Gest. Prod., São Carlos, v. 23, n. 1, p. 132- 145, 2016
http://dx.doi.org/10.1590/0104-530X1432-14

Uma abordagem multiobjetivo para o problema de sequenciamento e alocação de trabalhadores

A multi-objective approach to the scheduling problem with workers allocation

Guido Pantuza Júnior[1]

Resumo: O presente trabalho trata do problema de sequenciamento e alocação de trabalhadores (SPWA). No SPWA, objetiva-se minimizar o número de trabalhadores e o tempo total gasto para executar todas as tarefas (*makespan*). Para tanto, propõem-se o uso de dois modelos diferentes de programação matemática e uma heurística VNS-Multiobjetivo baseada no método heurístico VNS. Como os objetivos são conflitantes entre si, os métodos propostos geram um conjunto de soluções eficientes, cabendo ao gestor escolher qual solução deve ser adotada. Os métodos propostos obtiveram resultados satisfatórios para a resolução do SPWA, 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.

Palavras-chave: Otimização multiobjetivo. *Épsilon*-restrito. Método das somas ponderadas. VNS-Multiobjetivo. SPWA.

Abstract: *This paper addresses the scheduling problem with workers allocation (SPWA). In SPWA, the objective is to minimize the number of workers and the total time taken to perform all tasks (makespan). To this end, we propose the use of two different mathematical programming models and a VNS-based multi-objective heuristic. As the objectives are conflicting, the proposed methods generate a set of efficient solutions, and the manager chooses which solution should be adopted. The proposed methods achieved satisfactory results for the SPWA resolution, showing that it is possible to use a limited staff and finish all tasks in a timely manner, thereby utilizing the resources of a firm optimally.*

Keywords: *Multi-objective optimization. Epsilon-restricted. Method of weighted sums. VNS-Multi-objective. SPWA.*

1 Introdução

O mercado mundial está cada vez mais competitivo e exigente. Além disso, as crises mundiais e a ameaça de uma recessão na Europa ameaçam a saúde financeira das empresas. Diante desse cenário, as empresas necessitam reduzir o seu custo produtivo. Uma forma de reduzir este custo é pelo 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. Isto se deve aos recentes aumentos dos salários acima da inflação, adicionados aos encargos sociais e benefícios. Logo, as empresas procuram a redução do número de funcionários sem afetar os prazos de entrega.

Esse problema enfrentado pelas empresas é conhecido 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.

Em geral, a maioria dos trabalhos encontrados na literatura, trata esse problema como um problema monobjetivo. Isto significa que a função de avaliação multiobjetivo é transformada em uma função de avaliação monobjetivo. Essa transformação, geralmente, é feita atribuindo diferentes pesos aos diferentes objetivos. Trabalhos como o de Abensur (2012) utilizam esta abordagem. Ele utilizou a programação por metas para tratar um problema de otimização multiobjetivo. Dessa forma, ele converteu os múltiplos objetivos em um único.

Porém, os objetivos são conflitantes entre si, ou seja, não existe uma solução única que otimize todas elas ao mesmo tempo. De fato, quanto menor o número de funcionários, maior o tempo necessário para executar todas as tarefas, o contrário também é verdade. Logo, adotando o método de resolução

[1] Instituto Federal de Minas Gerais – IFMG, Av. Minas Gerais, 5391, Ouro Verde, Governador Valadares, MG, Brasil, e-mail: gpantuza@gmail.com

Recebido em Abr. 30, 2014 — Aceito em Out. 13, 2014

Suporte financeiro: FAPEMIG e IFMG.

monobjetivo, temos apenas uma única solução, que privilegia um objetivo em detrimento do outro.

Além disso, na prática, os gestores responsáveis pela designação dos trabalhadores e sequenciamento das tarefas, convivem, diariamente, com a mudança no número de funcionários disponíveis e de tarefas que precisam ser executas. Por isso, os gestores necessitam de uma solução flexível. Ela deve ser capaz de absorver as repentinas mudanças nos parâmetros de forma rápida, mantendo o sistema otimizado.

