UM ALGORITMO VNS MULTIOBJETIVO PARA O PROBLEMA DE SEQUENCIAMENTO COM ALOCAÇÃO DE TRABALHADORES

Guido Pantuza Jr.
Instituto Federal de Minas Gerais – IFMG
Rua Minas Gerais, Governador Valadares – MG.
guido.junior@ifmg.edu.br

RESUMO

O presente trabalho trata do problema de sequenciamento e alocação de trabalhadores (SPWA). Para tal, propõem-se um algoritmo heurístico chamado de VNS-Multiobjetivo baseado no método heurístico VNS. No SPWA, objetiva-se minimizar o número de trabalhadores e o tempo total gasto para executar todas as tarefas (*makespan*). Como os objetivos são conflitantes entre si, o método proposto no VNS-Multiobjetivo gera um conjunto de soluções eficientes, cabendo ao gestor escolher qual a solução deve ser adotada. Portanto, este trabalho propõe um método para resolução de problemas multiobjetivos, demostrando que é possível utilizar um número reduzido de funcionários e terminar todas as tarefas em tempo hábil, utilizando assim, os recursos de uma empresa de forma otimizada.

PALAVARAS CHAVE. Otimização Multiobjetivo, VNS-Multiobjetivo, SPWA.

MH - Metaheurísticas
OC - Otimização Combinatória

ABSTRACT

This paper deals with the problem of scheduling and allocation of workers (SPWA). To this, is proposed a heuristic algorithm called the VNS-based Multiobjective heuristic VNS. In SPWA, the objective is to minimize the number of workers and the total time taken to perform all jobs (makespan). As the objectives are conflicting, the proposed method in Multiobjective VNS-generates a set of efficient solutions, being the manager to choose which solution should be adopted. Therefore, this paper proposes a new method for solving multiobjective problems, showing that you can use a small number of employees and end all jobs in a timely manner, thereby using the resources of a company optimally.

KEYWORDS. Multiobjective Optimization, VNS-Multiobjective, SPWA.

MH - Metaheuristics
OC - Combinatorial Optimization

1. Introdução

O mercado mundial está cada vez mais competitivo e exigente. Além disso, as crises mundiais e a recessão na Europa ameaçam a saúde financeira das empresas. Diante deste cenário, as empresas necessitam reduzir o seu custo produtivo. Uma forma de reduzir este custo é através do uso otimizado dos fatores de produção.

Entre os diversos fatores de produção, um dos que possuem maior impacto nos custos produtivos é a mão-de-obra. Segundo a FIESP (2011), a mão de obra, em 2009, foi responsável, em média, por 32,4% do custo total dos bens produzidos no Brasil. Trata-se do valor mais alto de todos os países estudados (34 maiores economias) e 11% acima da média mundial que foi de 21,4%. Se comparado com outros países em desenvolvimento temos: 14,7% em Taiwan, 17% na Argentina e Coréia do Sul e 27% no México.

Isto se deve ao fato dos recentes aumentos dos salários acima da inflação, adicionados aos encargos sociais e benefícios. Por isso, as empresas procuram a redução do número de funcionários sem afetar os prazos de entrega e qualidade do produto final.

Esse problema enfrentado pelas empresas é conhecido na literatura como o problema de sequenciamento com alocação de trabalhadores (*Scheduling Problem with Worker Allocation* – SPWA). O problema consiste em alocar as tarefas aos trabalhadores de uma empresa, minimizando o instante de término da última tarefa executada (*makespan*) e o número de funcionários utilizados.

Esse problema é composto por dois objetivos conflitantes: minimizar o número de trabalhadores e o tempo total gasto. Dessa forma, não existe uma solução única que otimize todas elas ao mesmo tempo. Assim, adotamos o método de otimização multiobjetivo.

Esse método consiste na busca de um conjunto de soluções eficientes, o que torna o sistema mais flexível, uma vez que ele apresenta diversas soluções diferentes. Tal fato permite que o gestor escolha a solução que for mais conveniente.

Para a resolução de problemas multiobjetivos, os métodos variam entre métodos exatos e heurísticos. Para a abordagem exata tem-se a certeza de uma solução ótima, mas, para problemas complexos, geralmente o tempo despendido é superior ao tempo disponível para a tomada de decisão. A utilização de métodos heurísticos é recomendada quando é procurada uma boa solução em tempo hábil. Entretanto, esse método não garante a otimalidade da solução para o problema.

