UMA PROPOSTA BASEADA EM META-HEURÍSTICAS PARA SOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE MÁQUINAS NÃO-RELACIONADAS

Jhonatan Fernando de Oliveira

jhonatanfoliveira@gmail.com

UEMG - Universidade do Estado de Minas Gerais

Av. Paraná, 3001, Jardim Belvedere, 35501-170, Divinópolis, Minas Gerais, Brasil

Thiago Magela Rodrigues Dias

Gray Farias Moita

thiago@div.cefetmg.br

gray@dppg.cefetmg.br

CEFET-MG – Centro Federal de Educação Tecnológica de Minas Gerais

Resumo. *Este trabalho visa propor uma solução para o problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependentes da máquina e da sequência. O objetivo é minimizar a soma ponderada dos atrasos na execução de tais tarefas. É proposto o desenvolvimento de algoritmos para a resolução do problema baseados nos princípios de funcionamento do GRASP, Simulated Annealing e Iterated Local Search*

Palavras-chave: *Sequenciamento de Máquinas, Máquinas Paralelas, Heurísticas.*

1 INTRODUÇÃO

O problema de sequenciamento está totalmente associado à definição de recursos para a execução de tarefas seguindo as restrições definidas com o objetivo de executar todas as tarefas no menor tempo possível. Esse tipo de problema é encontrado em praticamente todas as áreas da ciência, mas é na área da indústria que este cenário ganha foco. Sendo assim o planejamento da produção deve apresentar todas as tarefas a serem executadas e quais os prazos de cada uma, acarretando penalidades no caso de atraso (Ribeiro, 2010).

Para Paula (2008), este cenário, por ser intimamente relacionado à qualidade do produto final, o estágio de conformação é a parte mais crítica da produção. Além disso, o plano de produção da fábrica é feito de acordo com o planejamento do estágio de conformação. Logo, que este é definido, o planejamento dos outros estágios é ajustado de tal forma que os objetivos definidos sejam atendidos. Por essa razão, a maior parte do esforço do planejamento é investida no estágio de conformação.

O Problema de Sequenciamento de Máquinas Paralelas não relacionadas com tempo de preparação dependentes da Máquina e da Sequência, agora em diante denotado por PSMPMS, consiste em sequenciar e determinar o momento e em qual máquina as tarefas devem ser executadas, com o objetivo de minimizar a soma ponderada dos atrasos na produção de tais tarefas.

É comum encontrar em um setor de produção, máquinas que apresentam rendimentos diferentes, mesmo executando a mesma tarefa ou em tarefas com características específicas, que necessitam ajustes especiais nas máquinas onde estas deverão ser executadas. Para a resolução de tal problema, neste trabalho são apresentados algoritmos implementados com os princípios de funcionamento do GRASP, *Simulated Annealing* e *Iterated Local Search*, os quais foram utilizados para a geração da solução inicial e do refinamento para apresentação de melhores resultados.

O restante deste trabalho está organizado como segue. Na Seção 2 são apresentados trabalhos relacionados. Na Seção 3 faz-se uma descrição detalhada do problema tratado. Já a formulação da programação matemática indexada no tempo é descrita na Seção 4, bem como as heurísticas implementadas. Na Seção 5 são apresentados os resultados encontrados e por fim, são descritas as conclusões.

2 TRABALHOS RELACIONADOS

Apesar de ser um problema que se encontra em diversas áreas da ciência, o PSMPMS é difícil de ser revolvido. Por esse motivo o número de trabalhos na literatura que trata sobre esse assunto vem crescendo nos últimos anos de forma significativa.

Em Mazzini (1998) realizou-se a programação das tarefas em máquinas paralelas idênticas minimizando o tempo de atraso em relação as suas datas de entregas, sendo o tempo de preparação entre tarefas dependente da sequência. Para a resolução do problema optou se por utilizar Algoritmos Genéticos e Busca por Espalhamento para se obter soluções de boa qualidade.