A otimização multiobjetivo, busca um conjunto de soluções eficientes, o que torna o sistema mais flexível, uma vez que ele apresenta soluções diferentes. O gestor, ao resolver o problema, adotando este método, tem à sua disposição diversas soluções, cada uma utilizando um número diferente de funcionários. As soluções disponíveis, obtidas pelo método multiobjetivo, estão mais próximas da realidade, tornando a tarefa do gestor mais rápida e flexível. Ou seja, ele consegue responder à falta de um funcionário sem precisar resolver o modelo novamente, economizando tempo.

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. A partir dessas observações feitas no setor industrial e de serviços, principalmente em pequenas e médias empresas, observamos que, geralmente, estas empresas possuem um número de funcionários e tarefas reduzido. Logo, este trabalho propõe uma abordagem de resolução multiobjetivo para o SPWA, utilizando métodos exatos e heurísticos. Estes devem gerar soluções flexíveis que possam ser implementadas.

Entre os diversos métodos exatos para a resolução de um problema multiobjetivo, este trabalho foca os modelos de Programação Matemática utilizando o método de peso variável e o método *épsilon*-restrito (ε-restrito). Pois, segundo Coello et al. (2002), estes métodos são os mais utilizados. Além disso, segundo, Tan et al. (2009), o SPWA é um problema NP-difícil, ou seja, para problemas com dimensões maiores, não é possível encontrar a solução ótima global em tempo computacional hábil. Logo, um modelo heurístico multiobjetivo baseado no algoritmo *Variable Neighborhood Search* – VNS também é proposto.

O presente trabalho está organizado como segue. Na seção 2, descreve-se o problema em estudo. Na seção 3, o referencial teórico. Na seção 4, apresentam-se os modelos de Programação Matemática utilizados. Na seção 5, é apresentado o algoritmo VNS-Multiobjetivo proposto. As apresentações dos problemas testes e dos resultados são feitas na seção 6. A conclusão é apresentada na seção 7.

2 O problema de sequenciamento com alocação de trabalhadores

O problema de sequenciamento com alocação de trabalhadores também é conhecido como *scheduling 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 sequência 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 o instante de término da última tarefa executada 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 têm o mesmo nível de esforço. Os trabalhadores, que são utilizados, possuem uma taxa de utilização máxima e mínima. 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.

O SPWA pode ser considerado como um caso especial do problema de *job shop scheduling*. Tal problema, em sua forma mais geral, é caracterizado pela alocação de tarefas a algum tipo de recurso necessário para sua execução. Este recurso geralmente é uma estação de trabalho ou algum tipo de máquina.

Na literatura, diversos trabalhos tratam do problema *job shop scheduling*, tais como, Silva & Rentes (2012), Tavares Neto & Godinho Filho (2013) e Mello & Ferreira (2014).

O trabalho de Silva & Rentes (2012) apresenta um novo modelo de otimização do problema *job shop* por meio da otimização do arranjo físico e o aplica em algumas empresas do setor metal mecânico. O objetivo do modelo consiste em desenvolver alternativas que estejam em harmonia com conceitos e princípios da filosofia de produção enxuta.

Tavares Neto & Godinho Filho (2013) utilizam um método formado por dois estágios para tratar do *job shop* em um ambiente de uma máquina com possibilidade de terceirização. No primeiro estágio, propõe-se que as tarefas sejam sequenciadas, utilizando-se a regra SPT (*Shortest Processing Time*). No segundo estágio, é proposto um algoritmo baseado em ACO (*Ant Colony Optimization*).

Mello & Ferreira (2014) abordam o problema *job shop*, com a minimização do *makespan*. Eles utilizaram modelos de simulação computacional de um sistema de manufatura com dois cenários diferentes. No primeiro, eles não consideram a utilização de máquinas alternativas, no segundo, eles adotam a inserção de máquinas alternativas para executar as mesmas tarefas.