Segundo, Tan *et al*. (2009), o SPWA é um problema NP-difícil, ou seja, para instâncias com dimensões maiores, não é possível encontrar a solução ótima global em tempo computacional hábil. Logo, o objetivo deste trabalho é propor um modelo heurístico multiobjetivo baseado no algoritmo *Variable Neighborhood Search* – VNS aplicado ao problema SPWA.

Para tanto, este trabalho está organizado como segue. Na seção 2 descreve-se o problema em estudo. Na seção 3 o referencial teórico. Na seção 4, apresenta-se a metaheurística VNS aplicada ao SPWA. A apresentação das instâncias testes e dos resultados são feitas na seção 5. A conclusão é apresentada na seção 6.

2. Problema de Sequenciamento com Alocação de Trabalhadores

O problema de sequenciamento com alocação de trabalhadores também é conhecido como S*cheduling Problem with Worker Allocation* (SPWA).

O problema é composto por um conjunto de trabalhadores, *Worker*, e um conjunto de tarefas, *Job*. Ele consiste em alocar as tarefas aos trabalhadores e propor uma sequencia de execução, respeitando as restrições de qualificação. Ou seja, um trabalhador só pode executar uma tarefa se ele for qualificado. Além disso, cada trabalhador, para executar uma mesma tarefa, necessita de tempos diferentes de acordo com suas habilidades.

O objetivo do SPWA é minimizar, simultaneamente, o *makespan* e minimizar o número total de funcionários utilizados.

Neste trabalho consideramos as seguintes restrições: Todas as tarefas devem ser executadas. Não consideramos o tempo de deslocamento dos funcionários ou o tempo de espera em fila. O horizonte de planejamento da alocação dos trabalhadores é fixo. Cada tarefa é executada por apenas um único funcionário. O excesso de trabalho para os trabalhadores não é considerado, ou seja, não consideramos que os trabalhadores ficarão sobrecarregados com excesso de trabalho. Todas as tarefas tem o mesmo nível de esforço. Todos os funcionários possuem o mesmo custo (salário). Cada colaborador possui uma habilidade diferente, ou seja, cada um executa a mesma tarefa com um tempo diferente.

Na literatura encontramos alguns trabalhos que abordam este problema. Iima e Sannomiya (2001), propuseram uma heurística para resolução do SPWA fundamentada no Algoritmo Genético chamada *Module Type Genetic Algorithm* (MTGA).

Iima e Sannomiya (2002), também utilizaram o MTGA, porém consideraram que cada trabalhador possui o mesmo nível de qualificação para executar as diferentes tarefas.

Osawa e Ida (2005), também utilizaram Algoritmo Genético, porém propuseram um novo método de seleção da população sobrevivente.

Todos os trabalhos citados anteriormente adotaram abordagens mono-objetivas. Ou seja, a cada iteração, as soluções são avaliadas atribuindo-se pesos para cada objetivo. Nesta abordagem, o peso atribuído a cada objetivo é constante, logo, algum objetivo pode ser priorizado em detrimento dos outros. Não é de conhecimento do autor deste trabalho uma abordagem multiobjetivo para o SPWA.

3. Fundamentação Teórica

O SPWA é composto por dois objetivos conflitantes (minimizar o número de funcionários e o *makespan*). Assim, não existe uma solução única que otimize todas elas ao mesmo tempo. Por exemplo, com um número reduzido de funcionários não é possível executar todas as tarefas em tempo hábil.

Problemas dessa natureza são chamados de problemas de otimização multiobjetivo. Eles envolvem minimização e/ou maximização simultânea de um conjunto de objetivos satisfazendo a um conjunto de restrições. Deve-se, assim, buscar um conjunto de soluções eficientes. Neste caso, a tomada de decisão será de responsabilidade do gestor. Ele poderá escolher a solução que melhor se adapta às necessidades de produção dentre as soluções eficientes.

Segundo Fonseca e Fleming (1995), a otimização multiobjetivo diverge da otimização mono-objetivo, devido ao fato de raramente admitir uma simples solução. Por isso, a solução é composta por uma família de soluções Pareto-ótimas que devem ser consideradas equivalentes, em vista da ausência de informação referente à importância de cada objetivo.

O conjunto de soluções eficientes também é conhecido como soluções Pareto-ótimas. Segundo Pareto (1896), o conceito de Pareto-ótimo constitui o objetivo da busca na otimização multiobjetivo. Por definição, um conjunto de soluções S é Pareto-ótimo se não existe outro conjunto de soluções viáveis S^* que possa melhorar algum objetivo, sem causar uma piora em pelo menos outro objetivo. Em outras palavras, uma solução s pertence ao conjunto de soluções Pareto-ótimo S se não existe solução s^* que domine s.

