400	3600	100%		4832	3255
0%		48792	3600	0%		4832	1010

Fonte: Autoria própria (2018)

À medida que se aumentava a porcentagem de resolução com heurística o tempo computacional para resolução aumentava em grandes proporções, com isso não era possível se comparar este tempo, pois o máximo estabelecido foram os 3.600 segundos. Então para que se tivesse uma melhor ideia do funcionamento da heurística generalizada para esses problemas de pesquisa operacional em questão, foi realizado o teste em duas instâncias de teste de quantidades de tarefas e maquinam iguais, obtendo-se os seguintes resultados:

Tabela 3 – Comparação da heurística generalizada com instâncias de testes com a mesma quantidade de tarefa e máquina

Problema	**Porcentagem de Heurística Utilizada**	**FO Corrente**	**Limite Inferior**	**GAP**	**Nó Atual**	**Tempo**	**FO Final**
Flow Shop	5%	73537	17205.0232	76.6%	70	50	73537
	10%	86843	17175.6289	80.2%	314	50	82111
	20%	79959	17152.9819	78.5%	-	50	79069
	30%	79496	17133.9225	78.4%	-	53	79496
	40%	83278	17108.4816	79.5%	-	51	80471
	50%	83278	17108.4816	79.5%	-	52	80471

Job Shop	5%	-	16630.9456	-	-	40	16631
	10%	62072	16617.6298	73.2%	-	40	51327
	20%	-	16522.7437	-	-	43	16565
	30%	-	16522.7437	-	-	42	16565
	40%	-	16522.7437	-	-	42	16565
	50%	-	16459.9032	-	-	42	16519

Fonte: Autoria própria (2018)

É possível perceber que à medida que se aumenta a porcentagem heurística utilizada, são encontras algumas melhoras. No problema de pesquisa operacional *Flow Shop*, o nó no qual se encontra durante a resolução está à frente quanto à porcentagem de heurística utilizada é maior. No problema *Job Shop* é notável que com 50% de heurística utilizada se obtém uma função objetivo melhor do que com 5%.

Entretanto, não foram encontradas melhoras tão significativas, por se tratar de uma heurística de uso geral, sendo aplicável a diversos tipos de problemas da área de pesquisa operacional. A mesma foi desenvolvida com foco em problemas de caixeiro viajante e mineração, onde se tem melhoras ainda mais significativas em relação à diminuição de restrições e variáveis.

Nos problemas de pesquisa operacional, *Job Shop* e *Flow Shop* encontramos melhoras, mas não na magnitude dos problemas para o qual a heurística generalizada foi desenvolvida. Tendo em vista isso esse fato foi realizado o teste com 0% de utilização de heurística para resolução.

Tabela 4 – Resultado sem uso da Heurística Generalizada

Flow shop				*Job shop*			
Normal	11x5	47875	3600	Normal	6x6	265	1
0%		48792	3600	0%		265	1
Normal	12x5	54687	3600	Normal	10x5	4832	1100
0%		58736	3600	0%		4832	1010
Normal	14x4	66330	3600	Normal	10x10	10602	3600
0%		71324	3600	0%		10900	3600
Normal	20x5	15841	3600	Normal	15x5	9006	3600
0%		17367	3600	0%		9460	3600
Normal	30x10	59988	3600	Normal	20x5	16651	3600
0%		70374	3600	0%		17365	3600
Normal	50x10	Inf	3593	Normal	20x15	13367	3600
0%		Inf	3599	0%		14104	3600
Normal	75x20	Inf	3600	Normal	30x10	48266	3600
0%		Inf	3558	0%		51007	3600
Normal	100x10	Inf	3600	Normal	50x10	136381	3592
0%		Inf	3600	0%		132290	3600

Fonte: Autoria própria (2018)

Observando os dados obtidos com 0% de heurística em relação ao normal de apenas 5% se nota que nem o tempo computacional de resolução nem a função objetivo sofreram grandes alterações, com exceção do *Job Shop* de 10 tarefas e 5 máquinas, porém o mesmo obteve menor tempo quando resolvido utilizando a ordenação decrescente.

Após os métodos testados para a resolução e não tendo sido encontrada melhora significativa, o próximo passo foi o desenvolvimento de uma heurística construtiva para o problema *Job Shop.* A lógica de resolução da heurística se baseia em alocar em cada máquina a tarefa que ela deve processar em ordem, se iniciando pela que leva menor tempo de processamento,

respeitando a ordem de operação necessária para a produção. Então foi utilizada a heurística para encontrar a solução inicial de cada instância de teste dos problemas de *Job Shop* e *Flow Shop.*

