HEURÍSTICAS PARA MINIMIZAÇÃO DO *MAKESPAN* DE PROBLEMAS DINÂMICOS DE SEQUENCIAMENTO *FLOWSHOP* HÍBRIDO E FLEXÍVEL COM TEMPO DE SETUP DEPENDENTE COM EVENTOS DE QUEBRA DE MÁQUINAS

Neuma Eufrázio Braz Moreira
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Minas Gerais, Brasil
neumapuc@gmail.com

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Minas Gerais, Brasil
sergio@dppg.cefetmg.br

Moacir Felizardo de França Filho
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, Nova Gameleira, Belo Horizonte, Minas Gerais, Brasil
franca@des.dppg.cefetmg.br

RESUMO

Este trabalho trata do problema de sequenciamento de tarefas *Flowshop* Híbrido e Flexível com tempo de *setup* dependente da sequência denominado HFFS-SDST (*Hybrid Flexible Flowshop With Sequence Dependent Setup Time*). O problema HFFS-SDST tratado neste trabalho é estudado em sua formulação dinâmica, ou seja, com eventos de quebra de máquinas. Para resolver este problema, foi implementado um algoritmo baseado nas metaheurísticas GRASP e ILS. Para testar o algoritmo, foi utilizado um conjunto de instâncias disponibilizado na literatura. Os resultados do problema dinâmico foram comparados com os resultados do problema estático.

PALAVRAS CHAVE. Sequenciamento de tarefas, *Flowshop* Híbrido e Flexível, Metaheurísticas, Quebra de Máquinas, *Makespan*.

Área Principal: Sequenciamento de Produção, Engenharia de Produção, Sistemas de Produção

ABSTRACT

This work presents a study about Hybrid Flexible Flowshops with Sequence Dependent Setups Time, denoted HFFS-SDST. This problem considers that machines can breakdown and this characteristic is denoted Dynamic Scheduling. To solve this problem, an algorithm based on GRASP-ILS were implemented and tested using instances available in the literature. The results of the dynamic scheduling is compared to static dynamic.4

KEYWORDS. Scheduling, Hybrid and Flexible Flowshop, Metaheuristics, Machine Breakdown, Makespan.

Main Area: (inform by priority the area of the article because JEMS system makes the classification alfabeticaly)

1. Introdução

O sequenciamento de tarefas é uma formulação para tomada de decisões e otimização de recursos muito utilizada em empresas de manufaturas e prestação de serviços, segundo Pinedo (2008). Através do sequenciamento, é possível alocar recursos e tarefas em um dado período de tempo.

Schuster e Framinan (2003) afirmam que existe uma lacuna entre a teoria de solução dos problemas de sequenciamento apresentada nos trabalhos encontrados na literatura ao longo do tempo e as questões práticas em sequenciamento. Assim, uma forte tendência dos estudos de sequenciamento de tarefas na atualidade é trabalhar com casos práticos, que respondam às necessidades reais dos ambientes produtivos, em especial que envolvam o tratamento de questões associadas à mudanças em soluções de instâncias de problemas já em curso. A literatura classifica a análise de situações como a exposta acima como problemas de sequenciamento dinâmico, em oposição a problemas que não possuem o tratamento de alterações de suas configurações, denominados de problemas de sequenciamento estático.

Para Ouelhadj e Petrovic (2009), existem muitos trabalhos que se propõem a resolver os problemas de sequenciamento estático. Entretanto, este tipo de solução muitas vezes é impraticável no mundo real, que está sujeito a sofrer, a todo momento, perturbações no sequenciamento pré-estabelecido. Logo, um sequenciamento que é considerado viável pode se tornar inviável após a interferência de um evento real.

Atualmente, há uma tendência recente em desenvolver trabalhos que busquem se aproximar de ambientes operacionais reais, como no caso do *Flow shop* simples, criticado pois raramente há situações reais em que existe uma única máquina para cada operação (Naderi et al., 2010). Considerando esta tendência, este trabalho apresenta uma variação do *Flow shop*, proposta em Naderi et al. (2010), denominado *Flow shop* Híbrido Flexível com tempo de Setup Dependente da Sequência- HFFS - SDST (*Hybrid Flexible Flow Shop - Sequence Dependent Setup Time*). O critério de otimização considerado neste trabalho é o de minimizar o maior tempo de conclusão das máquinas, denominado *makespan* ($C_{\max}$). Além das características propostas no modelo de Naderi et al. (2010), o presente trabalho considera a possibilidade de quebra de máquinas, caracterizando o problema estudado como sequenciamento dinâmico. Portanto, o problema estudado é denominado HFFS-SDST dinâmico. Para estudar o cenário com quebra de máquinas foi avaliada a quebra durante o processamento das tarefas.

O restante desse trabalho está dividido da seguinte forma. Na seção 2 é feito um levantamento bibliográfico dos problemas de *flowshop*. Na seção 3 o problema é apresentado e exemplificado. O algoritmo proposto é descrito na seção 4. Na seção 5 são mostrados os resultados encontrados, enquanto a última seção conclui o trabalho.

2. Revisão da Literatura

O *Flowshop* consiste em um fluxo de tarefas unidirecional, contínuo e ininterrupto, em que tarefas devem ser processadas, na mesma sequência, em máquinas distintas disponíveis em série. Gupta e Tunc (1991), Pinedo (2008) e Naderi et al. (2010) definem o *Flowshop Híbrido*, que é uma variação do *Flowshop*, como um conjunto de estágios em série, sendo que cada estágio possui uma ou mais máquinas paralelas. O *Flowshop* Híbrido é aplicado em muitas situações práticas (Yaurima et al., 2009). Processos industriais diversos (automotivos, químicos, petróleo, tabaco, têxtil e papel, dentre outros) podem ser modelados como *Flowshop* Híbrido (Gholami et al., 2009). Ziaeifar et al. (2012) e Yaurima et al. (2009) apresentaram uma modelagem matemática do problema *Flowshop* Híbrido para linhas de montagem de placas de circuito. Os dois trabalhos tiveram como objetivo a minimização do *makespan* e utilizaram Algoritmos Genéticos.