Considerando um problema de minimização, temos:

- s domina s^* se, e somente se, $s_l \leq s_l^* \ \forall l$ e $s_l < s_l^*$ para algum l;
- s e s^* são indiferentes ou possuem o mesmo grau de dominância se, e somente se, s não domina s^* e s^* não domina s.

Segundo Pantuza Jr. (2011), para otimizar os problemas de otimização multiobjetivo NP-difícil, os métodos baseados em heurísticas parecem ser a melhor alternativa. Pois estes são flexíveis, eficientes e de fácil implementação.

Entre os inúmeros trabalhos relacionados à abordagem heurística multiobjetiva,

destacam-se como os mais utilizados, os Algoritmos Genéticos, os quais são baseados na teoria da evolução.

Nos Algoritmos Genéticos Multiobjetivos, a cada geração, ou iteração, tem-se um conjunto de indivíduos, ou soluções-pais. Para gerar uma nova população (soluções-filho), os operadores genéticos (tais como recombinação e mutação) são aplicados sobre as soluções-pai. Dessa forma, obtêm-se uma nova população de soluções formada pelas soluções-pai e soluções-filho. No final de cada iteração, os indivíduos mais aptos sobrevivem e o restante é descartado.

Entre os inúmeros Algoritmos Genéticos Multiobjetivos, destacamos: VEGA, MOGA, NPGA, SPEA e NSGA.

No *Vector Evaluated Genetic Algorithm* (VEGA), proposto por Schaffer (1985), a cada geração, um grupo de indivíduos que supera os demais de acordo com um dos n objetivos é selecionado, até que n grupos sejam formados. Então os n grupos são misturados conjuntamente e os operadores genéticos são aplicados para formar a próxima geração.

No *Multiobjective Genetic Algorithm* (MOGA), proposto por Fonseca e Fleming (1993), cada indivíduo i é classificado em um nível de acordo com o número de indivíduos que esse indivíduo i domina. Todos os indivíduos não dominados são classificados no nível 1. A aptidão de cada indivíduo é atribuída de acordo com uma interpolação entre o melhor e o pior nível. A aptidão final atribuída a todos os indivíduos de um mesmo nível é a mesma e igual à média da aptidão do próprio nível. Dessa forma, todos os indivíduos do mesmo nível são indiferentes entre si.

No *Niche Pareto Genetic Algorithm* (NPGA), proposto por Horn *et al.* (1994), a seleção dos indivíduos se dá através de um torneio baseado no conceito de dominância de Pareto. Dois indivíduos são selecionados e comparados com um subconjunto da população de soluções, sendo selecionado para a próxima geração aquele que não for dominado.

No algoritmo *Non-dominated Sorting Genetic Algorithm* (NSGA), proposto por Srivivas e Deb (1995), os indivíduos são classificados em níveis de acordo com seu grau de dominância, tal como nos algoritmos anteriores. Entretanto, é atribuído um valor de aptidão a cada indivíduo de acordo com seu nível e sua distância em relação às outras soluções do mesmo nível, a chamada distância de multidão. A seleção é feita através de torneios utilizando o valor de aptidão até que todas as vagas para a próxima geração sejam preenchidas.

No algoritmo *Strength Pareto Evolutionary Algorithm* (SPEA), proposto por Zitzler (1999), é utilizada a seleção baseada na relação de dominância para avaliar e selecionar as soluções. Para avaliar essa relação de dominância e classificar os indivíduos em níveis de dominância, o SPEA usa um conjunto adicional da população. Porém, ao contrário dos algoritmos anteriores, os quais descartam os indivíduos não selecionados, ele utiliza os indivíduos não dominados da população da geração anterior para determinar a aptidão dos indivíduos da população corrente.

Apesar de serem os mais utilizados, os Algoritmos Genéticos nem sempre são os mais recomendados (GUIMARÃES *et al*., 2007). Além destes, também encontramos outras heurísticas para problemas Multiobjetivos, tais como o procedimento Busca Tabu Multiobjetivo (BTM) proposto por Arroyo (2002).

Além da heurística Busca Tabu, inicialmente proposta para problemas mono-objetivos, outras também podem ser utilizadas tais como os métodos heurísticos VNS e *Variable Neighborhood Descent* (VND).

O método heurístico VNS, exemplificado pela Figura 1, proposto por Mladenovic e Hansen (1997), é um método que consiste em explorar o espaço de soluções por meio de trocas sistemáticas das estruturas de vizinhança. Contrariamente às outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória. Ele explora vizinhanças diferentes da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado.