Em Paula (2008) se realiza a programação das tarefas em máquinas paralelas não relacionais com o tempo de preparação dependente da sequência. O autor desenvolveu as heurísticas construtivas *Earliest Due Dates* (EDD) e o *Nawaz-Enscore-Ham* (NEH). O EDD verifica todas as tarefas e as ordena em ordem crescente de data de entrega e o NEH verifica todas as tarefas e as posicionam na posição que melhor resultado apresenta. As heurísticas de Busca Local desenvolvidas foram *Greed Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS), além de uma híbrida que utilizou GRASP e VNS. Para Paula (2008), NEH se mostrou vantajoso sobre os outros métodos construtivos. Da mesma forma, a busca local baseada na união das duas vizinhanças consideradas se mostrou superior às outras. Todas as heurísticas propostas se mostraram superiores às heurísticas encontradas na literatura, encontrando soluções de melhor qualidade para a grande maioria das instâncias. O VNS, entretanto, apresentou a convergência mais rápida, já que é uma busca local bastante intensa e poderosa.

3 DESCRIÇÃO DO PROBLEMA

Tendo em vista a proposta de uma solução eficiente para o problema de PSMPMS, neste trabalho, tal problema possui as seguintes características:

As máquinas devem processar um conjunto J de n de tarefas;

Associado a cada tarefa $j \in J$ está:

Um tempo de processamento Tj dependente da máquina;

Uma data de entrega dj;

Um peso wj;

Há atraso de uma tarefa $j \in J$ quando seu processamento é concluído depois de Tj;

A máquina executa no máximo uma tarefa por vez e, uma vez iniciado o processamento de uma tarefa, não é permitida a sua interrupção;

Todas as tarefas estão disponíveis para processamento na data 0;

Entre duas tarefas i e $j \in J$ consecutivas, é necessário um tempo Sij de preparação da máquina, chamado tempo de setup. O tempo de preparação da máquina para a primeira tarefa é dado por $S0i$;

O intuito é encontrar uma sequência de tarefas com suas respectivas máquinas para execução de forma que todas as tarefas possam ser executadas em um menor tempo de atraso possível.

4 DESENVOLVIMENTO

Uma solução para o PSMPMS com n tarefas e m máquinas é representada por uma matriz de *m* por *n*, onde cada posição *j* = 1,2,.., *n* indica a ordem de execução daquela tarefa naquela máquina. Por exemplo, dada a sequencia de tarefas t = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} e as máquinas disponíveis *m* = {1,2,3} um solução possível poderia ser: *s* = [0] {1,5,7,8}

[1] {4,10,9}

[2] {2,3,6}

Para o PSMPMS com 10 tarefas e 3 máquinas, a tarefa 1 é a primeira a ser realizada e a tarefa 8 a última na máquina 0.

A função de avaliação é dada pelas Fórmulas 1 e 2:

$$T = \sum_{j \in J} w_j T_j = \sum_{j \in J} w_j \max\left(c_j - d_j, 0\right), \tag{1}$$

$$c_j = \sum_{i \in \text{Predecessors}(j) \cup j} p_i + s_{\text{Previous}(i),i} \tag{2}$$

Em (1) é apresentado a função objetivo que se dá pelo somatório dos pesos multiplicado pelo tempo de processamento final de cada tarefa. Sendo que o tempo de processamento final é dado pela função (2). Essa função realiza o somatório dos tempos de processamento de todas as tarefas anteriores com seus respectivos tempo de preparação acrescido do tempo de processamento da tarefa atual. Considera-se que a função objetivo irá utilizar somente as tarefas que não obtiveram atraso no seu processamento.

Os movimentos utilizados para que o espaço de soluções fosse explorado foram: troca da ordem de processamento de duas tarefas na mesma máquina, realocação de uma tarefa para outra posição e a realocação das demais tarefas na mesma máquina, troca da ordem de processamento de duas tarefas em máquinas distintas, realocação de uma tarefa para outra posição e a realocação das demais tarefas em máquinas distintas, três trocas de tarefas simultâneas na mesma máquina, três realocações de tarefas simultâneas na mesma máquina, uma realocação e uma troca de tarefas na mesma máquina.

