COLISÃO DE PARTÍCULAS SIMILAR PARA O PROBLEMA DE ESCALONAMENTO DA PRODUÇÃO EM OFICINA DE MÁQUINAS

Tatiana Balbi Fraga (UFPE)
tatiana.balbi@ufpe.br
Antonio Jose da Silva Neto (IPRJ-UERJ)
ajsneto@iprj.uerj.br

Nesse trabalho é proposta uma nova heurística para solução do Problema de Escalonamento da Produção em Oficina de Máquinas (PEPOM), chamada Colisão de Partículas Similar. Essa heurística é uma adaptação do Algoritmo Colisão de Partículas, ooriginalmente desenvolvido para solução de problemas de otimização contínuos. Para teste da aplicação da heurística Colisão de Partículas Similar ao PEPOM é apresentado um novo algoritmo, chamado Multi Colisão de Partículas Similar com Exploração pelo Busca Tabu, onde um algoritmo Busca Tabu é utilizado como operador de exploração local e um operador de mutação é utilizado para perturbação da solução. Como resultado este trabalho demonstra que esse novo algoritmo é capaz de atenuar a sensibilidade dos ajustes de parâmetros e evitar os processos cíclicos, típicos do Busca Tabu.

Palavras-chaves: Problema de Escalonamento da Produção, Colisão de Partículas, Busca Tabu, otimização combinatória

1. Introdução

O problema de escalonamento da produção em oficina de máquinas (PEPOM) tem recebido nas últimas décadas uma forte atenção por parte de pesquisadores do mundo inteiro. Uma das razões deste intenso interesse é devida à sua ampla e relevante aplicabilidade já que esse problema aparece no dia a dia de indústrias dos mais diferentes segmentos e a forma como ele é tratado geralmente exerce um forte impacto nos custos de produção. A outra razão está em sua alta complexidade. O PEPOM é classificado como NP-difícil (LENSTRA *et al.*, 1977) e considerado como um dos mais difíceis problemas de otimização combinatória até então estudados. Resumidamente, o PEPOM pode ser apresentado da seguinte forma: são considerados um conjunto de m máquinas $\{M_i\}_{i=1}^{m}$ e um conjunto de n tarefas $\{J_j\}_{j=1}^{n}$ as quais devem ser processadas uma, e uma única, vez em cada elemento do conjunto de máquinas em uma ordem conhecida, a qual poderá variar de uma tarefa para outra. Dessa forma cada tarefa pode ser entendida como uma sequência de operações, $J_j = \left(O_{k_i j}\right)_{i=1}^{m}$, onde $O_{k_i j}$ é a operação da terefa J_j que deve ser processada pela máquina M_{k_i} durante um tempo de processamento conhecido e determinístico $t_{k_i j},\ t_{k_i j} \geq 0,\ k_i = 1, \ldots, m$. Adicionalmente: cada máquina pode processar uma única tarefa por vez; cada tarefa não pode ser processada por duas ou mais máquinas em um mesmo momento; e não é permitido preempção. O objetivo deste problema consiste em determinar a ordem em que as tarefas devem ser processadas em cada máquina de forma a minimizar o tempo total necessário ao processamento do grupo de tarefas, no inglês referido como *makespan*, sem no entanto desrespeitar às restrições impostas pelo problema.

Desde o início da década de 50, na literatura uma série de algoritmos exatos e de aproximação tem sido proposta para solução do PEPOM (BŁAŻEWICZ *et al.*, 1996; JAIN e MEERAN, 1999). Dentre os algoritmos exatos, aqueles que obtiveram os melhores resultados foram os algoritmos construídos com base nos métodos de Ramificação e Poda (BRUCKER *et al.*, 1994; CARLIER e PINSON, 1989). Contudo, apesar dos avanços tecnológicos e dos avanços obtidos na evolução desses algoritmos, ainda hoje, sua aplicação para problemas maiores é computacionalmente inviável. Dentre os algoritmos de aproximação, destacam-se os

algoritmos construídos com base nas heurísticas de Busca Local, tais como Busca Tabu (DELL'AMICO e TRUBIAN, 1993; TAILLARD, 1994; NOWICKI e SMUTNICKI, 1996; ZHANG *et al.*, 2007), Recozimento Simulado (VAN LAARHOVEN *et al.*, 1992; STEINHÖFEL *et al.*, 2003; AZIZI e ZOLFAGHARI, 2004), Algoritmos Genéticos (CHENG *et al.*, 1996 e 1999; PARK *et al.*, 2003; QING-DAO-ER-JI e WANG, 2012) e, mais recentemente, Otimização por Multidão de Partículas (LIAN *et al.*, 2006; SHA e HSU, 2006; LIN *et al.*, 2010). O problema destes algoritmos é que, apesar deles serem capazes de produzir boas soluções rapidamente, não se pode garantir que soluções ótimas sejam encontradas para todos os problemas testes apresentados na literatura. Adicionalmente, estes algoritmos são muito sensíveis tanto às soluções iniciais quanto ao ajuste de seus parâmetros.