O VNS também inclui um procedimento de busca local a ser aplicado sobre a solução corrente. No presente trabalho adotou-se o método VND, exemplificado pela Figura 2 para fazer a busca local, também utilizado pelo VNS original.

```
Procedimento VNS (f(.),N(.),r,s);
1    Seja s_o uma solução inicial;
2    Seja r o número de estruturas diferentes de vizinhança;
3    s ← s_o ;
4    Enquanto ( critério de parada não satisfeito ) faça
5            k ← 1;
6            Enquanto ( k ≤ r) faça
7                    Gere um vizinho qualquer s' ∈ N^(k)(s);
8                    s'' ← BuscaLocal (s');
9                    se ( f(s'') < f(s) ) então
10                           s ←s'';
11                           k ← 1;
12                   Senão
13                           k ← k + 1;
14                   Fim se
15           Fim-enquanto;
16   Fim-enquanto;
17   Retorne s;
Fim VNS;
```

Figura 1: Heurística VNS

```
Procedimento VND (f(.),N(.),r,s);
1    Seja r o número de estruturas diferentes de vizinhança;
2    k ← 1 ;
3    Enquanto ( k ≤ r) faça
4            Encontre um vizinho s' ∈ N^(k)(s);
5            se ( f(s') < f(s) ) então
6                    s ←s';
7                    k ← 1;
8            Senão
9                    k ← k + 1;
10           Fim se
11   Fim-enquanto;
12   Retorne s;
Fim VND;
```

Figura 2: Heurística VND

O VND é uma técnica usada para o refinamento de soluções iniciais. Isto é feito através da análise da região de soluções factíveis por meio de trocas sistemáticas nas estruturas da vizinhança de uma solução corrente. Este método aceita apenas as soluções de melhora da solução corrente, retornando à primeira estrutura quando uma melhor solução é encontrada.

4. Metodologia

Para uma gama de problemas, encontrar a solução ótima global de instâncias de grande dimensão pode ser inviável. Para problemas desta natureza, como o SPWA, o uso de métodos exatos se torna bastante restrito. Este é o motivo pelo qual inúmeros trabalhos concentram esforços na utilização de heurísticas para solucionar problemas desse nível de complexidade. Heurísticas podem ser definidas como sendo uma técnica que procura boas soluções, ou seja, próximas do ótimo global.

Logo, também apresentamos um modelo heurístico multiobjetivo baseado no algoritmo VNS exemplificado pela Figura 1 na seção 3.

O algoritmo proposto chamado de VNS-Multiobjetivo (VNSM), exemplificado pela Figura 3, começa sua execução partindo de um conjunto de soluções iniciais S_0. Este conjunto é gerado através de um procedimento parcialmente guloso, utilizando um torneio binário. O número de soluções (*NSol*) do conjunto S_0 foi definido de forma empírica, considerando a complexidade do problema.

Procedimento *VNSM (f(.),N(.),r)*	
1	*Seja S_o um conjunto de soluções iniciais;*
2	*Seja r o número de estruturas diferentes de vizinhança;*
3	**para (** *l < NSol* **) faça**
4	*$s \leftarrow s_l \in S_0$;*
5	*it* ←0*;*
6	**Enquanto (** *it < It_SM*) **faça**
7	*k ← 1;*
8	**Enquanto (** k ≤ *r)* **faça**
9	*Gere um vizinho qualquer $s_l' \in N^{(k)}(s_l)$;*
10	*$s'' \leftarrow$ BuscaLocal (s');*
11	**se (** *$f(s_l'') < f(s_l)$* **) então**
12	*$s_l \leftarrow s_l''$;*
13	*k ← 1;*
14	**Senão**
15	*k ← k + 1;*
16	*it ← it*+1;
17	**Fim se**
18	**Fim-enquanto;**
19	**Fim-enquanto;**
20	*$s_l \in S$;*
21	**Fim-para**
22	**Retorne** *S*;
Fim *VNSM*	

Figura 3: heurística VNS-Multiobjetivo

Após esse passo, cada solução $s_l \in S_0$ é avaliada segundo uma função de avaliação com pesos variáveis. O valor do peso de cada objetivo varia a cada iteração, dessa forma a cada iteração um objetivo possui maior ou menor importância para determinar uma solução.