Figura 4 – Pseudocódigo Heurística

```
Heurística Construtiva ()
1   Backup dos dados de entrada
2   Selecionar a Tarefa de menor tempo de execução para que seja alocada primeiro;
3           Se (Posição de memória da Máquina está vazia) então
4                   Máquina <- Tarefa;
5                           Se (É a primeira Tarefa da Máquina) então
6                                   Tempo Inicial <- 0;
7                           Senão
8                                   Tempo Inicial da Tarefa de menor tempo na Máquina <- Tempo
9   Final da Tarefa de menor tempo anterior na Máquina atual;
10                                  Tempo Final da Tarefa  de menor tempo na Máquina  <- Tempo
11  Inicial da Tarefai de menor tempo na Máquina  + Tempo execução Tarefa de menor tempo na
12  Máquina:
13                                  Se (Não é a primeira Tarefa da Máquina) então
14                                          Tempo Inicial da Tarefa de menor tempo na Máquina <-
15  Tempo Final da Tarefa anterior na Máquina;
16                                  fimSe;
17                                          Se (Tempo Final da Tarefa atual de menor tempo na
18  Máquina anterior > Tempo Final da Tarefa anterior à Tarefa de menor tempo na Máquina atual)
19  então
20          Tempo Inicial da Tarefa de menor tempo na Máquina <- Tempo Final da Tarefa atual de
21  menor tempo na Máquina anterior;
22                                          Senão
23          Tempo Inicial da Tarefa de menor tempo na Máquina <- Tempo Final da Tarefa anterior a
24  Tarefa de menor tempo na Máquina atual;
25                                          fimSe;
26          Atualizar o valor de menor tempo para não ser selecionado novamente;
27                  fimSe;
28          FO <- Soma os Tempos Finais de cada Tarefa em sua última Máquina a ser processada;
29  Fim Heurística Construtiva;
```

Fonte: Autoria própria (2018)

4. Considerações finais

Os problemas que foram propostos não tiveram melhorias reais em relação à ordenação, tendo muitas variações o que nos mostra que a ordenação nestes casos não é vantajosa e inconclusiva.

Com o aumento da porcentagem de heurística generalizada constatou-se que o tempo para a resolução dos problemas aumentava à medida que ela era aumentada, ou seja, não era viável. A resolução com 0% de heurística generalizada mostrou que o tempo pode até diminuir em certos casos, mas não se trata de algo tão relevante visto que a função objetivo (FO) tende a ser afetada, obtendo um valor maior do que o que era encontrado com 5% de heurística generalizada.

Após essas análises se pode constatar que como a ordenação e a heurística generalizada não dão resultados conclusivos em relação à otimização do tempo computacional para resolução e o mesmo se aplica em relação a FO.

Como a heurística específica desenvolvida para a resolução dos problemas aqui tratados obtemos uma solução inicial e então se torna possível partir para a utilização de meta-heurísticas visando a minimização do tempo de produção e do tempo de processamento. Será utilizada para tal a *Variable Neighborhood Search* (VNS), este método ao contrário de outras buscas locais avalia uma vizinhança que pode ser maior ou menor dependendo da precisão desejada da solução, não seguindo uma única trajetória.

O princípio do VNS é simples, ele visa criar conjuntos de soluções e combiná-las entre si para chegar a melhor solução possível. Ele age explorando as soluções através de trocas sistemáticas de estruturas de vizinhança, e focaliza a busca em torno de uma nova solução, a qual é alcançada somente quando um movimento de melhora é identificado (MLADENOVIC & HANSEN, 1997). Com isso será possível testar um novo método de solução para encontrar o resultado esperado, mas ainda não obtido.

REFERÊNCIAS

ARENALES, M., ARMENTANDO, V., MORABITO, R., YANASSE, H. (2007). **Pesquisa Operacional**. Rio de Janeiro: Elsevier

BARBARA, D. (2000), **An introduction to cluster analysis for data mining**. http://wwwusers.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf [acessado em 02 de abril de 2011].

CUNHA, C. B. (2006). **Contribuição à Modelagem de Problemas em Logística e Transportes**. Tese (livre docência), Escola Politécnica 1. ed. São Paulo: Universidade de São Paulo. 315 p.

GLOVER, F.; KOCHENBERG, G. A. (2003). **Handbook of Metaheuristics. International Series in Operations Research & Management Series**, Kluwer's International Series, Stanford University, 556p.

HAX, A & CANDEA, D. (1984). **Production and Inventory Management**, Englewood Cliffs, N.J., PrenticeHall

JAIN, A S, MEERAN, S. **Deterministic job-shop scheduling: past, present and future**.

Departament of Applied Physics and Eletronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, UK, DD1 4HN. 1998.

LUSTOSA, L. J. et al. (2008) **Planejamento e controle da produção**. Rio de Janeiro: Elsevier.
MLADENOVIC, N. E HANSEn, P.(1997), **Variable neighborhood search**. *Computers and Operations Research*, 24, 1097-1100.

REIS, J. A. E CUNHA, C. B. (2010) **Uma heurística baseada em VNS para o problema bidimensional de binpacking com frota heterogênea**. In: *XVI PANAM*, 15-18 Julho, 2010 – Lisboa, Portugal.

SCOFIELD, W. C. L. (2002). **Aplicação de Algoritmos Genéticos ao Problema Job-Shop.** Monografia (Graduação em Ciência da Computação) – UFOP, Ouro Preto.
TAHA, Hamdy A. (2008) **Pesquisa operacional: uma visão geral /** Hamdy A. Taha ; tradução Arlete Simille Marques; revisão técnica Rodrigo Arnaldo Scarpel. – São Paulo: Pearson Prentice Hall.

TUBINO, Dalvio Ferrari (2009**) Planejamento e controle da produção: teoria e prática /** Dalvio Ferrari Tubino. - 2. ed. – São Paulo: Atlas.