O Problema de *Flowshop* Híbrido Flexível com tempo de preparação dependente da sequência (*HFFS-SDST*), considerado neste trabalho, é uma generalização do Problema de *Flowshop* Híbrido. O *HFFS-SDST* possui três características básicas, de acordo com Naderi et al. (2010): (i) a configuração híbrida, que está relacionada à existência de máquinas paralelas em cada estágio; (ii)

a flexibilidade, devido à possibilidade de pular estágios, ou seja, nem todas as tarefas necessitam de visitar todos os estágios; e (iii) o tempo de preparação dependente da sequência (SDST), que torna o problema muito mais complexo.

Segundo Naderi et al. (2010), existem diversas revisões de literatura do Problema de *Flowshop* Híbrido Flexível, mas muitas delas não consideram o tempo de preparação dependente da sequência de processamento. Para resolver o problema HFFS-SDST com o objetivo de minimização do *makespan*, Kurz e Askin (2004) faz uso da programação inteira. Já Kurz e Askin (2003) utilizam de regras de despacho e Naderi et al. (2010), além das regras de despacho, propõem a utilização da metaheurística *Iterated Local Search (ILS)* para solucionar o mesmo problema.

Segundo Lin et al. (1997), o sequenciamento dinâmico pode ser classificado como determinístico ou estocástico. O sequenciamento dinâmico determinístico prevê que uma nova tarefa pode chegar e o instante de chegada é previamente conhecido ou quando o período de indisponibilidade de um recurso é previamente conhecido, como em caso de manutenção preventiva de máquinas Gholami et al. (2009). Já no dinâmico estocástico o instante de chegada da nova tarefa é um evento estocástico. O sequenciamento estocástico pode ser decomposto em um série de sequenciamentos estáticos. Ouelhadj e Petrovic (2009) também trata o sequenciamento dinâmico como um conjunto de sequenciamentos estáticos, que podem ser resolvidos com algoritmos clássicos. Assim, o sequenciamento é executado e não é revisado até que surjam novas informações que irão gerar necessidades de alterações no horizonte de sequenciamento estabelecido.

Ouelhadj e Petrovic (2009), Liu et al. (2005) e Gourgand et al. (2013) classificam os eventos em tempo real que caracterizam o sequenciamento dinâmico em duas categorias: (i) Relacionados a recursos: dentro desta categoria estão as quebras de equipamentos, tempo de processamento subestimado, recursos humanos limitados, escassez de materiais, atraso na chegada das matérias primas, material defeituoso, dentre outros; (ii) Relacionados a tarefas: dentre os eventos relacionados às tarefas estão a chegada de tarefas urgentes, modificação da data de entrega, cancelamento de tarefas, reprocessamento de lotes, dentre outros. O sequenciamento dinâmico tratado neste trabalho está relacionado aos recursos.

Alcaide et al. (2002) argumenta que, durante o sequenciamento, algumas máquinas podem não estar disponíveis durante um certo período de tempo. Estas indisponibilidades das máquinas são denominadas "*Quebras*". Este autor propõe uma abordagem para minimizar $C_{\max}$ de problemas classificados como *Flowshop* sujeitos a quebras de máquinas. Nesta abordagem, os problemas de sequenciamento com quebras são convertidos em uma sequência finita de problemas de sequenciamento sem quebras. Para realizar esta conversão, considera-se que, em cada instante de tempo t, $m(t)$ representa o número de máquinas disponíveis, ou seja, que não estão quebradas. É este conjunto de máquinas disponíveis que determina o status do sistema no instante t.

Diversos autores apresentam formulações para o Problema *Flowshop* sob condições diversas de quebra de máquinas, com o Allahverdi e Mittenthal (1994), Safari e Sadjadi (2011) e Mirabi et al. (2013). Poucos, no entanto, se dedicam a estuda o problema de quebras de ma´quinas no Problema *Flowshop* Híbrido Flexível. Gholami et al. (2009) propõe a solução para o *Flowshop* Híbrido Flexível com Tempo de Setup Dependente da Sequência através de algoritmos genéticos. Este autor, assim como Alcaide et al. (2002) aborda o problema dinâmico como uma série de problemas estáticos.

3. Descrição do problema

O Sequenciamento Dinâmico está associado à análise da ocorrência de alterações que possam vir a acontecer, sob alguma perspectiva, no sequenciamento determinado para o instante de início de operação. Estas alterações podem ocorrer a qualquer instante de tempo durante a operação do sistema ou processo considerado. O Sequenciamento Dinâmico, conforme apresentado na seção 2, pode ser classificado como Determinístico ou Estocástico. O problema HFFS-SDST Dinâmico abordado pertence à classe de Sequenciamento Dinâmico Estocástico, pois considera a quebra de forma inesperada durante a execução do processamento.

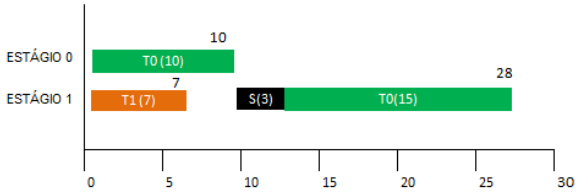

Figura 1: Exemplo das Restrições (vii) e (viii).