Em seguida, a primeira solução, s_1, do conjunto S_0 é selecionada e seleciona-se aleatoriamente um vizinho s_1' dentro da vizinhança $N^{(1)}(s_1)$ da solução s_1 corrente. Esse vizinho s_1' é então submetido a um procedimento de busca local retornando a solução s_1''. Se a solução ótima local, s_1'', for melhor que a solução s_1 corrente, a busca continua de s_1'' recomeçando da primeira estrutura de vizinhança $N^{(1)}(s_1'')$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N^{(k+1)}(s_l)$. Esta etapa é repetida até que se obtenha um número máximo de iterações sem melhora, *It_SM*, definido empiricamente. Ao final desta etapa, a melhor

solução encontrada é adicionada ao conjunto de soluções finais *S*.

Após o final da etapa anterior, o procedimento retorna ao seu início, selecionando a próxima solução s_l do conjunto S_0 e repete-se todo o procedimento anterior.

O algoritmo é repetido para todas as soluções s_i do conjunto de soluções iniciais S_0.

4.1 Representação de uma solução

Uma solução pode ser representada por uma matriz $(s_l)_{[Worker \times (Job+1)]}$. Dessa forma, os movimentos das estruturas de vizinhança se tornam mais simples e naturais. Assim, o algoritmo torna-se menos complexo e a avaliação das soluções é facilitada. O conjunto de soluções *S* é formado pela união de todas as *NSol* (número de soluções) matrizes s_l.

A Figura 4 exemplifica uma possível solução s_l. A primeira coluna apresenta os trabalhadores. A coluna "**Util**" indica se o funcionário *i* realiza uma tarefa *j* qualquer. As colunas "**Tarefas**" indicam as tarefas alocadas a cada trabalhador e sua respectiva sequência.

No exemplo da Figura 4, o valor 1 da primeira linha (**Trabalhador 1**) e coluna "**Util**" indica que o funcionário 1 está sendo utilizado. Os valores das outras colunas indicam que o funcionário 1 executará as tarefas 1, 3 e 4, nesta ordem. O valor 0 na segunda linha (**Trabalhador 2**), coluna "**Util**" indica que o trabalhador 2 está ocioso. Logo ele não executará nenhuma tarefa. Para o trabalhador 4 temos que ele executará as tarefas 2 e 5, nesta ordem.

		Util	Tarefas				
Trabalhador	1	1	1	3	4	0	0
	2	0	0	0	0	0	0
	3	0	0	0	0	0	0
	4	1	2	5	0	0	0

Figura 4: Representação de uma solução s_l.

4.2 Geração da solução Inicial

A solução inicial gera *NSol* matrizes s_l através de um procedimento parcialmente guloso utilizando um torneio binário.

Para cada matriz solução s_l, a cada iteração do procedimento, uma tarefa *j* é selecionada. Além disso, dois funcionários também são escolhidos aleatoriamente. Aquele funcionário que executar a tarefa *j* selecionada em menor tempo é escolhido. Por exemplo, para a tarefa 1 são escolhidos os funcionários 2 e 4. O funcionário 2 executa a tarefa 1 em 3 horas e o funcionário 4 em 2 horas. Neste caso a tarefa *j* será executa pelo funcionário 4.

Este procedimento é executado até que todas as tarefas sejam atribuídas a algum funcionário *i* para todas as matrizes $s_l \in S$.

4.3 Avaliação de uma solução

Uma solução $s_l \in S$ é avaliada segundo a função de avaliação $f(s_l)$. Esta função busca minimizar o número de funcionários e o *makespan* através de penalidades. A função de avaliação $f(s_l)$ é exemplificada pela Eq. (1).

$$\min \; f(s_l) = \frac{w_l}{\Delta^{\sigma}}\sigma(s_l) + \left(\frac{1-w_l}{\Delta^{\tau}}\right)\tau(s_l) \qquad \forall\, s_l \in S \qquad (1)$$

Sendo:

w_l : Penalidade da solução s_l.

σ : *makespan*.

τ : Número de trabalhadores utilizado.
Δ^{σ} : Fator de correção para o parâmetro σ.
Δ^{τ} : Fator de correção para o parâmetro τ.

O valor de w_l é alterado a cada iteração do algoritmo VNSM para cada solução s_l avaliada. Inicialmente ele assume o valor 1, e a cada iteração, para cada solução s_l, seu valor é decrescido segundo a Eq. 2. Nesta equação, a cada iteração, o novo valor de w_l é igual ao valor da penalidade da iteração anterior, w_{l-1}, menos o valor da divisão de 1 pelo número de soluções adotado. Dessa forma, a cada iteração, altera-se o espaço de busca privilegiando algum objetivo.

$$w_l = w_{l-1} - \frac{1}{NSol} \quad (2)$$

Na função de avaliação, Eq. 1, também adotamos fatores de correção, Δ. Estes fatores garantem que as varáveis de decisão τ e σ estejam dentro da mesma ordem de grandeza.