Nesse trabalho é proposta uma nova heurística para solução do PEPOM chamada Colisão de Partículas Similar, inspirada no Algoritmo Colisão de Partículas criado por Sacco e Oliveira (2005) e até então aplicado a solução de problemas de otimização de natureza contínua (SACCO *et al.*, 2006; SACCO *et al.*, 2008; KNUPP *et al.*, 2009). Para teste da heurística aqui proposta é apresentado um novo algoritmo para solução do PEPOM, chamado Multi Colisão de Partículas Similar com Exploração pelo Busca Tabu (MCPS-BT), onde o método Busca Tabu de Nowicki e Smutnicki (1996), sem retropropagação, é aplicado como operador de exploração local, e o operador de mutação definido por Lian *et al.* (2006) como M7 com reposicionamento de apenas uma operação, M7 (1opt), é utilizado para pertubação da solução. A performance desse algoritmo é testada através dos conhecidos problemas testes das famílias FT (FISHER e THOMPSON, 1963) e AZT (ADAMS *et al.*, 1988).

As próximas seções são organizadas da seguinte forma: na próxima seção o algoritmo Colisão de Partículas, conforme figurado por Sacco *et al.* (2006) para solução de problemas contínuos é apresentado, assim como as adaptações propostas para solução do PEPOM; a seção 3 introduz o algoritmo MCPS-BT e apresenta os operadores M7 (1opt) e Busca Tabu que são utilizados, respectivamente, para pertubação e exploração local da solução nesse mesmo algoritmo; na seção 4 são apresentados alguns dos resultados computacionais assim como uma análise desses resultados; e, finalmente, na seção 5 são apresentadas as considerações finais e algumas propostas para trabalhos futuros.

2. Colisão de Partículas Similar

Ao observar a física do processo de colisão entre partículas nucleares dentro de um reator nuclear é possível verificar que dentre as partículas que colidem, aquelas que atingem os núcleos, ou seja, regiões de alta aptidão, são absorvidas e exploram as redondezas enquando que aquelas que atingem regiões de baixa aptidão podem ser absorvidas ou espalhadas para outras regiões de acordo com alguma função de probabilidade. Analisando essas interações, Sacco e Oliveira (2005) observaram que a sucessão dos eventos de absorção e espalhamento permitia que a movimentação das partículas promovesse uma exploração do espaço de busca completo ao mesmo tempo em que uma exploração mais aprofundada dos espaços mais promissores. Assim, com base nesta observação, os autores propuseram um novo algoritmo de busca o qual nomearam Algoritmo Colisão de Partículas, ou PCA (do inglês *Particle Collision Algorithm*). Nesse algoritmo inicialmente uma solução é escolhida e reservada como solução atual. Então, essa solução atual é pertubada através de uma função Pertubação(), gerando uma nova solução. Caso a nova solução seja "melhor" do que a solução atual (de acordo com algum critério previamente definido), a ultima solução é substituida pela primeira e sobre esta é aplicado um procedimento de busca local definido por uma função Exploração(). Caso a nova solução seja "pior" do que a solução atual, então é aplicada uma função Espalhamento(), onde uma função de probabilidade é utilizada para definir se a solução atual será explorada ou substituida por uma solução randômica. Observe que esse algoritmo apresenta uma estrutura similar ao Recozimento Simulado exceto pelo fato de que o algoritmo pode ser levado para novos espaços de busca não diretamente relacionados à solução inicial. Adicionalmente não é necessário que seja definida uma temperatura inicial. A Fig. 1 apresenta um pseudo-código para o PCA, conforme proposto por Sacco *et al*. (2006) para solução de problemas contínuos de maximização. Observe que neste algoritmo a função de probabilidade é definida pelo critério de Metropolis (METROPOLIS *et al.*, 1953). As Figs. 2 e 3 apresentam, respectivamente, as funções Perturbação() e Pequena_Perturbação() aplicadas nesse mesmo algoritmo. Nessas figuras é possível observar que, quando trabalhando com problemas de variáveis contínuas, as perturbações são geradas através de variações randômicas no valor de cada variável dentro de limites previamente estabelecidos, sendo as pequenas perturbação similares às perturbações, diferindo apenas nos limites que são mais estreitos. No caso de problemas discretos como o PEPOM, é necessário definir como essas perturbações serão realizadas.

A Fig. 4 apresenta um esquema geral para a heurística Colisão de Partículas Similar onde as

funções, Pertubação() e Exploração() são respectivamente substituidas por operadores de

Pertubação e Exploração Local. Como operadores de Pertubação, que têm como finalidade

levar o algoritmo para novos espaços de busca, são sugeridos os mesmo operadores de

mutação utilizados nos Algoritmos Genéticos, como por exemplo os operadores M1 a M10

apresentados por Lian *et al.* (2006). Como alternativa, caso o algoritmo Colisão de Partículas

Similar seja aplicado simultaneamente à todos os indivíduos de uma população inicial, pode-