No problema HFFS-SDST dinâmico considera-se um conjunto $N = \{1, 2, 3, ..., n\}$ de tarefas, que devem ser processadas em um conjunto $M = \{1, 2, 3, ..., m\}$ de estágios. Para cada estágio i, sendo $i \in M$, existe um conjunto de $M_i = \{1, 2, 3, \ldots, m_i\}$ máquinas idênticas. Nem todas as máquinas em todos os estágios estão habilitadas para processar todas as tarefas. As definições a seguir estão associadas a este problema: (i) i: representa o estágio; (ii) j: representa a tarefa; (iii) (i, j): representa a operação; (iv) P_{ij}: representa o tempo de processamento da tarefa j no estágio i (operação (i, j)); (v) S_{ijk}: representa o tempo de preparação de uma tarefa j para uma tarefa k dentro de um estágio i; (vi) C_j: instante de conclusão da tarefa j (última operação); (vii) $C_{\max}$: representa o *makespan*, ou seja, o instante de término da última tarefa no sistema; (viii) F_j: conjunto de estágios que a tarefa j deve visitar, considerando que $1 \leq F_j \leq m$.

O objetivo do problema HFFS-SDST dinâmico é a minimização do *makespan* ($C_{\max}$). Algumas premissas devem ser consideradas: (i) todas as tarefas são independentes e estão disponíveis para serem processadas no instante 0 (zero); (ii) nem todas as máquinas estão disponíveis; (iii) cada máquina só pode processar uma tarefa de cada vez e uma tarefa só pode ser processada em uma máquina por vez; (iv) o processamento de uma tarefa pode ser interrompido; (v) a tarefa deve esperar uma máquina ser liberada, caso ela tenha que ser processada nesta máquina; (vi) cada estágio possui uma ou mais máquinas idênticas; (vii) nem todas as tarefas necessitam visitar todos os estágios; (viii) o tempo de preparação é dependente da sequência e não antecipativo, ou seja, a preparação da máquina exige que a tarefa esteja na máquina. Este requisito significa que a tarefa deve ser liberada no estágio anterior para que a preparação possa ser realizada.

A Figura 1 exemplifica as premissas (vii) e (viii). Neste exemplo, existem dois estágios (estágio 0 e estágio 1). A tarefa $T0$ visita os dois estágios e a tarefa $T1$ visita somente o estágio 1, ou seja, ela salta o primeiro estágio. Nesta mesma figura, pode ser observado que a preparação da tarefa $T1$ para a tarefa $T0$ só foi iniciada após a tarefa $T0$ ter sido liberada no estágio 0 (zero) e o tempo de preparação foi de 3 unidades de tempo. Neste exemplo, a última tarefa processada é a $T0$ e, portanto, o $C_{\max}$ é 28 unidades de tempo.

O cenário de quebra estudado neste trabalho considera que existe um sequenciamento planejado que está sendo executado e, durante a execução deste, uma máquina se torna indisponível. Devido a este fato, deverão ser avaliados os seguintes grupos de tarefas: (i) grupo de tarefas que tiveram o processamento concluído até o instante em que ocorreu o evento da quebra; tarefas que estavam no meio do processamento quando ocorreu o evento de quebra; e tarefas que ainda não foram processadas. A abordagem tratada neste cenário é considerada por Gholami et al. (2009) e Alcaide et al. (2002). A estratégia de resequenciamento utilizada é a apresentada por Novas e Henning (2010).

As tarefas que tiveram o seu processamento concluído, considerando todos os estágios que deveriam visitar, serão desconsideradas (TeAE). Já as tarefas que estavam sendo processadas na máquina que ficou indisponível deverão ter seu processamento reiniciado em outra máquina que esteja disponível e devem visitar todos os estágios que estavam programados para serem visitados (TeIP). As tarefas que estavam sendo processadas em alguma máquina que não ficou indisponível deverão finalizar o seu processamento na máquina que já estava processando-as e devem visitar todos os estágios conforme planejado (TeAE). Assim que as máquinas que ficaram disponíveis fi-

Tabela 1: Representação da Solução em forma de Matriz.

$$M = \begin{vmatrix} M_0 & T0 & T1 & -1 \\ M_1 & T2 & T1 & -1 \\ M_2 & T0 & -1 & -1 \end{vmatrix}$$

nalizarem o processamento que já haviam iniciado, elas se tornam disponíveis para aquelas tarefas que ainda não haviam iniciado o seu processamento ou para aquelas que haviam iniciado o processamento na máquina que ficou indisponível (TeNE). A partir do instante que ocorreu o evento, considera-se a existência de um novo problema estático.

4. Metodologia

4.1. Representação Computacional da Solução

Uma solução para o problema HFFS-SDST é representada por uma matriz M, em que as linhas representam as máquinas de todos os estágios e as colunas a sequência de processamento das tarefas dentro de cada máquina pertencente a cada estágio. A Tabela 4.1 mostra que existem quatro tarefas para serem processadas e quatro máquinas, sendo M_0 e M_1 máquinas paralelas idênticas pertencentes ao estágio um e M_2 e M_3 máquinas paralelas idênticas pertencentes ao segundo estágio. Nesta solução, no primeiro estágio, as tarefas $T1$, $T2$ e $T3$ são processadas na máquina M_0 e a tarefa $T4$ é processada na máquina M_1. Na máquina M_0, a tarefa $T1$ precede a $T2$ que, por sua vez, precede a tarefa $T3$. No segundo estágio, as tarefas $T2$ e $T3$ são processadas na máquina M_2 e a tarefa $T1$ é processada na máquina M_3. A tarefa $T4$ não aparece em nenhuma das máquinas do segundo estágio, pois não necessita ser processada neste estágio (flexibilidade). As posições preenchidas com o valor -1 indicam que não existe nenhuma tarefa a ser processada na máquina nesta sequência.