Segundo Osawa & Ida (2007), a principal diferença entre o SPWA e o *job shop* clássico, é a utilização do recurso mão de obra, considerando diferentes níveis de qualificação e uma baixa taxa de utilização. Esta baixa taxa de utilização pode ser atribuída à queda de rendimento devido ao cansaço, às condições ambientais do local de trabalho (temperatura e iluminação, por exemplo), além das leis trabalhistas que garantem o direito do trabalhador ao descanso durante a jornada de trabalho.

Na literatura, encontramos alguns trabalhos que abordam o SPWA. Iima & Sannomiya (2001) propuseram uma heurística para resolução do SPWA fundamentada no Algoritmo Genético chamada *Module Type Genetic Algorithm* (MTGA). Osawa & Ida (2005) utilizaram um Algoritmo Genético, porém propuseram um novo método de seleção da população sobrevivente. Osawa & Ida (2007) adotaram o algoritmo proposto por Osawa & Ida (2005). Entretanto, eles consideraram que cada trabalhador possui um nível de habilidade diferente para cada máquina utilizada.

Todos os trabalhos citados anteriormente adotaram abordagens mono-objetivas. Entretanto, encontramos na literatura, diversos trabalhos que adotaram métodos multiobjetivos, em sua maioria, utilizando meta-heurísticas para a resolução do problema *job shop*.

Ishibushi & Murata (1998) propuseram um Algoritmo Genético de busca local. Este consiste na inserção de soluções não dominadas na população (conjunto de soluções) a partir do método proposto, buscando uma população mais adaptada. Suresh & Mohanasundaram (2004) propuseram um algoritmo multiobjetivo, ao qual chamaram de *Pareto Archived Simulated Annealing* (PASA). Neste algoritmo, foi utilizado um novo método de perturbação das soluções para encontrar soluções vizinhas, chamado *Segment-Randon Service* (SRI).

Garcia et al. (2004) adotaram um método heurístico e um exato. O método heurístico é baseado em um algoritmo evolutivo multiobjetivo. O modelo de programação matemática é não linear e o método de resolução utilizado é o *épsilon*-restrito. Seus resultados mostraram que ambas as técnicas alcançaram soluções similares e são capazes de evitar ótimos locais. Iima (2005) propôs um Algoritmo Genético com uma nova forma de seleção da população mais adaptada e um novo operador de mutação baseados em distribuições de probabilidade. Lei & Wu (2005) utilizaram um algoritmo evolutivo multiobjetivo (CMOEA). Este é baseado na *crowding mesure*, ou distância de multidão. Ele faz uso da distância de multidão para ajustar a população externa e atribuir aptidão diferente para as pessoas. Eles consideraram dois objetivos, minimizar o *makespan* e o atraso total da data de entrega.

Xia & Wu (2005) utilizaram um algoritmo evolutivo baseado na otimização por enxame de partícula. Tal algoritmo imita o comportamento de pássaros voando e seus meios de troca de informações. Ele combina tal algoritmo com a busca local, utilizando o algoritmo *Simulated Annealing*. Quian et al. (2006) desenvolveram um algoritmo baseado no Algoritmo Memético com base em evolução diferencial chamado MADE. Neste algoritmo, primeiro, uma regra é utilizada para converter o problema de modo que a evolução diferencial possa ser aplicada. Em segundo lugar, um mecanismo de evolução paralela é aplicado para explorar a vizinhança, executando uma busca local. Além disso, o conceito de dominância de Pareto é utilizado para seleção das soluções. Eles consideraram como objetivos minimizar o atraso máximo.

Li et al. (2010) desenvolveram um método híbrido multiobjetivo que combina duas meta-heurísticas, Busca Tabu e *Variable Neighborhood Search* (VNS). Este trabalho propõe um algoritmo híbrido de Busca Tabu (HTSA) considerando três objetivos de minimização: o tempo máximo de conclusão (*makespan*), a carga horária total do equipamento e da carga de trabalho da máquina. A estrutura de vizinhança proposta combina dois critérios de adaptação, uma para o método de busca local, outra para a seleção das máquinas. Em seguida, um método de designação de tarefas é executado. Além disso, um algoritmo VNS é utilizado adotando três estruturas de vizinhanças.