4.4 Estruturas de vizinhança

Para explorar o espaço de soluções do problema, ou seja, para encontrar uma solução s_l', dita vizinha de s_l, foram utilizados dois movimentos apresentados a seguir:

Movimento Realoca Tarefa, $N^{RT}(s_l)$: Este movimento consiste em escolher aleatoriamente um trabalhador *i* que está sendo utilizado. Em seguida, a última tarefa *j* é selecionada e realocada para outro trabalhador escolhido aleatoriamente. Na Figura 5, o trabalhador 1 é selecionado e a tarefa 4 é realocada para o trabalhador 4.

		Util	Tarefas				
Trabalhador	1	1	1	3	**4**	0	0
	2	0	0	0	0	0	0
	3	0	0	0	0	0	0
	4	1	2	5	0	0	0

		Util	Tarefas				
Trabalhador	1	1	1	3	**0**	0	0
	2	0	0	0	0	0	0
	3	0	0	0	0	0	0
	4	1	2	5	**4**	0	0

Figura 5: Movimento Realoca Tarefa, $N^{RT}(s_l)$

Movimento Trabalhador, $N^{T}(s_l)$: Este movimento consiste em escolher aleatoriamente um trabalhador *i* que está sendo utilizado e realocar todas as suas tarefas para os demais trabalhadores, aleatoriamente. Na Figura 6 o trabalhador 1 é selecionado, a tarefa 3 é realocada para o trabalhador 4 e a tarefa 1 é realocada para o trabalhador 2.

		Util	Tarefas				
Trabalhador	1	1	1	3	0	0	0
	2	1	4	0	0	0	0
	3	0	0	0	0	0	0
	4	1	2	5	0	0	0

		Util	Tarefas				
Trabalhador	1	0	0	0	0	0	0
	2	1	4	1	0	0	0
	3	0	0	0	0	0	0
	4	1	2	5	**3**	0	0

Figura 6: Movimento Trabalhador, $N^T(s_l)$

4.5 Busca local

A busca local é aplicada a todas as soluções $s_l \in S$. Ela consiste na aplicação do VND (vide Fig. 2).

Inicialmente, considera-se um conjunto de r = 2 vizinhanças distintas, cada qual definida por um dos tipos de movimentos definidos na seção 4.4. Na sequencia, a partir de uma solução s_l', um determinado número de vizinhos s_l'' da primeira vizinhança são analisados.

Em seguida, parte-se para a segunda estrutura de vizinhança. Novamente, um determinado número de vizinhos é analisado. O vizinho s_l'' que apresentar maior melhora, em relação à s_l', segundo a função de avaliação da seção 4.3, é escolhido, e encerra-se a busca local.

5. Resultados

Para testar o método proposto foram utilizadas 4 instâncias-teste. Elas podem ser encontradas em http://www.4shared.com/zip/m7xmUfpi/Instance_-_SPWA.html. Elas diferem entre si pelo número de trabalhadores e de tarefas. A tabela 1 apresenta algumas características das instâncias. As colunas ***#Worker*** e ***#Job*** mostram, respectivamente, o número de trabalhadores e o número de tarefas que precisam ser executadas.

Tabela 1: Características das instâncias-teste

	#Worker	***#Job***
Instância 1	5	10
Instância 2	10	50
Instância 3	15	100
Instância 4	25	100

O algoritmo VNSM proposto foi desenvolvido na linguagem C, usando o compilador C++ *Builder* 5.0 da *Borland*. Ele foi testado em um PC *Pentium Core* 2 *Duo*, com 2,4 GHz e 4 GB de RAM sob plataforma *Windows* 7.

Inicialmente o algoritmo proposto foi submetido a uma bateria preliminar de testes para calibrar os diversos parâmetros existentes. Tais parâmetros são: o número de iterações de cada execução (*It_SM*), número de soluções (*NSol*), a penalidade da solução s_l (w_l) e o valor dos fatores de correção (Δ^{σ} e Δ^{τ}) .

Após a determinação dos parâmetros, o algoritmo foi submetido a uma bateria de testes com 100 execuções para cada instância teste.

A tabela 2 mostra o tempo gasto, em segundos, pelo algoritmo proposto. Na linha *Menor* tem-se o menor tempo gasto entre as 100 execuções. Na linha *Médio*, mostra-se o tempo médio e na linha Maior tem-se o *Maior* tempo gasto pelo algoritmo proposto.

Tabela 2: Tempo de execução.