4.2. Estrutura de Vizinhança

Os tipos de movimentos utilizados para este trabalho foram: (i)] Troca de tarefas na mesma máquina (Troca Interna); (ii) Troca de tarefas na mesma posição em máquinas paralelas idênticas do mesmo estágio (Troca Externa); (iii) Realocação de tarefas dentro da mesma máquina; (iv) Realocação de tarefas em máquinas paralelas idênticas de um determinado estágio; (v) Troca de bloco de três tarefas na mesma máquina (Troca Interna de Bloco).

4.3. Busca Local

A técnica de busca local utilizada neste trabalho é o Método de Descida (*Best Improvement*). Este método parte de uma solução inicial e, a cada passo, todos os seus possíveis vizinhos são analisados. O método só move para um outro vizinho desde que ele represente uma melhoria no valor corrente da função de avaliação. Para realizar a busca local, os tipos de movimentos são embaralhados e um deles é escolhido aleatoriamente. Somente após a definição do tipo do movimento escolhido que o método de descida é aplicado na solução corrente.

4.4. Construção

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é um método iterativo, proposto por Feo e Resende (1995). Este método consiste em duas fases: construção e busca local. A etapa de busca local pesquisa por um ótimo local na vizinhança da solução inicial construída, elemento a elemento.

Na fase de construção, apresentada no Algoritmo 1, é gerada uma solução parcialmente gulosa, por meio de uma função adaptativa gulosa, que estima o benefício de cada um dos elementos. A aleatoriedade da construção é controlada por um parâmetro real $\alpha \in [0, 1]$. Para $\alpha = 1$, tem-se uma solução totalmente aleatória; para $\alpha = 0$, tem-se uma solução totalmente gulosa. Após a construção da solução, é aplicado um método de busca local. A construção de uma solução no método GRASP consiste em inserir elementos, obedecendo a um valor calculado pela função adaptativa gulosa $g(\cdot)$ e a uma regra de seleção que contém um fator aleatório.

Algoritmo 1: Construção GRASP

```
   Entrada: g(.), α
   Saída: s
1  início
2      s ← ∅;
3      Inicia o conjunto LC de itens candidatos;
4      Ordene o Conjunto LC de acordo com g(.);
5      enquanto LC ≠ ∅ faça
6          LRC = {conj. dos k melhores itens de LC};
7          Selecione, aleatoriamente, um item t ∈ LRC;
8          s ← s ∪ {t};
9          Atualize o conjunto LC de itens candidatos;
10     fim
11     Retorne s;
12 fim
```

Para construir a solução inicial do problema tratado neste artigo, foi utilizado o algoritmo 1. A função $g(t)$ utilizada para ordenar cada candidato (tarefa) da lista C é a soma dos tempos de processamento de cada tarefa em cada estágio. Após esta ordenação, todos os candidatos que possuem valor de função $g(t)$ ¡= $g(tmin)$ + α($g(tmax)$ - $g(tmin)$) são inseridos na LRC e selecionados aleatoriamente para serem sequenciados em cada máquina. Assim, a lista C é atualizada até que não exista nenhum candidato para ser selecionado. Após selecionar a tarefa a ser sequenciada, é necessário escolher a máquina responsável por processá-la. Para a escolha da máquina, foi utilizado o critério guloso, de acordo com a função de avaliação da máquina, que considera o instante de disponibilidade da máquina, o tempo de processamento de cada tarefa em cada estágio e o tempo de preparação no estágio i da tarefa anterior à tarefa escolhida para ser processada na máquina. Portanto, a máquina selecionada para processar a tarefa escolhida é a que apresenta menor instante de liberação.

4.5. Algoritmo GRASP-ILS Problema Dinâmico

O Algoritmo 2 apresenta o pseudocódigo do algoritmo padrão proposto, baseado no procedimento metaheurístico GRASP - ILS (Lourenço et al., 2003). Neste Algoritmo, a etapa de geração da solução inicial (linha 2) foi feita com base no procedimento de construção do GRASP. O método de Busca Local utilizado (linha 3) foi o método de Descida (*Best Improvement*), descrito na seção 4.3.

Das linhas 5 até 10 são aplicados procedimentos de forma iterativa até que o critério de parada seja atendido. Na linha 6 é aplicado o procedimento de perturbação da solução corrente, que consiste em realizar 20 vezes os movimentos de troca interna, troca externa, realocação interna (nível de perturbação). Na linha 7 é aplicado novamente o procedimento de busca local. Após a aplicação da perturbação e busca local, se a nova solução for melhor que a solução corrente, então a solução corrente é atualizada. O critério de parada adotado foi o número de 100 iterações sem melhora. Para resolver o problema HFFS-SDST dinâmico com evento de quebra de máquinas foi utilizado o Algoritmo 3 que simula a quebra de máquinas durante o processamento das tarefas de forma inesperada (dinâmica estocástica). Este algoritmo se diferencia do Algoritmo 2 nas linhas 2, 3 e 4. Para executar este algoritmo é necessário ter executado o algoritmo do problema estático. Isto porque é necessário conhecer a solução s^* do problema estático, ou seja, a sequência de cada tarefa em cada máquina de cada um dos estágios e o instante que cada tarefa será liberada em cada estágio ($disponibilidade_tarefa$).