Figura 1 – Pseudo-código para o Algorítmo Colisão de Partículas

```
Gere uma solução inicial Solução_Atual
Melhor_Aptidão = : Aptidão (Solução_Atual)
De n = 1 até # iterações
        Pertubação ()
        Se Aptidão (Nova_Solução) > Aptidão (Solução_Atual)
                Se Melhor_Aptidão > Aptidão (Nova_Solução)
                        Melhor_Aptidão = : Aptidão (Nova_Solução)
                Fim Se
                Solução_Atual = : Nova_Solução
                Exploração ()
        Se não
                Espalhamento ()
        Fim Se
Fim De

Exploração ()
        De n = 1 até # iterações
                Pequena_Pertubação ()
                Se Aptidão (Nova_Solução) > Aptidão
(Solução_Atual)
                        Solução_Atual = : Nova_Solução
                Fim Se
        Fim De
retorna

Espalhamento ()
        Probabilidade_Espalhamento = 1 − Aptidão (Nova_Solução) / Melhor_Aptidão
        Se
                Probabilidade_Espalhamento > valor ranômico (0, 1)
                Solução_Atual = : solução randômica
        Se não
                Exploração ()
        Fim Se
retorna
```

se utilizar os mesmos operadores de cruzamento dos Algoritmos Genéticos, como por exemplo os operadores C1 a C4 também apresentados por Lian *et al.* (2006), sobre os pares

de soluções. No caso dos operadores de Exploração Local, que têm como finalidade explorar as redondezas de uma dada solução, sugere-se os diversos algoritmos construídos com base nas heurísticas de busca local, tais como, Recozimento Simulado, Algoritmos Genéticos,

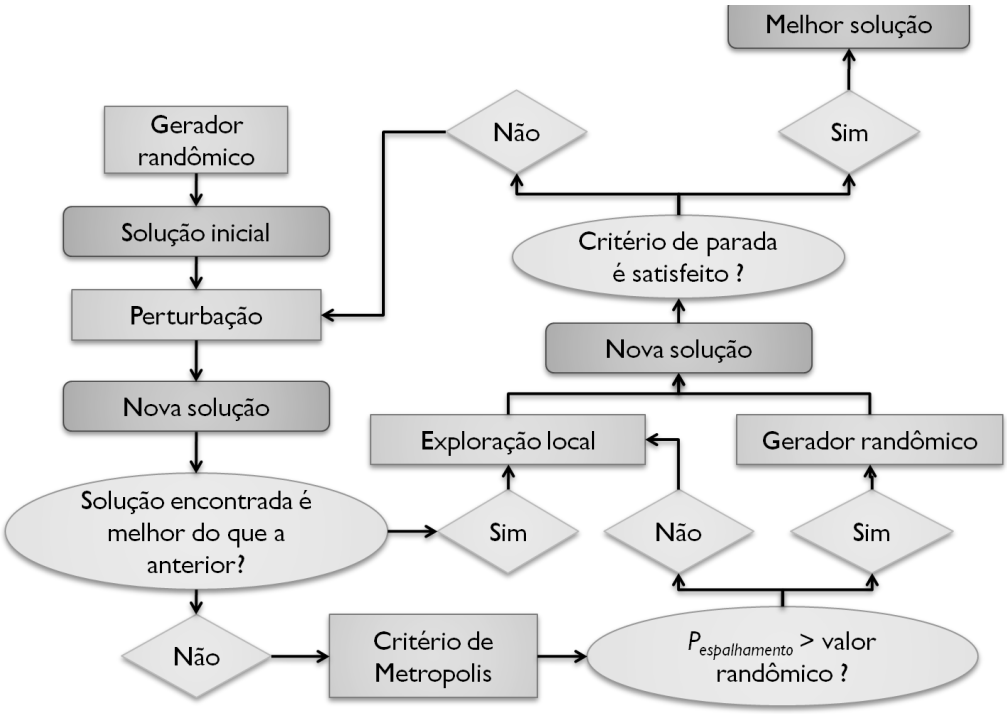
Figura 2 – Função Pertubação ()

```
Pertubação ()
        De i = 0 até (Dimensão – 1)
                Sup = : Limite Superior [i]
                Inf = : Limite Inferior [i]
                Rand = : Radômico (0,1)
                Nova_Solução = : Solução_Atual [i] + ((Sup - Solução_Atual [i])*Rand) –
                                        ((Solução_Atual [i] – Inf)*(1-Rand))
        Fim De
        Se Nova_Solução [i]   >  Sup
                Nova_Solução [i] = : SupLimit [i]
        Se não
                Se Nova_Solução [i] < Inf
                        Nova_Solução [i] = : InfLimit [i]
                Fim Se
        Fim Se
retorna
```

Colisão de Partículas e Busca Tabu.

```
Pequena_Pertubação ()
        De i = 0 até (Dimensão – 1)
                Sup = : Radômico (1.0,1.2)* Solução_Atual [i]
                Se (Sup > Limite Superior [i])
                        Sup = : Limite Superior [i]
                Fim Se
                Inf = : Radômico (0.8,1.0)* Solução_Atual [i]
                Se (Inf > Limite Inferior [i])
                        Inf = : Limite Inferior [i]
                Fim Se
                Rand = : Radômico (0,1)
                Nova_Solução = : Solução_Atual [i] + ((Sup - Solução_Atual [i])*Rand) –
                                ((Solução_Atual [i] – Inf)*(1-Rand))
        Fim De
retorna
```

Figura 4 – Esquema para a heurística Colisão de Partículas Similar.