	Instância 1	***Instância 2***	***Instância 3***	***Instância 4***
Menor	5,5	78,5	61,4	146,7
Médio	5,8	101,8	146,8	211,7
Maior	6,8	136,5	265,7	277,3

O gráfico da Figura 7 mostra os resultados obtidos depois de 100 execuções do VNSM. O eixo das abscissas representa o tempo total gasto para executar todas as tarefas. O eixo das ordenadas representa o número de trabalhadores utilizados. O gráfico apresenta o melhor conjunto de soluções encontrado entre as execuções do VNSM (Melhor). Ele também apresenta o conjunto de soluções composto pelos valores médios calculados (Médio). Para calcular os valores médios, a cada execução do VNSM, um conjunto de soluções, *S*, é encontrado. A primeira solução, s_1, de cada execução do VNSM é coletada, e o valor Médio, mostrado na Figura 7, corresponde à média dos valores dos objetivos de todas as s_1 soluções encontradas nas 100 execuções do VNSM. Este cálculo é repetido para todas as segundas soluções s_2, e para as demais soluções do conjunto *S*. Nota-se que com o aumento da complexidade da instância o tempo de execução do algoritmo aumenta e sua eficiência diminui. Como não é de conhecimento do autor outro trabalho adotando uma abordagem multiobjetivo, e a resolução por método exato é inviável computacionalmente, ainda não há parâmetros para determinar o quanto são boas as soluções encontradas.

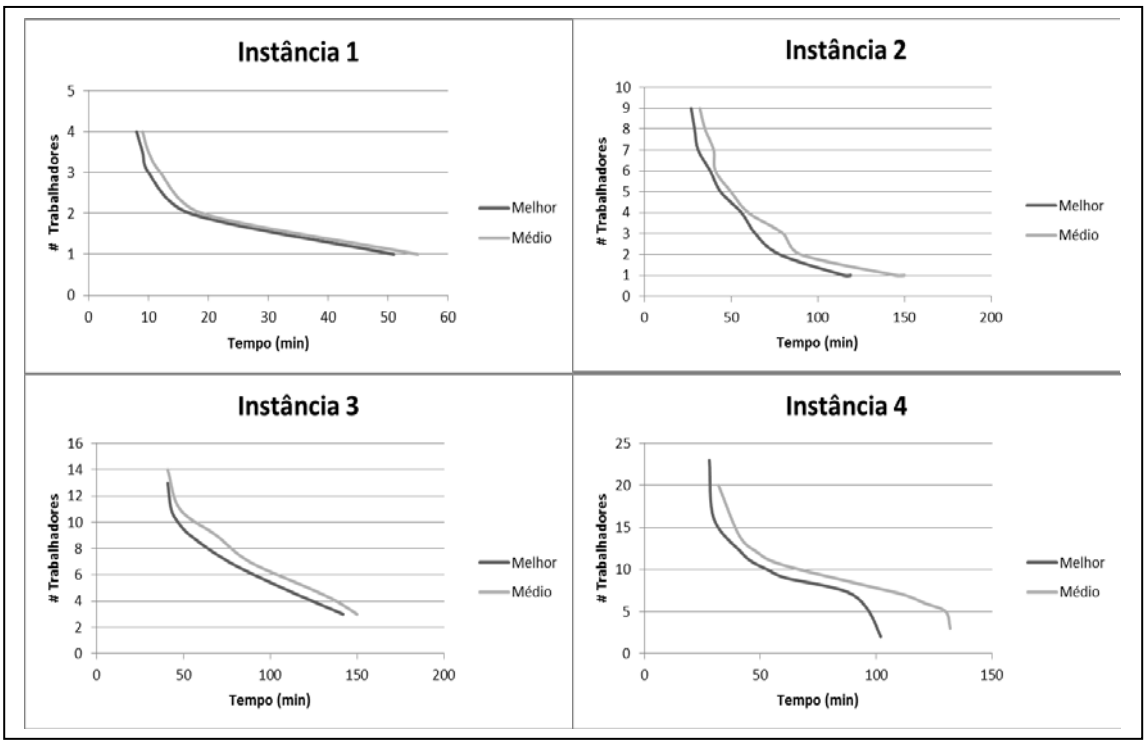

Figura7: Resultados do VNSM.

6. Conclusões

Este trabalho propôs um método de solução para o problema de sequenciamento com

alocação de trabalhadores (SPWA), adaptando o algoritmo heurístico mono-objetivo, VNS, para um algoritmo multiobjetivo, VNSM. O objetivo deste problema é minimizar o número de funcionários e o tempo total gasto para executar todas as tarefas.