O procedimento da linha 2 estabelece qual máquina de determinado estágio que ficará indisponível. O procedimento da linha 3 determina o instante do evento, ou seja, o instante da quebra da máquina. O procedimento de recorte (linha 4) é o procedimento que avalia: (i) quais as tarefas que estavam sendo processadas na máquina que ficou indisponível, ou seja, que deverão ter seu processamento reiniciado em outra máquina (TeIP); (ii) quais as tarefas que ainda não foram

Algoritmo 2: GRASP-ILS PROBLEMA ESTÁTICO

Entrada: α, *criterioParada*

1. **início**
2. $s \leftarrow$ ConstrucaoGRASP(α);
3. $s^* \leftarrow$ BuscaLocal(s); {Descida}
4. $nivel \leftarrow 20$;
5. **repita**
6. $s \leftarrow$ Perturbacao(s^*, $nivel$);
7. $s'' \leftarrow$ BuscaLocal(s); {Descida}
8. **se** $C_{\max}(s'') \text{ ¡ } C_{\max}(s^*)$ **então**
9. $s^* \leftarrow s''$;
10. **até** *criterioParada* ser satisfeito;
11. **retorne** s^*;
12. **fim**

Algoritmo 3: GRASP-ILS PROBLEMA DINÂMICO

Entrada: α, *criterioParada*

1. **início**
2. GeraEventoQuebra($qte_estagios$,$maquinas_estagio$);
3. InstanteQuebra(mi);
4. GeraRecorte(mi,s^*,$disponibilidade_tarefa$);
5. $s \leftarrow$ ConstrucaoGRASP(α);
6. $s^* \leftarrow$ BuscaLocal(s); {Descida}
7. $nivel \leftarrow 20$;
8. **repita**
9. $s \leftarrow$ Perturbacao(s^*, $nivel$);
10. $s'' \leftarrow$ BuscaLocal(s); {Descida}
11. **se** $C_{\max}(s'') \text{ ¡ } C_{\max}(s^*)$ **então**
12. $s^* \leftarrow s''$;
13. **até** *criterioParada* ser satisfeito;
14. **retorne** s^*;
15. **fim**

Tabela 2: Família de Instâncias

Família	Qte_Inst	Qte_Tar	Qte_Est
20x2	80	20	2
20x4	80	20	4
20x8	80	20	8
50x2	80	50	2
50x4	80	50	4
50x8	80	50	8
80x2	80	80	2
80x4	80	80	4
80x8	80	80	8
120x2	80	120	2
120x4	80	120	4
120x8	80	120	8

processadas no instante que ocorreu a quebra, considerando todos os estágios (TeNE); e (iii) quais as tarefas que já haviam iniciado o seu processamento, em máquinas que não ficaram indisponíveis, e que deverão ter seu processamento finalizado (TeAE).

5. Resultados

O algoritmo GRASP - ILS para o problema dinâmico, com evento de quebra de máquinas, foi implementado na linguagem C++, utilizando o Code Blocks 12.1, e testado em um computador Intel (R) Core(TM) i7-3632QM CPU @ 2.2 GHz, Memória RAM de 8 GB, com um sistema operacional de Windows 8, 64 bits. Para estudar a quebra inesperada durante o processamento das tarefas, foi selecionada uma instância de cada família. A Tabela 2 demonstra as características de cada família. Os resultados obtidos com o problema dinâmico, abordado neste artigo, foram comparados com os resultados do problema estático, com o objetivo de avaliar o desvio em %. As

Tabela 3: Instâncias Selecionadas para avaliação do HFFS-SDST Dinâmico

Inst	Tar	Est	E1	E2	E3	E4	E5	E6	E7	E8	T_E1	T_UL	% T_E1	%T_UL	MAQ
62-20x2	20	2	2	2							4	4	20%	20%	4
113-20x4	20	4	2	2	2	2					3	3	15%	15%	8
31-20X8	20	8	2	2	2	2	2	2	2	2	2	4	10%	20%	16
22-50x2	50	3	2	2							4	7	8%	14%	4
30-50X4	50	4	2	2	2	2					21	25	42%	50%	8
26-50X8	50	8	2	2	2	2	2	2	2	2	27	26	54%	52%	16
36-80X2	80	2	2	2							38	33	48%	41%	4
28-80X4	80	4	2	2	2	2					35	36	44%	45%	8
34-80X8	80	8	3	4	3	1	3	4	4	3	9	11	11%	14%	25
21-120x2	120	2	2	2							17	12	14%	10%	4
32-120x4	120	4	2	4	1	2					10	16	8%	13%	9
55-120x8	120	8	4	4	3	1	3	4	4	3	9	13	8%	11%	26

Tabela 4: Instâncias Selecionadas para avaliação do HFFS-SDST Dinâmico: resultados para o caso estático

Inst	Tar	Est	$s^*_{\min}$	s^*_{med}	Bib.[1]	$\Delta_{\min}$	Δ_{med}	T(s)
62-20x2	20	2	663	721	586	13%	23%	0,15
113-20x4	20	4	802	802	1030	-22%	-22%	0,47
31-20x8	20	8	1041	1239	1657	-37%	-25%	1,65
22-50x2	50	3	1235	1720	1736	-29%	-1%	1,12
30-50x4	50	4	1168	1407	1357	-14%	4%	1,63
26-50x8	50	8	1288	1390	1723	-25%	-19%	4,74
36-80x2	80	2	1499	1822	1875	-20%	-3%	1,31
28-80x4	80	4	2074	2507	2446	-15%	2%	5,37
34-80x8	80	8	3976	4046	5114	-22%	-21%	63,18
21-120x2	120	2	4364	4364	4132	6%	6%	10,49
32-120x4	120	4	5868	5941	6795	-14%	-13%	47,39
55-120x8	120	8	6407	6530	8915	-28%	-27%	188,38

[1]Melhor resultado apresentado em Naderi et al. (2010)

instâncias selecionadas, bem como as suas características, estão descritas na Tabela 3. Nesta Tabela, a primeira coluna (“Inst”) descreve o nome da instância; a segunda (“Tar”) e a terceira (“Est”) colunas mostram a quantidade de tarefas e a quantidade de estágios, respectivamente, na instância. Da quarta (“E1”) à décima-primeira (“E8”) colunas são apresentadas as quantidades de máquinas em cada estágio para cada instância. A coluna doze (“T_E1”) mostra a quantidade de tarefas que saltam o primeiro estágio. A coluna treze (“T_UL”) mostra a quantidade de tarefas que saltam o último estágio. As colunas quatorze (“% T_E1”) e quinze (“% T_UL”) apresentam o percentual de tarefas que saltam o primeiro estágio e o percentual de tarefas que saltam o último estágio, respectivamente. A última coluna (“MAQ”) apresenta o total geral de máquinas para cada instância.