3. Multi Colisão de Partículas com Exploração pelo Busca Tabu

Para teste da heurísitica Colisão de Partículas Similar (Fig. 4), nesta seção é apresentado um novo algoritmo para solução do PEPOM chamado Multi Colisão de Partículas Similar com Exploração pelo Busca Tabu (MCPS-BT). Nesse algoritmo a heurísitica Colisão de Partículas Similar é aplicada sobre um conjunto de soluções iniciais geradas randomicamente e nesse caso ela é definida como Multi Colisão de Partículas Similar. Como operador de perturbação é aplicado o operador de mutação M7 com reposicionamento de apenas uma operação, M7 (1opt) (LIAN *et al.*, 2006), e como operador de exploração local, é aplicada uma versão determinística do método Busca Tabu de Nowicki e Smutnicki (1996), sem retropropagação. Nas seções 3.2 e 3.3 estes dois operadores serão apresentados de forma mais detalhada. Uma vez que o PCA foi originalmente desenvolvido para problemas de maximização (Fig. 1), algumas alterações são necessárias para tratamento do PEPOM, essas alterações são apresentadas na seção 3.4. A proxima seção discute sobre a forma de representação das soluções adotada neste trabalho.

3.1. Representação das soluções

Durante o desenvolvimento de um algoritmo para solução do PEPOM, uma das questões chaves é a escolha da forma como as soluções serão representadas. Dentre as diversas formas de representação que aparecem na literatura, o Código de Processamento por Tarefas (CPT) possui a vantagem de representar apenas soluções viáveis e por esta razão foi adotado nesse trabalho. No CPT uma solução é representada por uma senquência contendo $n \times m$ algarismos. Nessa sequência o índice correspondentes à cada tarefa, $(i, i = 1 \dots n)$ é repetido m vezes, ou seja pelo número de operações da respectiva tarefa. Assim, uma dada operação é definida pelo índice de sua respectiva tarefa e pela posição em que este índice aparece. Por exemplo, na Fig. 5 é apresentada uma solução representada por um CPT para um PEPOM com 3 tarefas e 3 máquinas. Nessa sequência o primeiro algarismo representa a primeira operação da terceira tarefa, o segundo algarismo representa a primeira operação da segunda tarefa, o terceiro algarismo representa a segunda operação da terceira tarefa, e assim sucessivamente.

Figura 5 – CPT para um PEPOM com 3 tarefas e 3 máquinas.

CPT = (3,2,3,1,2,2,1,3,1)

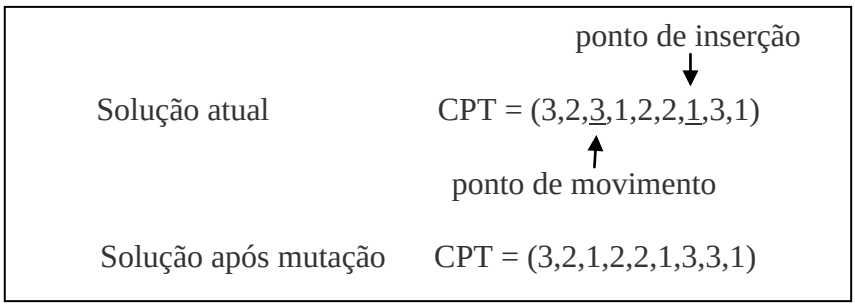
No CPT a posição de cada operação determina a ordem de prioridade em que ela deve ser processada de forma que a sequência em que as tarefas deverão ser processadas em cada máquina é determinada levando as operações para as suas respectivas máquinas na mesma ordem em que estas aparecem na solução dada pelo CPT. Uma das incoveniências desta forma de representação é que duas soluções aparentemente distintas, quando traçadas pelo CPT, podem retratar uma mesma solução (sequência de tarefas por máquina).

3.2. Operador de mutação M7 (1opt)

Os operadores de mutação forão introduzidos pelos diversos Algoritmos Genéticos no intuito de simular as alterações nos gens que ocorrem de tempos em tempos em determinados indivíduos, definidas no modelo de evolução natural de Darwin como mutações. Esses operadores têm como principal finalidade promover uma pertubação significativa nas

soluções, forçando os algoritmos a vasculhar novos espaços de busca, fungindo dessa forma dos mínimos locais. Assim sendo estes operadores se apresentam como bons substitutos para a função Pertubação () utilizada no PCA originalmente desenvolvido para solução de problemas contínuos. Dentre todos os operadores de mutação que são aplicados nos algoritmos de solução do PEPOM, o operador M7 (1opt) foi escolhido neste trabalho em função de sua simplicidade. Nesse operador, inicialmente duas posições são escolhidas randomicamente ao longo de uma solução representada pelo CPT, sendo a primeira posição definida como ponto de movimento e a segunda como ponto de inserção. Em seguida a operação posicionada no ponto de movimento é retirada deste ponto e reposicionada no ponto de inserção (Fig. 6).

Figura 6 – Operador de mutação M7 (1opt).