4.1 Algoritmo Proposto

O algoritmo proposto neste trabalho visa utilizar estratégias de algumas metaheurísticas para apresentar melhores valores para o PSMPMS. Na implementação do algoritmo foram utilizados dois tipos de heurísticas, as Construtivas e as de Busca Local.

Uma heurística construtiva tem por objetivo construir uma solução, elemento por elemento. A forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, a qual, por sua vez, depende do problema abordado. Além disso, o autor define que um método de busca local pode ser visto como um procedimento que

percorre um caminho em um grafo não-orientado **G** = (**S,E**), onde **S** representa o conjunto de soluções s do problema e **E** o conjunto de arestas (**s,s'**), com **s'** ∈ **N(s)**.

Para a fase de geração da solução inicial as seguintes heurísticas foram aplicadas: *Earliest Due Date* (EDD), *Weight Shortest Processing Time* (WSPT) e Fase de Construção GRASP com EDD .

Já na fase de refinamento as seguintes heurísticas foram aplicadas: *Simulated Annealing* (SA) e *Iterated Local Search* (ILS).

4.1.1 *Earliest Due Date EDD*

Earliest Due Date, tem como objetivo verificar todas as tarefas existentes e ordená-las de acordo com suas datas de entrega, visando as que possuem a data de entrega mais próxima. Após determinada qual tarefa possui a menor data de entrega é verificado em qual máquina será alocada tal tarefa. Para isso utilizou-se a ideia de balanceamento, ou seja, a máquina que apresentar o menor tempo total de processamento receberá a próxima tarefa.

O algoritmo da fase de construção de uma solução com EDD é apresentado na Figura 1.

```
Procedimento EDD();
1   v ← Ø;
2   Inicialize o conjunto C de tarefas candidatas;
3   Inicialize outro conjunto D de tarefas candidatas;
4   enquanto D ≠ Ø faça
5     tarefaEscolhida = 0;
6     enquanto C ≠ Ø faça
7           se Di < Cj entao
8               tarefaEscolhida = d;
9           Atualize o conjunto C de tarefas candidatas;
10    fimenquanto;
11    melhorMaquina <- máquina que possui menor tempo de processamento.
12    tarefaEscolhida é inserido na melhorMaquina
13    v ← v ∪ { melhorMaquina };
14    Atualize o conjunto D de tarefas candidatas;
15  fimenquanto;
16  Retorne v;
          Fim EDD;
```

Figura 1: Construção de uma Solução com EDD

4.1.2 *Weight Shortest Processing Time*

Já *Weight Shortest Processing Time*, tem como objetivo verificar todas as tarefas existentes e ordená-las de acordo com seus pesos, visando as que possuem o maior peso. Após determinada qual tarefa possui o maior peso, é verificado em qual máquina será alocada essa tarefa. Para isso utilizou-se a ideia de balanceamento, ou seja, a máquina que apresentar o menor tempo total de processamento receberá a próxima tarefa.

O algoritmo da fase de construção de uma solução utilizando WSPT é praticamente o mesmo apresentado na Figura 1, a única diferença é que será levado em consideração o maior peso e não a menor data de entrega.

4.1.3 *Greed Randomized Adaptive Search Procedure* com *Earliest Due Date*

Na proposta desta heurística, o intuito é utilizar a fase de construção do GRASP com a funcionalidade do EDD, colocando dessa forma um percentual de aleatoriedade no comportamento do EDD. O princípio de funcionamento desse algoritmo é selecionar as 5 tarefas que possuem a menor data de entrega. Após escolher uma tarefa aleatoriamente e alocá-la na máquina que possui o menor tempo de processamento total.

O pseudocódigo da fase de construção de uma solução utilizando GRASP com EDD é apresentado na Figura 2.

```
Procedimento GRASP();
1   v ← Ø;
2   Inicialize o conjunto C de tarefas candidatas;
3   enquanto C ≠ Ø faça
4     LRC = { 5 tarefas com menor data de entrega };
5     Selecione, aleatoriamente, uma tarefa t ∈ LRC;
6     melhorMaquina <- máquina que possui menor tempo de processamento.
7     t é inserido na melhorMaquina
8     v ← v ∪ { melhorMaquina };
9     Atualize o conjunto C de tarefas candidatas;
10  fimenquanto;
11  Retorne v;
        Fim GRASP;
```

Figura 2: Construção de uma Solução GRASP com EDD

4.1.4 *Simulated Annealing*

Simulated Annealing, trata-se de uma técnica de busca local probabilística, que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como recozimento.