Ruiz et al. (2012) basearam-se em um Algoritmo Genético Multiobjetivo chamado *Elitist Non-Dominated Sorting Genetic Algorithm* (NSGA-II). Eles consideraram três objetivos: minimizar o *makespan*, o custo de energia e acidentes de trabalho.

3 Referencial teórico

O SPWA é composto por dois objetivos conflitantes (minimizar o número de funcionários e o instante de término da última tarefa executada). Assim, não existe uma solução única que otimize todas elas ao mesmo tempo. Em vista disso, encontrar soluções viáveis que otimizem simultaneamente todos os objetivos é o maior desafio da otimização multiobjetivo.

Para autores como Zitzler (1999), Fonseca & Fleming (1995) e Arroyo (2002), para a resolução de problemas dessa natureza deve-se, assim, buscar um conjunto de soluções eficientes. Neste caso, a tomada de decisão será de responsabilidade do gestor, que poderá escolher a solução que melhor se adapta às necessidades dentre as soluções eficientes. Este critério será utilizado pelo responsável, ou gestor, para a tomada de decisão, ou seja, ele poderá ponderar entre as diferentes soluções conflitantes.

O conjunto de soluções eficientes também é conhecido como soluções Pareto-ótimas. 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, $s \leq s^*$ para todos os objetivos;
- *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*.

Para os problemas multiobjetivos, os métodos convencionais de otimização monobjetivo não são eficientes (COELLO et al., 2002). Portanto, a busca por novos métodos de otimização que consigam vencer o grande desafio deste tipo de problema tornou-se necessária. Uma forma de vencer este desafio é a utilização dos métodos exatos de otimização multiobjetivo. Estes métodos surgiram da necessidade de encontrar soluções com prioridades, ou pesos, associados aos objetivos. Entre os métodos exatos clássicos utilizados, para resolver esta gama de problemas destacamos: Método da soma ponderada e Método *épsilon*-restrito (ε-restrito).

O método da soma ponderada consiste na transformação do problema multiobjetivo em um problema monobjetivo por meio da atribuição de pesos para cada objetivo. Para se alcançar as soluções Pareto-ótimas, este problema deve ser resolvido iterativamente. Ou seja, a cada iteração, novos pesos são atribuídos aos diferentes objetivos. Sendo, geralmente, a soma dos pesos igual a 1.

Segundo Arroyo (2002), a principal desvantagem deste método é que ele não consegue gerar todas as soluções Pareto-ótimas quando o espaço objetivo é não convexo. O método de programação por metas, semelhante ao método da soma ponderada, também consiste na transformação do problema multiobjetivo em um problema monobjetivo. Isto se dá pela atribuição de pesos para cada objetivo. Para esse método, considera-se que cada meta possui uma importância diferente na otimização representada por meio de pesos. Quanto maior a importância da meta, maior será o seu peso.

O método exato épsilon-restrito foi inicialmente proposto por Ritzel et al. (1994). Ele consiste na otimização do objetivo mais importante, representado pela Equação 1, sujeitando-se às restrições dos outros objetivos, representadas pela Equação 2.

Considerando, em um problema de minimização, f_1 como sendo o objetivo mais importante, temos:

minimizar

$$f_1(x) \tag{1}$$

Sujeito a:

$$f_i(x) \leq \varepsilon_i \quad i = 2,3,...,q \tag{2}$$

no qual, ε_i é o limite superior do objetivo *i* e *q* o número de objetivos.

Para construir o conjunto Pareto-ótimo, mesmo quando o espaço objetivo é não convexo, deve-se apenas variar o limite superior. Porém, se este limite não é adequado, o subconjunto de possíveis soluções obtido pode ser vazio, ou seja, não existe solução viável.

Outra forma de resolver problemas multiobjetivos é por métodos heurísticos. Entre os inúmeros trabalhos relacionados à abordagem heurística multiobjetivo, 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 & 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 por meio 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 & 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 por meio 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 pelo Quadro 1, proposto por Mladenovic & 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 meta-heurísticas 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 pelo Quadro 2 para fazer a busca local, também utilizado pelo VNS original.