3.3. Operador Busca Tabu

Dentre os diversos procedimentos baseados nas heurísticas de busca local, o algoritmo Busca Tabu apresentado por Nowicki e Smutnicki (1996), com estrutura de vizinhança gerada pelo método N5, é ainda hoje considerada uma das técnicas que apresentam os melhores resultados, em termos de eficiência e eficácia, para solução do PEPOM, e por esta razão o operador de exploração local aplicado neste trabalho foi desenvolvido com base neste algoritmo. Esse operador pode ser resumidamente descrito da seguinte forma: inicialmente a solução inicial é definida como solução atual e sobre esta é aplicado o operador de geração de vizinhança N5, gerando pequenas pertubações que darão origem à um conjunto de novas soluções definido como vizinhança. Dentre os elementos da vizinhança é escolhida a melhor solução (solução com menor *makespan*) não pertencente a uma lista tabu (lista contendo referências das soluções que não devem ser escolhidas). Então essa solução é definida como solução atual, sobre essa são realizadas novas pertubações gerando uma nova vizinhança, é

feita uma nova seleção, e o processo continua até que um críterio de parada previamente estabelecido seja atingido. Nas próximas seções o operador de geração de vizinhança N5, o processo de seleção da melhor solução juntamente com a lista tabu, e o critério de parada aplicados nesse algoritmo serão descritos de forma mais detalhada.

Operador de geração de vizinhança N5

Dada uma solução (sequência de tarefas por máquina) qualquer, um caminho crítico pode ser definido como uma sequência de operações onde:

- a última operação na ordem desta sequência é a operação que é finalizada por último e que, portanto, tem tempo final de processamento igual ao *makespan*;
- a operação precedente a cada operação, exceto à primeira, na ordem desta sequência é a operação que a precede na ordem de máquina ou na ordem de tarefa, sendo entre as duas àquela que apresentar maior tempo final de processamento;
- o momento inicial de processamento da primeira operação na ordem desta sequência é igual a zero.

É importante observar que uma solução pode conter mais do que um caminho crítico, como no caso em que há duas ou mais operações que são finalizadas por último, ou quando as operações que precedem uma dada operação nas ordens de máquina e de tarefa são finalizadas em um mesmo momento. Em seu artigo, Nowicki e Smutnicki (1996) afirmam que a escolha do caminho crítico não gera forte influência no resultado final e os autores sugerem que o mesmo seja escolhido de forma randômica. Neste trabalho, como deseja-se que o operador Busca Tabu apresente-se como um método determinístico, o caminho crítico é selecionado de acordo com as seguintes prioridades:

- caso duas operações sejam finalizadas por último, a operação escolhida será aquela pertencente a máquina de menor índice;
- caso as operações que precedem uma dada operação nas ordens de máquina e de serviço sejam finalizadas em um mesmo momento, a operação escolhida será aquela precede na ordem de serviço.

O caminho crítico também pode ser subdivido em blocos de operações onde cada bloco é composto pela subsequência de operações sucessivas que devem ser processadas por uma mesma máquina. Pelo método de geração de vizinhança N5, uma vizinhança é formada pela

troca entre as duas primeiras e as duas últimas operações de cada bloco de operações no caminho crítico, sendo que no caso do primeiro bloco, apenas as duas últimas operações são trocadas e no caso do último bloco apenas as duas primeiras operações são trocadas. Caso algum bloco seja formado por apenas uma operação, nenhuma troca sera realizada e, caso o mesmo seja formado por duas operações, será realizada apenas uma troca.

Seleção da melhor solução não tabu

No operador Busca Tabu aplicado nesse trabalho, a lista tabu é constituída de um vetor contendo os movimentos inversos aos utilizados na geração das t últimas soluções atuais (onde t é um parâmetro cujo valor pode ser ajustado). Assim, o melhor vizinho (solução escolhida como nova solução atual) será a solução de menor *makespan* cujo movimento gerador não consta na lista tabu. Adicionalmente é considerado aqui um critério de aspiração que permite a substituição da solução atual por uma solução cujo movimento gerador pertence à lista tabu caso o *makespan* dessa solução seja menor do que o *makespan* da melhor solução até então encontrada.

Critério de parada

Nesse algoritmo o processo iterativo é interrompido quando durante $Nite$ iterações a melhor solução encontrada não é atualizada, ou seja, nenhuma nova solução atual apresenta menor *makespan* do que o *makespan* da melhor solução encontrada. Sendo $Nite$ um parâmetro cujo valor pode ser ajustado.

3.3. Função de probabilidade

Conforme anteriormente citado, o Algoritmo Colisão de Partículas foi originalmente desenvolvido para solução de problemas de maximização. No caso do PEPOM, como se busca minimizar o valor da função objetivo definida pelo *makespan*, o critério de Metropolis deve ser alterado dando à função de probabilidade de espalhamento a seguinte forma:

$$\text{Probabilidade\_Espalhamento} = 1 - \frac{\text{Melhor\_Aptidão}}{\text{Aptidão (Nova\_Solução)}}$$

Sendo o valor da variável Melhor_Aptidão definido pelo *makespan* da melhor solução até então encontrada e o valor da variável Aptidão (Nova_Solução) definido pelo *makespan* da solução analisada. Dessa forma a probabilidade de espalhamento irá aumentar a medida em que o *makespan* da solução analisada se afasta do *makespan* da melhor solução e será nula caso esses dois valores sejam iguais.