Ao contrário dos trabalhos encontrados na literatura, este trabalho tratou o SPWA como um problema multiobjetivo, ou seja, um problema composto por vários objetivos conflitantes entre si.

Assim, apresenta-se ao final da execução do algoritmo, um conjunto de soluções eficientes, as quais ou priorizam alguma meta ou, então, estão equilibradas entre os diversos valores dos objetivos. O gestor, responsável pela tomada de decisão, torna-se responsável pela escolha de qual alternativa mais se adapta à realidade operacional da empresa naquele instante.

O algoritmo proposto, VNSM, baseado no algoritmo VNS, parte de uma solução inicial parcialmente gulosa através de um torneio binário. Para a busca local utiliza o algoritmo VND e duas estruturas de vizinhanças para explorar o espaço de busca das soluções. Dessa forma, procura-se um algoritmo flexível simples e eficiente que seja capaz de se adaptar a qualquer tipo de problema multiobjetivo.

Pelos testes realizados percebe-se que o algoritmo proposto é capaz de encontrar um conjunto de soluções eficientes para o problema em estudo. O algoritmo gerou soluções com valores próximos uns dos outros na quase totalidade dos testes. Uma proposta para trabalhos futuros é a utilização de novos movimentos para o SPWA e o teste do VNSM para outros problemas.

Agradecimentos

O autor agradece ao IFMG e à FAPEMIG, pelo apoio financeiro para o desenvolvimento deste trabalho de pesquisa.

Referências

Arroyo, J. E. C. (2002), *Heurísticas e metaheurísticas para otimização combinatória multiobjetivo*. Tese de Doutorado. Programa de Pós-Graduação em Engenharia Elétrica, Unicamp, Campinas, SP.

FIESP. Encargos trabalhistas sobre folha de salários e seus impactos no Brasil e no mundo. Relatório Técnico. Fiesp/Decomtec. São Paulo – SP, 2011.

Fonseca, C. M. e Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Anais do Fifth International Conference on Genetic Algorithms*, p. 416-423, San Mateo, USA, 1993.

Fonseca, C. M. e Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, v. 3, n. 1, p. 1-16.

Guimarães, I. F.; Pantuza, G. e Souza, M. J. F. (2007). Modelo de simulação computacional para validação dos resultados de alocação dinâmica de caminhões com atendimento de metas de qualidade e de produção em minas a céu aberto. *Anais do XIV Simpósio de Engenharia de Produção - SIMPEP*, 11 p. (CD-ROM), Bauru.

Horn, J.; Nafpliotis, N. e Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. *Anais do the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, p. 82-87, Piscataway, USA.

Iima, H. e Sannomiya, N. (2001). Module Type Genetic Algorithm for Modified Scheduling Problems with Worker Allocation. *Anais do American Control Conference.* Arlington, VA.

Iima, H. e Sannomiya, N. (2002). Proposition of Module Type Genetic Algorithm and Its Application to Modified Scheduling Problems with Worker Allocation. *IEEE Japan*, v.122-C, p. 409-416.

Mladenovic, N. e Hansen, P. (1997). Variable Neighborhood Search. *Computers and Operations Research*, n. 24 pp. 1097–1100.

Osawa, A. e Ida, K. (2005). Scheduling Problem with Worker Allocation using Genetic Algorithm. *Japan-Australia workshop on intelligent and evolutionary systems*, pp.1-8.

Pantuza Jr, G. (2011). *Métodos de otimização multiobjetivo e de simulação aplicadas ao problema de planejamento operacional de lavra em minas a céu aberto*. Dissertação de Mestrado em Engenharia Mineral, Universidade Federal de Ouro Preto, UFOP, Ouro Preto, MG, 2011.

Pareto, V.. *Cours D'Economie Politique*, vol. 1, F. Rouge, 1896.

Schafler, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. Genetic Algorithms and their Applications: *Anais do First International Conference on Genetic Algorithms*, v. I, p. 93-100.

Srivivas, N. e Deb, K. (1995). *Multiobjective optimization using non dominated sorting in genetic algorithms*. Evolutionary Computation, v. 2, n. 3, p. 221-248, 1995.

Tan, S.; Weng, W. e Fujimura, S. (2009). Scheduling of Worker Allocation in the Manual Labor Environment with Genetic Algorithm. *Anais do International MultiConference of Engineers and Computer Scientists – IMECS.* Hong Kong, v. 1.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: methods and applications.* Tese de Doutorado, Federal Institute of Technology Zurich, Zurich, Swiss.