Os resultados obtidos para o caso estático para estas instâncias estão apresentados na Tabela 4. Nesta Tabela, as três primeiras colunas são semelhantes às três primeiras colunas da Tabela 3. A quarta e a quinta colunas mostram o menor valor ($s^*_{\min}$) e o valor médio (s^*_{med}) de $C_{\max}$ encontrado para cada instância, considerando o sequenciamento sem eventos de quebra de máquinas. A sexta coluna possui o menor valor de $C_{\max}$ encontrado por Naderi et al. (2010) e a sétima e oitava colunas apresentam o desvio do algoritmo proposto neste trabalho em relação ao valor mínimo encontrado por Naderi et al. (2010), considerando o menor valor e o valor médio do $C_{\max}$. A última coluna apresenta o tempo médio em segundos que o algoritmo do problema estático gasta para executar a instância. O algoritmo proposto para o problema dinâmico foi executado 30 vezes para cada problema teste, e toda vez que os algoritmos ficavam 100 iterações sem melhorar a solução corrente, a execução era interrompida. Para testar o algoritmo foi simulada a quebra de uma máquina no primeiro estágio. Como a máquina pode ficar indisponível a qualquer instante durante a execução do processamento, foram feitas simulações considerando a quebra da máquina em 20%, 50% e 80% do valor de $C_{\max}$ do problema estático (sem quebra de máquinas). Estes percentuais foram escolhidos para avaliar o impacto da quebra diante de um determinado instante do horizonte de processamento.

Tabela 5: Tempo Faltante

Inst	SEM QUEBRA			TEMPO FALTANTE NO INSTANTE DA QUEBRA					
				$s^*_{\min}$			s^*_{med}		
	$s^*_{\min}$	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0,153	530	332	133	577	361	144
113-20x4	802	802	0,473	642	401	160	642	401	160
31-20x8	1041	1239	1,648	833	521	208	991	619	248
22-50x2	1235	1720	1,121	988	618	247	1376	860	344
30-50x4	1168	1407	1,634	934	584	234	1125	703	281
26-50x8	1288	1390	4,739	1030	644	258	1112	695	278
36-80x2	1499	1822	1,309	1199	750	300	1457	911	364
28-80x4	2074	2507	5,367	1659	1037	415	2005	1253	501
34-80x8	3976	4046	63,180	3181	1988	795	3237	2023	809
21-120x2	4364	4364	10,492	3491	2182	873	3491	2182	873
32-120x4	5868	5941	47,388	4694	2934	1174	4753	2970	1188
55-120x8	6407	6530	188,377	5126	3204	1281	5224	3265	1306

Tabela 6: Tempo Gasto

Inst	SEM QUEBRA			TEMPO GASTO PARA FINALIZAR O SEQUENCIAMENTO					
				$s^*_{\min}$			s^*_{med}		
	$s^*_{\min}$	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0,153	329	260	155	361	260	152
113-20x4	802	802	0,473	449	227	186	479	268	186
31-20x8	1041	1239	1,648	901	544	328	970	560	328
22-50x2	1235	1720	1,121	1274	683	252	1343	708	252
30-50x4	1168	1407	1,634	1398	807	420	1455	811	420
26-50x8	1288	1390	4,739	4685	2381	478	4788	2496	478
36-80x2	1499	1822	1,309	2279	1352	468	3200	1800	589
28-80x4	2074	2507	5,367	2350	1361	671	2978	1789	805
34-80x8	3976	4046	63,180	3239	2234	812	3410	2182	831
21-120x2	4364	4364	10,492	3560	2567	893	3541	2561	972
32-120x4	5868	5941	47,388	5100	3270	1213	5321	3300	1250
55-120x8	6407	6530	188,377	5261	3450	1322	5550	3670	1345

Este Cenário foi analisado considerando, primeiramente, as seguintes situações: o tempo faltante para finalizar o processamento no instante que ocorreu a quebra (Tabela 5), o tempo gasto para finalizar o processamento (Tabela 6) e o desvio entre o tempo gasto e o tempo faltante (Tabela 7). Como pode ser visto, as instâncias que apresentaram o maior desvio foram a instância 26-50×8, a instância 36-80×2, a instância 28-80×4 e a instância 30-50×4.

A Tabela 7 apresenta o tempo médio em segundos que o algoritmo gastou para sequenciar as tarefas faltantes no instante que ocorreu a quebra. Este tempo foi menor que o tempo que o algoritmo gastou para sequenciar o problema estático. Neste cenário de quebra não é possível afirmar que as instâncias mais afetadas foram as que têm menor número de máquinas, pois os resultados não mostram relação com esta variável. Como o tempo de processamento varia de instância para instância e o *setup* depende da sequência, no momento que ocorreu o evento uma instância que tem um alto tempo de processamento pode ter sido afetada.

Uma segunda linha de análise deste Cenário é mostrada considerando o tempo total de processamento antes de ocorrer a quebra da maquina (Tabela 8); o tempo gasto para finalizar o sequenciamento (Tabela 9); e o tempo total gasto, considerando o tempo gasto antes da ocorrência da quebra da maquina mais o tempo gasto para finalizar o sequenciamento (Tabela 10).

A Tabela 11 mostra o desvio do tempo total gasto em relação ao tempo total previsto no sequenciamento estático. Analisando o desvio do tempo total gasto em relação ao tempo total previsto, pode ser observado que as instâncias que apresentaram o maior desvio foram as instância 26-50×8, 36-80×2, 28-80×4 e 30-50×4.