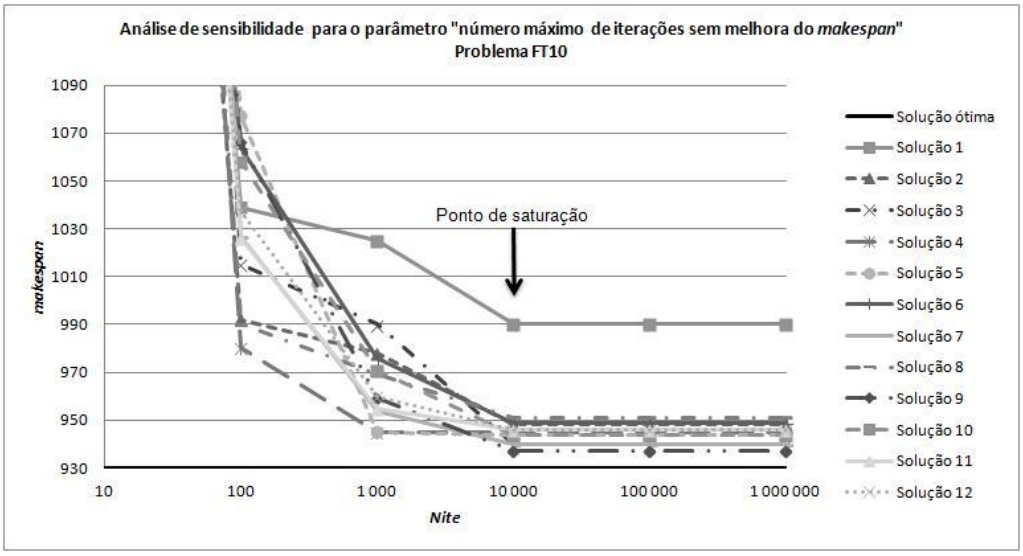
4. Resultados e discussões

Para teste do algoritmo MCPS-BT, proposto neste trabalho foram utilizados os problemas FT6 (com 6 máquinas e 6 tarefas), FT10 (com 10 máquinas e 10 tarefas) e FT20 (com 5 máquinas e 20 tarefas) propostos por Fisher e Thompson (1963) e os problemas ABZ5 e ABZ6 (com 10 máquinas e 10 tarefas), e ABZ7, ABZ8 e ABZ9 (com 15 máquinas e 20 tarefas) propostos por Adams, Balas e Zawack (1988). Para cada problema foram geradas randomicamente 12 soluções iniciais e as mesmas foram utilizadas como soluções iniciais para todos os testes aplicados. Os problemas testes das famílias FT e ABZ assim como as tabelas contendo os resultados de todos os testes aqui discutidos podem ser consultados na tese de doutorado de Fraga (2010).

Inialmente foi realizada uma análise de sensibilidade do algoritmo Busca Tabu, utilizado como operador de pertubação local, com relação aos parâmetros: número máximo de iterações sem melhora do *makespan* $(Nite)$; e tamanho da lista tabu (t). Desta análise observou-se que este algoritmo, quando aplicado sozinho, não é robusto, uma vez que sua eficácia depende do valor estipulado para o parâmetro t, cujo valor adequado irá depender de cada problema específico. Diferentes valores aferidos a t resultam em diferentes resultados. Também foi verificado que, caso este parâmetro não seja adequadamente ajustado, o algoritmo entra em ciclos e, a partir de um determinado valor definido aqui como ponto de saturação (PS), nenhum aumento do parâmetro $Nite$ é capaz de melhorar o resultado. Também verificou-se que a ordem do valor deste ponto de saturação pode estar diretamente associada ao tamanho do problema.

Na Fig. 7 é apresentada uma análise da variação do *makespan* ao aumento do valor do parâmetro $Nite$ para o problema FT10 onde verifica-se que para este problema teste $PS \sim 10.000$. O mesmo ponto de saturação é verificado para os problemas ABZ5 e ABZ6, os quais apresentam o mesmo número de máquinas e tarefas que o FT10. Para o problema FT20 foi verificado que $PS \sim 1.000$ e para os problemas com 15 máquinas e 20 tarefas (ABZ7, ABZ8 e ABZ9) foi verificado que $PS \sim 100.000$. No caso do problema FT6, todas as soluções convergiram para ótimos globais com pouquíssimas iterações, de forma que esse problema foi utilizado apenas para validação dos algoritmos apresentados neste trabalho.

Figura 7 – Análise de sensibilidade do *makespan* ao valor do parâmetro "número máximo de iterações sem melhora do *makespan*" do algoritmo Busca Tabu para o problema FT10 (tamanho da lista tabu = 8).