O VND é uma técnica usada para o refinamento de soluções iniciais. Isto é feito pela 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 solução melhor é encontrada.

Entre os inúmeros VNS aplicados à otimização multiobjetivo, destacamos Geiger (2004), denominado MOVNS. Neste procedimento, a estrutura de vizinhança e a solução a ser explorada são escolhidas de forma aleatória, mudando a

Quadro 1. Heurística VNS.

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

Quadro 2. Heurística VND.

	Procedimento VND (*f*(.), *N*(.), *r*, *s*);
1	Seja *r* o número de estruturas diferentes de vizinhança;
2	$k \leftarrow 1$;
3	**Enquanto** ($k \leq r$) **faça**
4	Encontre um vizinho $s' \in N^{(k)}(s)$;
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**.**	

cada iteração. Uma solução, entre as soluções não dominadas, é selecionada. A partir desta solução, novas soluções vizinhas são geradas.

Em Ottoni et al. (2011), os autores tratam o problema *job shop scheduling* em que n tarefas devem ser processadas em uma única máquina, que pode processar uma tarefa de cada vez. Eles propuseram a incorporação de um novo método de intensificação ao algoritmo MOVNS, proposto por Geiger (2004). Esta intensificação consiste em uma perturbação na solução, gerando novas soluções.

Arroyo et al. (2011) também tratam do problema *job shop scheduling*, considerando uma máquina com janelas de tempo com tempo de preparação de máquina. Os autores consideraram dois objetivos, o primeiro é minimizar os atrasos e adiantamentos das tarefas, e o segundo é minimizar o fluxo total. Para resolver o problema, foi proposto um novo algoritmo MOVNS combinado com um processo de perturbação, diferente do proposto por Ottoni et al. (2011).

4 Modelos de programação matemática

4.1 Método *épsilon*-restrito

Para a resolução do SPWA utilizando o método *épsilon*-restrito, adotamos como objetivo principal o instante de término da última tarefa executada. O número de funcionários é tido como objetivo secundário, por isso, a cada execução do modelo matemático proposto, o número máximo de funcionários que pode ser utilizado (ε) é reduzido. Para este modelo, consideramos os seguintes parâmetros de entrada:

Job : Conjunto de tarefas.

Worker : Conjunto de trabalhadores.

T_{ij} : Tempo necessário para o funcionário *j* executar a tarefa *i*.

ε : Número máximo de trabalhadores.

H : Horizonte de planejamento.

Tu_i : Taxa de utilização máxima do trabalhador *i*.

Tl_i : Taxa de utilização mínima do trabalhador *i*.

Sejam as seguintes variáveis de decisão:

σ : Instante de término da última tarefa executada.

$$x_{ij} : \begin{cases} 1 \text{ se o funcionário } i \text{ executa a tarefa } j. \\ 0 \text{ caso contrário.} \end{cases}$$

$$y_i : \begin{cases} 1 \text{ se o funcionário } i \text{ é utilizado.} \\ 0 \text{ caso contrário.} \end{cases}$$

O modelo de programação matemática proposto relativo ao SPWA é apresentado pelas Equações 3 a 10:

Função Objetivo:

A Equação 3 visa minimizar o instante de término da última tarefa.

$$\min f = \sigma \tag{3}$$

Sujeito às restrições:

A Equação 4 garante que toda tarefa *j* será executada apenas uma vez por um único trabalhador *i*.

$$\sum_{i \in \text{Worker}} x_{ij} = 1 \qquad \forall\, j \in Job \tag{4}$$

A Equação 5 estipula o número máximo de trabalhadores *i* que podem ser utilizados. O valor de ε é reduzido a cada iteração do modelo proposto.

$$\sum_{i \in \text{Worker}} y_i \leq \varepsilon \tag{5}$$

A Equação 6 determina o instante de término da última tarefa *j* executada pelo operador *i*.

$$\sum_{j \in \text{Job}} T_{ij} x_{ij} \leq \sigma \qquad \forall\, i \in Worker \tag{6}$$