Como essa heurística se baseia no processo de resfriamento de um conjunto de átomos, é necessário informar ou calcular a temperatura inicial antes de iniciar o processo. Para o PSMPMS a temperatura inicial foi calculada em um algoritmo desenvolvido que percorre o espaço de solução executando movimentos aleatórios até que obtenha certo número de soluções com melhora, ou seja, que depois de percorrer o espaço de solução durante uma quantidade de iterações (valor configurável) apresente um número (valor configurável) de soluções com melhora.

Depois de obtido a temperatura inicial o resfriamento é inicializado definindo valores para os seguintes parâmetros:

- Número de iterações sem melhora;
- Valor da temperatura final;
- Temperatura inicial.

Para percorrer o espaço de solução os seguintes movimentos foram definidos:

(0) Realocação de tarefas na mesma máquina;

(1) Troca de tarefas na mesma máquina;

(2) Realocação de tarefas em duas máquinas;

(3) Troca de tarefas entre máquinas;

No movimento (0) é escolhida aleatoriamente uma máquina e depois aleatoriamente uma tarefa e logo em seguida essa tarefa é enviada para o fim da fila dessa máquina e realocada todas as outras tarefas para frente.

Já no movimento (1) é escolhida aleatoriamente uma máquina e depois aleatoriamente duas tarefas distintas. Logo em seguida é realizada a troca de posições entre essas duas tarefas.

No movimento (2) é escolhida aleatoriamente duas máquinas distintas e depois aleatoriamente uma tarefa na primeira máquina escolhida e outra tarefa na segunda máquina escolhida. Logo em seguida é realizada a realocação nas duas máquinas.

Por fim, no movimento (3) é escolhida aleatoriamente duas máquinas e depois é escolhida aleatoriamente duas tarefas sendo uma de cada máquina escolhida. Logo em seguida é realizada a troca entre essas duas tarefas.

O pseudocódigo do *Simulated Annealing* é apresentado na Figura 3.

```
Procedimento SA(f(.),N(.),α, SAmax; T0, s)
1   s* ← s;        {Melhor solução obtida até o momento}
2   IterT ← 0;     {Número de iterações na temperatura}
3   T ← T0;        {Temperatura corrente}
4   enquanto (T > 0) faça
5     enquanto (IterT < SAmax) faça
6       IterT ← IterT + 1;
7       Escolhe aleatoriamente um movimento (0 a 3);
8       Gere um vizinho qualquer s' € N(s);
9       Δ= f(s') - f(s);
10      se (Δ < 0) então
11        s ← s';
12        se (f(s') < f(s*)) então
13          s* ← s';
14        fimse
15      senão
16        Tome x € [0,1];
17        se (x < e^-Δ/t) então
18          s ← s';
19        fimse
20      fimse
21    fimenquanto
22    T ← α x T;
23    IterT ← 0;
24    fimenquanto
25    s ← s*;
26    Returne s;
Fim SA;
```

Figura 3: Construção de uma Solução com SA

4.1.5 *Iterated Local Search*

Para Lourenço et. al (2003), o método *Iterated Local Search* (ILS) é baseado na ideia de que um procedimento de busca local, pode ser melhorado, gerando-se novas soluções de partida, soluções estas obtidas por meio de perturbações na solução ótima local. Sendo assim para essa heurística foram desenvolvidos movimentos para que as perturbações geradas consigam fugir de um ótimo local e permita o algoritmo explorar diferentes regiões do espaço de solução. Na Figura 4 o pseudocódigo do *Iterated Local Search* é apresentado.