As inconveniências do algoritmo Busca Tabu aqui destacadas, i.e. relacionadas ao ponto de saturação e à sensibilidade do algoritmo ao valor do parâmetro "tamanho da lista tabu", foram anteriormente comentadas por Nowicki e Smutinicki (1996) na apresentação de seu algoritmo Busca Tabu com estrutura de vizinhança gerada pela técnica N5. Para superar essa dificuldade os autores propuseram a utilização de uma técnica de retropropagação através da qual os ciclos são identificados, fazendo com que o algoritmo saia do *looping* infinito e retorne à uma das melhores soluções até então visitadas. A desvantagem desta técnica é o significativo aumento da complexidade do algoritmo e do uso de memória. Adicionalmente essa técnica

não garante que soluções ótimas sejam encontradas e não diminui de forma significativa a sensibilidade do algoritmo ao ajuste do parâmetro "tamanho da lista tabu".

Para análise de convergência do algoritmo MCPS-BT, foi gerada uma análise de sensibilidade do *makespan* à variação do parâmetro "número de ciclos". No caso dos problemas FT10, FT20, ABZ5 e ABZ6, foi verificado que o algorítimo convergiu para a solução ótima em todas as rodadas sendo que esta conversão foi um pouco mais lenta para os problemas ABZ5 e ABZ6. No caso dos problemas ABZ7, ABZ8 e ABZ9, o teste de convergência extrapolou o tempo máximo de processamento estabelecido (1 semana) sem nem mesmo terminar os resultados da primeira rodada. Esses resultados refletem a complexidade que existe na solução do PEPOM discutida na introdução deste trabalho.

Finalmente, para análise da robustez do algoritmo MCPS-BT, foi realizada uma análise de sensibilidade do parâmetro t do operador de exploração local Busca Tabu. Uma vez que o algoritmo tem natureza randômica, cada teste foi realizado 5 vezes. Como resultado verificou-se que, independente do valor ajustado para esse parâmetro, quase todas as soluções geradas foram melhores do que a melhor solução encontrada pela simples aplicação algoritmo Busca Tabu, demonstrando a superioridade do algoritmo desenvolvido. Apesar de ter sido reparada uma pequena flutuação dos valores encontrados, foi constatado que essa flutuação se deve mais à natureza randômica do algoritmo do que a variação do valor do parâmetro analisado.

Através dos testes anteriormente descritos foi possível constatar que o componete randômico da heurística Colisão de Partículas Similar permite que o algoritmo seja conduzido a novos espaços de busca, superando as dificuldades encontradas pelo algoritmo Busca Tabu e produzindo dessa forma resultados melhores e mais robustos com um custo computacional equivalente. A Tab. 1 apresenta uma comparação entre os resultados encontrados pelo algoritmo de Nowicki e Smutinick (1996) com retropropagação (TSAB), e os melhores resultados encontrados para o MCPS-BT, para cada um dos problemas testes considerados. Conforme pode ser observado nesta tabela, o MCPS-BT foi capaz de encontrar soluções ótimas para os problemas FT20, ABZ5 e ABZ6, superando o algoritmo TSAB. Os autores não realizaram testes para os problemas ABZ7, ABZ8 e ABZ9 de forma que nesse caso não foi possível uma comparação.

Tabela 1 – Comparação entre os resultados apresentados por Nowicki e Smutnicki para seu algoritmo TSAB e os melhores resultados obtidos pelo algoritmo MCPS-BT, para os problemas das famílias FT e ABZ

Problema	Solução ótima	TSAB	MCPSEBT
FT6	55	55	55
FT10	930	930	930
FT20	1165	1178	1165
ABZ5	1234	1238	1234
ABZ6	943	945	943
ABZ7	656	***	660
ABZ8	645	***	671
ABZ9	661	***	684

5. Conclusões e Trabalhos Futuros

Nesse trabalho foi apresentada uma nova heurística para solução do Problema de Escalonamento da Produção em Oficina de Máquinas chamada Colisão de Partículas Similar (CPS). Para teste dessa heurística foi proposto um novo algoritmo chamado Multi Colisão de Partículas Similar com Exploração pelo Busca Tabu onde o operador de mutação M7(1opt) e o algoritmo Busca Tabu de Nowicki e Smutnicki, sem retropropagação, foram aplicados como operadores de pertubação e exploração local, respectivamente. Como resultado foi demostrado que as técnicas de pertubação e expalhamento do CPS levam o algoritmo a novos espaços de busca, evitando que a solução encalhe em mínimos locais conforme ocorre com o algoritmo Busca Tabu considerado neste trabalho, quando aplicado sozinho. Por esta razão, a heurística CPS foi capaz de apresentar resultados melhores e mais robustos para todos os problemas testes analisados, por um custo computacional equivalente.

Uma vez que todos os testes foram realizados com um conjunto contendo 12 soluções iniciais fixas seria necessário testar a eficiência do algoritmo para diferentes soluções iniciais assim como realizar análises de sensibilidade para o tamanho da população inicial. Adicionalmente novos operadores de pertubação e exploração podem ser testados buscando melhorar a eficiência do algoritmo. Esses testes ficam como recomendações para trabalhos futuros.

Agradecimentos: Esse trabalho foi possível graças ao suporte financeiro oferecido pela CAPES, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

REFERÊNCIAS

ADAMS, J., BALAS, E., ZAWACK, D. **The shifting bottleneck procedure for job-shop scheduling.** Management Science, 34, p. 391-401. 1988.