A Equação 7 define se o trabalhador *i* está executando alguma tarefa *j*.

$$\frac{\sum_{j \in Job} x_{ij}}{|\text{Job}|} \leq y_i \qquad \forall\, i \in Worker \tag{7}$$

As Equações 8 e 9 definem o intervalo para a taxa de utilização trabalhador *i* que está sendo utilizado.

$$\frac{\sum_{j \in Job} x_{ij} T_{ij}}{H} \leq y_i Tu_i \qquad \forall\, i \in Worker \tag{8}$$

$$\frac{\sum_{j \in Job} x_{ij} T_{ij}}{H} \geq y_i Tl_i \qquad \forall\, i \in Worker \tag{9}$$

As Equações 10 a 12 asseguram o domínio das variáveis de decisão.

$$x_{ij} \in \{0,1\} \;\forall\, i \in Worker \text{ e } \forall\, j \in Job \tag{10}$$

$$y_i \in \{0,1\} \quad \forall i \in Worker \tag{11}$$

$$\sigma \in R^+ \tag{12}$$

4.2 Método das somas ponderadas

Para a resolução do SPWA, utilizando o método das somas ponderadas, adotamos como objetivos o instante de término da última tarefa executada (*makespan*) e o número de funcionários utilizados. Neste método, a cada iteração *k*, o valor da penalidade (w_k) da função objetivo é alterado. Inicialmente ele assume o valor 1, e a cada iteração, seu valor é decrescido segundo a Equação 13. Nesta equação, a cada iteração, o novo valor de w_k é igual ao valor da penalidade da iteração anterior, w_{k-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.

$$w_k = w_{k-1} - \frac{1}{NSol} \tag{13}$$

Sendo:
w_k: Penalidade da função objetivo na iteração *k*.
NSol : Número de soluções.

As variáveis de decisão e os parâmetros são os mesmos do modelo anterior, exceto pela retirada do parâmetro (número máximo de trabalhadores) e pela inclusão do parâmetro *w* (penalidade do objetivo). O modelo de programação matemática proposto, para o método das somas ponderadas, é semelhante ao método da seção 4.1, porém eles diferem pela substituição da Equação 3, referente à função objetivo, pela Equação 14 e pela exclusão da restrição representada pela Equação 5.

Função Objetivo

A Equação 14 visa minimizar o instante de término da última tarefa.

$$\min f = w_k \sigma + (1 - w_k) \sum_{i \in \text{Worker}} y_i \tag{14}$$

5 Algoritmo proposto

Para uma gama de problemas, encontrar a solução ótima global de problemas 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 pelo Quadro 1 na seção 3.

O algoritmo proposto chamado de VNS-Multiobjetivo (VNSM), exemplificado pelo Quadro 3, começa sua execução partindo de um conjunto de soluções iniciais S_0. Este conjunto é gerado por meio 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.

Depois desse passo, cada solução $s_1 \in S_0$ 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 à 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 $N^{(k+1)}(s_1)$. Esta etapa é repetida até que se obtenha um número máximo de iterações sem melhora, *It_SM*, definido

Quadro 3. Heurística VNS-Multiobjetivo.

	Procedimento VNSM (*f*(.), *N*(.), *r*, S_0)
1	Seja S_0 um conjunto de soluções iniciais;
2	Seja *r* o número de estruturas diferentes de vizinhança;
3	**para** ($l < NSol$) **faça**
4	$s \leftarrow s_1 \in S_0$;
5	$it \leftarrow 0$;
6	**Enquanto** ($it < It\_SM$) **faça**
7	k $\leftarrow 1$;
8	**Enquanto** ($k \leq r$) **faça**
9	Gere um vizinho qualquer $s_1' \in N^{(k)}(s_1)$;
10	$s'' \leftarrow$ Busca Local (s');
11	**se** ($f(s_1'') < f(s_1)$) **então**
12	$s_1 \leftarrow s_1''$;
13	$k \leftarrow 1$;
14	**Senão**
15	$k \leftarrow k + 1$;
16	$it \leftarrow it+1$;
17	**Fim se**
18	**Fim-enquanto**;
19	**Fim-enquanto;**
20	$s_1 \in S$;
21	**Fim-para**
22	Retorne *S*;
Fim VNSM.	

empiricamente. Ao final desta etapa, a melhor solução encontrada é adicionada ao conjunto de soluções finais *S*.