6. Conclusões e trabalhos futuros

Esta artigo tratou do Problema *Flow shop* Híbrido e Flexível com Tempo de *Setup* Dependente da Sequência (HFFS-SDST), com eventos de quebra de máquinas, denominado dinâmico.

Tabela 7: Desvio Tempo Gasto X Tempo Faltante

Inst	SEM QUEBRA			DESVIO TEMPO GASTO X TEMPO FALTANTE						T(s)
				$s^*_{\min}$			s^*_{med}			
	$s^*_{\min}$	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%	
62-20x2	663	721	0,153	-38%	-22%	17%	-37%	-28%	5%	0,075
113-20x4	802	802	0,473	-30%	-43%	16%	-25%	-33%	16%	0,221
31-20X8	1041	1239	1,648	8%	5%	58%	-2%	-9%	32%	0,178
22-50x2	1235	1720	1,121	29%	11%	2%	-2%	-18%	-27%	0,723
30-50X4	1168	1407	1,634	50%	38%	80%	29%	15%	49%	0,292
26-50X8	1288	1390	4,739	355%	270%	86%	331%	259%	72%	2,241
36-80X2	1499	1822	1,309	90%	80%	56%	120%	98%	62%	2,29
28-80X4	2074	2507	5,367	42%	31%	62%	49%	43%	61%	3,21
34-80X8	3976	4046	63,180	2%	12%	2%	5%	8%	3%	32,1
21-120x2	4364	4364	10,492	2%	18%	2%	1%	17%	11%	5,52
32-120x4	5868	5941	47,388	9%	11%	3%	12%	11%	5%	28,21
55-120x8	6407	6530	188,377	3%	8%	3%	6%	12%	3%	90,12

Tabela 8: Tempo Total de Processamento antes da Quebra

Inst	SEM QUEBRA			TEMPO TOTAL DE PROCESSAMENTO ANTES DA QUEBRA					
				$s^*_{\min}$			s^*_{med}		
	$s^*_{\min}$	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0	133	332	530	144	361	577
113-20x4	802	802	0	160	401	642	160	401	642
31-20X8	1041	1239	2	208	521	833	248	619	991
22-50x2	1235	1720	1	247	618	988	344	860	1376
30-50x4	1168	1407	2	234	584	934	281	703	1125
26-50x8	1288	1390	5	258	644	1030	278	695	1112
36-80x2	1499	1822	1	300	750	1199	364	911	1457
28-80x4	2074	2507	5	415	1037	1659	501	1253	2005
34-80x8	3976	4046	63	795	1988	3181	809	2023	3237
21-120x2	4364	4364	10	873	2182	3491	873	2182	3491
32-120x4	5868	5941	47	1174	2934	4694	1188	2970	4753
55-120x8	6407	6530	188	1281	3204	5126	1306	3265	5224

A função objetivo proposta para resolver o problema foi a minimização do *Makespan*, denominado de $C_{\max}$. O problema tratado considera que existem tarefas que devem ser processadas em um conjunto de estágios dispostos em série, e que cada estágio possui uma ou mais máquinas paralelas idênticas. Além disso, nem todas as tarefas necessitam visitar todos os estágios e o tempo de *setup* é dependente da sequência. Foi realizado um estudo envolvendo cenários de quebras de máquinas. Para a solução do problema, foi desenvolvida metaheurística GRASP-ILS. Para a construção da solução inicial, foi utilizada a etapa de construção da metaheurística GRASP.

Para testar o Algoritmo, foi feita uma amostragem das instâncias de Naderi et al. (2010) utilizadas para resolver o problema FFFS-SDST em sua versão estática. Foram selecionadas 12 instâncias, uma de cada família (20x2, 20x4, 20x8, 50x2, 50x4, 50x8, 80x2, 80x4, 80x8, 120x2, 120x4 e 120x8), e, para cada cenário, foram utilizadas condições de testes diferentes. Os resultados obtidos mostram que o tempo médio em segundos que o algoritmo gastou para sequenciar as tarefas faltantes no instante em que ocorreu a quebra foi menor que o tempo gasto para sequenciar o problema estático.

Agradecimentos

Os autores agradecem às agências CAPES, FAPEMIG e CNPq, bem como ao CEFET-MG, pelo apoio ao desenvolvimento deste trabalho.

Referências

Alcaide, D.; Rodriguez-Gonzalez, A. e Sicilia, J. (2002). An approach to solve the minimum expected makespan flow-shop problem subject to breakdowns. *European Journal of Operational Research*, v. 140, p. 384–398.

Allahverdi, A. e Mittenthal, J. (1994). Two-machine ordered flow shop scheduling under random breakdowns. *Mathl. Comput. Modelling*, v. 20, p. 9–17.

Tabela 9: Tempo Gasto para finalizar o Sequenciamento

Inst	SEM QUEBRA			TEMPO GASTO PARA FINALIZAR O SEQUENCIAMENTO					
				s^*_{min}			s^*_{med}		
	s^*_{min}	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0	329	260	155	361	260	152
113-20x4	802	802	0	449	227	186	479	268	186
31-20x8	1041	1239	2	901	544	328	970	560	328
22-50x2	1235	1720	1	1274	683	252	1343	708	252
30-50x4	1168	1407	2	1398	807	420	1455	811	420
26-50x8	1288	1390	5	4685	2381	478	4788	2496	478
36-80x2	1499	1822	1	2279	1352	468	3200	1800	589
28-80x4	2074	2507	5	2350	1361	671	2978	1789	805
34-80x8	3976	4046	63	3239	2234	812	3410	2182	831
21-120x2	4364	4364	10	3560	2567	893	3541	2561	972
32-120x4	5868	5941	47	5100	3270	1213	5321	3300	1250
55-120x8	6407	6530	188	5261	3450	1322	5550	3670	1345