AZIZI, N., ZOLFAGHARI, S. **Adaptive temperature control for simulated annealing: a comparative study.** Computers & Operations Research, 31, p. 2439-2451. 2004.

BŁAŻEWICZ, J., DOMSCHKE, W., PESCH, E. **The job shop scheduling problem: Conventional and new solution techniques.** European Journal of Operational Research, 93, p. 1-33. 1996.

BRUCKER, P., JURISCH, B., SIEVERS, B. **A branch and bound algorithm for the job-shop scheduling problem.** Discrete Applied Mathematics, 49, p. 107-127. 1994.

CARLIER, J., PINSON, E. An **algorithm for solving the job-shop problem.** Management Science, 35 (2), p. 164-176. 1989.

CHENG, R., GEN, M., TSUJIMURA, Y. **A tutorial survey of job-shop scheduling problems using genetic algorithms – I. Representation.** Computers & Industrial Engineering, 30 (4), p. 983-997. 1996.

CHENG, R., GEN, M., TSUJIMURA, Y. **A tutorial survey of job-shop scheduling problems using genetic algorithms: Part II. Hybrid Genetic Search Strategies.** Computers & Industrial Engineering, 37, p. 51-55. 1999.

DELL'AMICO, M., TRUBIAN, M. **Applying tabu search to the job-shop scheduling problem.** Annals of Operations Research, 41, p. 231-252. 1993.

FISHER, H., THOMPSON, G. L. **Probabilistic learning combinations of local job-shop scheduling rules.** Muth, J. F., Thompson, G. L. (eds.). Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey. 1963.

FRAGA, T. B. **Proposição e análise de modelos híbridos para o problema de escalonamento de produção em oficina de máquinas.** Tese de doutorado. Nova Friburgo, RJ : Universidade do Estado do Rio de Janeiro - Instituto Politécnico. 2010.

JAIN, A. S., MEERAN, S. **Deterministic job-shop scheduling: Past, present and future.** European Journal of Operational Research, 113, p. 390-434. 1999

KNUPP, D. C., SILVA NETO, A. J. da, SACCO, W. F. **Radiative properties estimation with the particle collision algorithm based on a sensitivity analysis.** High Temperatures – High Pressures, 38, p. 137-151. 2009.

LENSTRA, J. K., RINNOOY KAN, A. H. G., BRUCKER, P. **Complexity of machine scheduling problems.** Annals of Discrete Mathematics, 1, p. 343-362. 1977.

LIAN, Z., JIAO, B., GU, X. **A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan.** Applied Mathematics and Computation ,183, p. 1008-1017. 2006.

LIN, T.-L., HORNG, S.-L., KAO, T.-W., CHEN, Y.-H., RUN, R.-S., CHEN, R.-J., LAI, J.-L., KUO, I-H. **An efficient job-shop scheduling algorithm based on particle swarm optimization.** Expert Systems with Applications, 37, p. 2629-2636. 2010.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., TELLER, E. **Equations of state calculations by fast computing machines.** J. Chem. Phys., 21, p. 1087-1092. 1953.

NOWICKI, E., SMUTNICKI, C., **A fast taboo search algorithm for the job shop problem.** Management Science, 42 (6), p. 797-813. 1996.

PARK, B. J., CHOI, H. R., KIM, H. S. **A hybrid genetic algorithm for the job shop scheduling problems.** Computers & Industrial Engineering, 45, p. 597-613. 2003.

QING-DAO-ER-JI, R., WANG, Y. **A new hybrid genetic algorithm for job shop scheduling problem.** Computers & Operations Research, 39, p. 2291-2299. 2012.

SACCO, W. F., LAPA, C. M. F., PEREIRA, C. M. N. A. ALVES FILHO, H. **A metropolis algorithm applied to a nuclear power plant auxiliary feedwater system surveillance tests policy optimization.** Progress in Nuclear Energy, 50, p. 15-21. 2008.

SACCO, W. F., OLIVEIRA, C. R. E. de. **A new stochastic optimization algorithm based on a particle collision methaheuristic.** 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil. 2005.

SACCO, W. F., OLIVEIRA, C. R. E. de, PEREIRA, C. M. N. A. **Two stochastic optimization algorithms applied to nuclear reactor core design.** Progress in Nuclear Energy, 48, p. 525-539. 2006.

SHA, D. Y., HSU, C.-Y. **A hybrid particle swarm optimization for job shop scheduling problem.** Computers & Industrial Engineering, 51, p. 791-808. 2006.

STEINHÖFEL, K., ALBRECHT, A., WONG, C. K. **An experimental analysis of local minima to improve neighborhood search.** Computers & Operations Research, 30, p. 2157-2173. 2003.

TAILLARD, É. D. **Parallel taboo search techniques for the job shop scheduling problem.** ORSA Journal on Computing, 6 (2), p. 108-117. 1994.

VAN LAARHOVEN, P. J. M., AARTS, E. H. L., LENSTRA, J. K. **Job shop scheduling by simulated annealing.** Operations Research, 40 (1), p. 113-125. 1992.

ZHANG, C. Y., LI, P. G., GUAN, Z. L., RAO, Y. Q. **A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem.** Computers & Operations Research, 34, p. 3229-3242. 2007.