Os níveis de perturbação e os movimentos gerados foram os seguintes:

- Nível de perturbação I: Para esse nível de perturbação foram realizadas duas trocas de tarefas entre máquinas. Isso consiste em escolher aleatoriamente duas máquinas e uma tarefa de cada máquina e em seguida realizada a troca entre as tarefas.
- Nível de perturbação II: Para esse nível de perturbação foram realizadas duas realocações de tarefas entre máquinas. Isso consiste em escolher aleatoriamente duas máquinas e uma tarefa de cada máquina e em seguida inserir no final de cada máquina as tarefas invertidas.
- Nível de perturbação III: Para esse nível de perturbação foi realizado uma troca de tarefas e uma realocação de tarefas entre máquinas.
- Nível de perturbação IV: Para esse nível de perturbação foi realizado três movimentos de troca de tarefas entre máquinas.

```
Procedimento ILS(s, iter, ILSmax)
1   s0 ← s;  {Melhor solução obtida até o momento}
2   s ← Descida_Randomica(s0);
3   enquanto (iter < ILSmax) faça
4       iter++;
5       s' ← Perturbacao(historico, s);
6       s'' ← Descida_Randomica(s');
7       s  ← Criterio_Avaliacao(s,s'',historico);
8   fimenquanto
Fim ILS;
```

Figura 4: Construção de uma Solução com ILS

Para o bom funcionamento do ILS o mesmo necessita de um método de Busca Local para percorrer o espaço de solução após os movimentos realizados, sendo assim o método da Descida Randômica foi utilizado.

Segundo Silva et. al (2009), o método da Descida Randômica consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente; não o sendo, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é interrompido após um número fixo de iterações sem melhora no valor da melhor solução obtida até o momento.

Os movimentos gerados no método da Descida Randômica foram os mesmo utilizados no método SA:

(0)Realocação de tarefas na mesma máquina;
(1)Troca de tarefas na mesma máquina;
(2)Realocação de tarefas em duas máquinas;
(3)Troca de tarefas entre máquinas;

O pseudocódigo da Descida Randômica é apresentado na Figura 5.

```
Procedimento RandomicoDescida(f.),N(.), IterMax, s);
1   Iter ← 0 {Contador de Iterações sem melhora}
2   s ← Descida_Randomica(s0);
3   enquanto (Iter < ILSmax) faça
4       Iter++;
5       Escolhe aleatoriamente um movimento (0 a 3);
6       Gere um vizinho qualquer s' € N(s);
7       se (f(s) < f(s')) entao
8           Iter ← 0;
9           s ← s';
10      fimse;
11  fimenquanto;
12  Retorne s;
Fim RandomicoDescida;
```

Figura 5: Construção de uma Solução com Descida Randômica

5 RESULTADOS

Para testar o modelo proposto foram gerados problemas-teste baseados no trabalho de Cicirello (2003), que foi utilizado para definir a estrutura do arquivo de dados e seus valores. O arquivo contém a definição da quantidade de tarefas, o tempo de processamento, tempo de setup depende da sequência de tarefas, peso e data de entrega de cada tarefa.

Foram gerados conjuntos de problemas-teste com 60 tarefas e até 5 máquinas. Na Tabela 1 são apresentados os valores obtidos, além de compará-los com os resultados do trabalho de Cicirello (2003), eles são apresentados de acordo com os métodos utilizados na geração da solução inicial e na busca local SA.

Tabela 1 – Resultados obtidos com a busca local SA

Máquinas	Tarefas	Solução Inicial	Valor conhecido	SA	Média SA	Gap (%)	Melhoria (%)	Tempo Médio (s)
1	60	EDD	978.00	777.00	1001.83	2.47	-20.55	14.53
1	60	WSPT	978.00	842.00	979.93	0.19	-13.91	15.12
1	60	GRASP	978.00	810.00	1005.53	2.81	-17.18	15.20
2	60	EDD	978.00	0.00	0.00	100.00	-100.00	~ 0.00
2	60	WSPT	978.00	0.00	0.00	100.00	-100.00	~ 0.00
2	60	GRASP	978.00	0.00	0.00	100.00	-100.00	~ 0.00

Na Tabela 2 são apresentados os valores referentes à utilização da busca local ILS.