Depois do final da etapa anterior, o procedimento retorna ao seu início, selecionando a próxima solução s_1 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.

5.1 Representação de uma solução

Uma solução pode ser representada por uma matriz $(s_1)[Worker \times (Job+1)]$. 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_1.

A Tabela 1 exemplifica uma possível solução s_1. 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 Tabela 1, 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.

5.2 Geração da solução inicial

A solução inicial gera *NSol* matrizes s_1 por meio de um procedimento parcialmente guloso utilizando um torneio binário. Para cada matriz solução s_1, 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 $s_1 \in S$.

5.3 Avaliação de uma solução

Uma solução $S_l \in S$ é avaliada segundo a função de avaliação $f(s_1)$. Esta função busca minimizar o número de funcionários e o *makespan* por meio de penalidades. A função de avaliação $f(s_1)$ é exemplificada pela Equação 15.

$$\min f(s_l) = \frac{w_l}{\Delta^{\sigma}}\sigma(s_l) + \left(\frac{1-w_l}{\Delta^{\tau}}\right)\tau(s_l) \quad \forall s_l \in S \qquad (15)$$

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_1 é alterado a cada iteração do algoritmo VNSM para cada solução s_1 avaliada. Inicialmente ele assume o valor 1, e a cada iteração, para cada solução s_1, seu valor é decrescido segundo a Equação 16. Nesta equação, a cada iteração, o novo valor de s_1 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.

$$w_l = w_{l-1} - \frac{1}{NSol} \qquad (16)$$

Na função de avaliação, Equação 16, também adotamos fatores de correção, Δ. Estes fatores garantem que as varáveis de decisão τ e σ estejam dentro da mesma ordem de grandeza.

5.4 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

Tabela 1. Representação de uma solução s_1.

		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

trabalhador escolhido aleatoriamente. Na Tabela 2, o trabalhador 1 é selecionado e a tarefa 4 é realocada para o trabalhador 4.

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 Tabela 3, o trabalhador 1 é selecionado, a tarefa 3 é realocada para o trabalhador 4 e a tarefa 1 é realocada para o trabalhador 2.

5.5 Busca local

A busca local é aplicada a todas as soluções $s_l \in S$. Ela consiste na aplicação do VND (vide Quadro 3). Inicialmente, considera-se um conjunto de $r = 2$ vizinhanças distintas, cada qual definida por um dos tipos de movimentos definidos na seção 5.4. Na sequência, a partir de uma solução s_l', um determinado número de vizinhos s_l'' da primeira vizinhança é analisado.

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 5.3, é escolhido, e encerra-se a busca local.

6 Resultados

Para testar os modelos propostos, foram utilizados 4 problemas testes, que podem ser encontrados em http://www.4shared.com/zip/m7xmUfpi/Instance_-_SPWA.html. Eles diferem entre si pelo número de trabalhadores e de tarefas. A Tabela 4 apresenta algumas características dos problemas. As colunas **#Worker** e **#Job** mostram, respectivamente, o número de trabalhadores e o número de tarefas que precisam ser executadas.

Os modelos de programação matemática desenvolvidos na Seção 4 foram implementados no aplicativo de otimização LINGO 10.0, interfaceando com planilhas do EXCEL 2010. O algoritmo VNSM proposto foi desenvolvido na linguagem C, usando o compilador C++ *Builder* 5.0 da *Borland*. Todos os testes foram realizados em um PC *Pentium Core* 2 *Duo*, com 2,4 GHz e 4 GB de RAM sob plataforma *Windows* 7.