Tabela 10: TEMPO TOTAL GASTO

Inst	SEM QUEBRA			TEMPO TOTAL GASTO					
				s^*_{min}			s^*_{med}		
	s^*_{min}	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0	462	592	685	505	621	729
113-20x4	802	802	0	609	628	828	639	669	828
31-20x8	1041	1239	2	1109	1065	1161	1218	1180	1319
22-50x2	1235	1720	1	1521	1301	1240	1687	1568	1628
30-50x4	1168	1407	2	1632	1391	1354	1737	1514	1545
26-50x8	1288	1390	5	4943	3025	1508	5066	3191	1590
36-80x2	1499	1822	1	2579	2102	1667	3564	2711	2046
28-80x4	2074	2507	5	2765	2398	2330	3479	3042	2810
34-80x8	3976	4046	63	4034	4222	3993	4219	4205	4068
21-120x2	4364	4364	10	4433	4749	4384	4414	4743	4463
32-120x4	5868	5941	47	6274	6204	5907	6509	6270	6003
55-120x8	6407	6530	188	6542	6654	6448	6856	6935	6569

Tabela 11: Desvio Tempo Total Previsto X Tempo Total Gasto

Inst	SEM QUEBRA			DESVIO TEMPO TOTAL PREVISTO X TEMPO TOTAL GASTO					
				s^*_{min}			s^*_{med}		
	s^*_{min}	s^*_{med}	T(s)	20%	50%	80%	20%	50%	80%
62-20x2	663	721	0	-30%	-11%	3%	-30%	-14%	1%
113-20x4	802	802	0	-24%	-22%	3%	-20%	-17%	3%
31-20x8	1041	1239	2	7%	2%	12%	-2%	-5%	6%
22-50x2	1235	1720	1	23%	5%	0%	-2%	-9%	-5%
30-50x4	1168	1407	2	40%	19%	16%	23%	8%	10%
26-50x8	1288	1390	5	284%	135%	17%	264%	130%	14%
36-80x2	1499	1822	1	72%	40%	11%	96%	49%	12%
28-80x4	2074	2507	5	33%	16%	12%	39%	21%	12%
34-80x8	3976	4046	63	1%	6%	0%	4%	4%	1%
21-120x2	4364	4364	10	2%	9%	0%	1%	9%	2%
32-120x4	5868	5941	47	7%	6%	1%	10%	6%	1%
55-120x8	6407	6530	188	2%	4%	1%	5%	6%	1%

Feo, T.A. e Resende, M.G.C. (1995). Greedy randomized adaptive search. *Journal of Global Optimization*, v. 3, p. 109–133.

Gholami, M.; M.Zandieh, e A.Alem-Tabriz,. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *Int J Adv Manuf Technol*, v. 42, p. 189–201.

Gourgand, Michel; Grangeon, Nathalie e Norre, Sylvie. (2013). A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, v. 151, p. 415–433.

Gupta, J. N. D e Tunc, E. A. (1991). chedules for a two-stage hybrid flow shop scheduling with parallel machines at the second stage. *Int J Production Research*, v. 29, p. 1489–1502.

Kurz, M. E. e Askin, R. G. (2003). Comparing scheduling rules for flexible flowlines. *Journal of Production Economics*, v. 85, n. 3, p. 371–88.

Kurz, M. E. e Askin, R. G. (2004). Scheduling flexible flowlines with sequence-dependent setup times. *European Journal of Operational Research*, v. 59, n. 1, p. 66–82.

Lin, Shyh-Chang; Goodman, Erik D. e Punch III, William F. (1997). A genetic algorithm approach to dynamic job shop scheduling problem. Bäck, T., editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, p. 481–488, East Lansing, MI, USA.

Liu, S.Q.; Ong, H.L. e Ng, K.M. (2005). Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem. *Advances in Engineering Software*, v. 36, p. 199–205.

Lourenço, H.R.; Martin, O.C. e Stützle, T. (2003). Iterated local search. Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, p. 321–353. Kluwer Academic Publishers, Boston.

Mirabi, M.; Ghomi, S. M. T. Fatemi e Jolai, F. (2013). A two-stage hybrid flow shop scheduling problem in machine breakdown condition. *J Intell Manuf*, v. 24, p. 193–199.

Naderi, B.; Ruiz, R. e Zandieh, M. (2010). Algorithms for a realistic variant of flow shop scheduling. *Computers & Operations Research*, v. 37, p. 236–246.

Novas, Juan. M. e Henning, Gabriela. P. (2010). Reactive scheduling framework based on domain knowledge and constraint programming. *Computers and Chemical Engineering*, v. 34, n. 12, p. 2129–2148.

Ouelhadj, D. e Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, v. 12, n. 4, p. 417–431.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edição.

Safari, Ehram e Sadjadi, Seyed Jafar. (2011). A hybrid method for flow shops scheduling with condition-based maintenance constraint and machines breakdown. *Expert Systems with Applications*, v. 38, p. 2020–2029.

Schuster, Christoph J. e Framinan, J. M. (2003). Approximative procedures for no-wait job shop scheduling. *Operations Research Letter*, v. 31, n. 4, p. 308–318.

Yaurima, Victor; Burtseva, Larisa e Tchernykh, Andrei. (2009). Hybrid flow shop with unrelated machines, sequence-dependent setup time availability constraints and limited buffers. *Computers & Industrial Engineering*, v. 56, p. 1452–1463.

Ziaeifar, Amin; Tavakkoli-Moghaddam, Reza e Pichk, Khosro. (2012). Solving a new mathematical model for a hybrid flow shop scheduling problem with a processor assignment by a genetic algorithm. *Int J Adv Manuf Technol*, v. 61, p. 339–349.