Tabela 2 – Resultados obtidos com a busca local ILS

Máquinas	Tarefas	Solução Inicial	Valor conhecido	ILS	Média ILS	Gap (%)	Melhoria (%)	Tempo Médio (s)
1	60	EDD	978.00	910.00	981.37	0.34	-6.95	7.40
1	60	WSPT	978.00	914.00	999.07	2.15	-6.54	7.39
1	60	GRASP	978.00	906.00	999.60	2.21	-7.36	7.50
2	60	EDD	978.00	0.00	0.00	100.00	-100.00	~ 0.00
2	60	WSPT	978.00	0.00	0.00	100.00	-100.00	~ 0.00
2	60	GRASP	978.00	0.00	0.00	100.00	-100.00	~ 0.00

Para realizar o cálculo do Gap a Fórmula 3 foi utilizada:

$$gap = \frac{(Media - ValorConhecido)}{ValorConhecido} \times 100\% \tag{3}$$

O valor do Gap representa o percentual de melhoria em relação à média dos valores encontrados. Já para o cálculo da melhoria, foi utilizada a Fórmula 4.

$$melhoria = \frac{(ValorMetodo - ValorConhecido)}{ValorConhecido} \times 100\% \tag{4}$$

Nesse caso o valor resultante representa o percentual de melhoria em relação ao melhor valor conhecido.

Como se pode observar, o objetivo de minimizar o atraso na execução das tarefas foi alcançado quando se utilizou duas máquinas, sendo assim não foi necessária a realização de testes com a quantidade de máquinas superior a 2 (dois).

6 CONCLUSÕES

Esse trabalho tratou do problema de sequenciamento de máquinas paralelas não-relacionadas com tempo de preparação dependente da máquina e da sequencia de tarefas. Com o desenvolvimento do algoritmo foi possível encontrar boas soluções com um gasto computacional não muito alto. Nos resultados apresentados mesmo o SA ter conseguido encontrar o menor valor, a sua média apresentou valores maiores do que o ILS, sendo assim os valores do Gap não foram tão bons quanto o ILS. Para mais de uma máquina os dois algoritmos conseguiram chegar ao menor valor possível com facilidade, tendo em vista que os algoritmos que geram a solução inicial conseguiam um valor muito próximo de 0 (zero). Já sobre o tempo computacional médio foi possível constatar que o ILS consegue gerar boas soluções mais rápido que o SA, gastando praticamente a metade do tempo que o SA consome.

REFERENCIAS

Cicirello, V. A. Weighted Tardiness Scheduling with Sequence-Dependent Setups: A Benchmark Library, In: Intelligent Coordination and Logistics Laboratory The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania. 2003.

Lourenço, H. R., Martin, O., Stützle, T. Iterated Local Search. In F.Glover and G. Kochenberger (eds), Handbook of Metaheuristics, p. 321 - 353, Kluwer Academic Publishers, Norwell, MA, 2003.

Mazzini, R. Estudo de Meta-Heurísticas Populacionais para a Programação de Máquinas Paralelas com Tempos de Preparação Dependentes da Seqüência e Datas de Entrega, Dissertação de Doutorado, Programa de Pós-Graduação em Engenharia Elétrica, UNICAMP, São Paulo, 1998.

Paula, M. R. Heurísticas para a Minimização dos Atrasos em Seqüenciamento de Máquinas Paralelas com Tempos de Preparação Dependentes da Seqüência, Dissertação de Mestrado, Programa de Pós-Graduação em Ciências da Computação, UFMG, Belo Horizonte, 2008.

Ribeiro, F. F. Técnicas Heurísticas Evolutivas Adaptativas Aplicadas ao Problema de Seqüenciamento em uma Máquina com Penalização por Antecipação e Atraso da Produção, In Anais do XII Simpósio de Pesquisa Operacional e Logística da Marinha, SPOLM, 2010.

Silva, M. S. A.; Mine, M. T.; Ochi, L. S.; Souza, M. J. F. Um algoritmo evolutivo h´ibrido para o problema de recobrimento de rotas com coleta de premios. Revista da Sociedade Brasileira de Redes Neurais (SBRN). 2009.