6.1 Resultados do modelo exato

O conjunto de soluções S, Pareto-ótimas, encontrado, utilizando o método ε-restrito, para o Problema 1, é apresentado na Tabela 5. A coluna *Épsilon* apresenta o valor adotado para ε, que representa o número máximo de funcionários. A coluna *#Worker* apresenta o número de funcionários utilizados. A coluna *makespan* apresenta, em minutos, o instante de término da última tarefa.

De acordo com a Tabela 5, para o Problema 1, podemos observar que obtivemos os mesmos resultados para os dois métodos exatos, ambos com 5 soluções diferentes cada. Na solução 1, adotamos o valor ε = 0. Utilizaremos 5 trabalhadores que executarão todas as tarefas em 8 minutos. Para a

Tabela 2. Movimento Realoca Tarefa, $N^{RT}(s_1)$.

		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
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

Tabela 3. Movimento Trabalhador, $N^{T}(s_1)$.

		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
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

solução 5, adotamos o valor ε = 4. Utilizaremos 1 trabalhador que executará todas as tarefas em 43 minutos.

A Tabela 6 apresenta o conjunto de soluções Pareto-ótimas, utilizando o método ε-restrito e das somas ponderadas, respectivamente, para o Problema 2. Neste problema adotamos 10 soluções diferentes. O valor de ε varia de 0 a 9 (número máximo de funcionários disponíveis).

Tabela 4. Características dos problemas testes.

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

A Tabela 7 apresenta os resultados do modelo exato para o Problema 3. O conjunto de soluções Pareto-ótimas é formado por 7 soluções diferentes. O valor de ε varia de 0 a 14, sendo reduzido em duas unidades para cada solução.

Os resultados do modelo exato, para o Problema 4, é apresentado na Tabela 8. Neste problema adotamos 10 soluções diferentes. O valor de ε varia de 0 a 24, sendo reduzido em três unidades para cada solução.

Para o Problema 4, ao contrário dos demais problemas (1, 2 e 3), não se encontrou um conjunto de soluções Pareto-ótimas. Isto se deve ao fato de o problema SPWA ser NP-difícil (TAN et al., 2009). Ou seja, para problemas com dimensões maiores, não é possível encontrar a solução ótima global em tempo computacional hábil. Neste caso, adotamos

Tabela 5. Resultado do Método Exato para o Problema 1.

		Épsilon-Restrito			Somas Ponderadas	
	Solução	**Épsilon - ε**	**#Worker**	***makespan* (min)**	**#Worker**	***makespan* (min)**
Problema 1	1	0	5	8	5	8
	2	1	4	10	4	10
	3	2	3	13	3	13
	4	3	2	20	2	20
	5	4	1	43	1	43

Tabela 6. Resultados Método Exato para o Problema 2.

		Épsilon-Restrito			Somas Ponderadas	
	Solução	**Épsilon - ε**	**#Worker**	***makespan* (min)**	**#Worker**	***makespan* (min)**
Instância 2	1	0	10	20	10	20
	2	1	9	22	9	22
	3	2	8	25	8	25
	4	3	7	28	7	28
	5	4	6	33	6	33
	6	5	5	39	5	39
	7	6	4	49	4	49
	8	7	3	65	3	65
	9	8	2	98	2	98
	10	9	1	215	1	215

Tabela 7. Resultados Método Exato para o Problema 3.

		Épsilon-Restrito			Somas Ponderadas	
	Solução	**Épsilon - ε**	**#Worker**	***makespan* (min)**	**#Worker**	***makespan* (min)**
Instância 3	1	0	15	27	15	27
	2	2	13	31	13	31
	3	5	10	39	8	49
	4	7	8	49	7	58
	5	9	6	65	5	81
	6	12	3	130	4	107
	7	14	1	430	1	430

Tabela 8. Resultados Método Exato para o Problema 4.

	Solução	Épsilon-Restrito			Somas Ponderadas	
		Épsilon - ε	#Worker	*makespan* (min)	#Worker	*makespan* (min)
Instância 4	1	0	25	16	25	16
	2	3	22	18	22	18
	3	6	19	22	19	22
	4	9	16	25	15	28
	5	12	13	31	11	37
	6	15	10	40	10	40
	8	18	7	57	8	49
	9	21	4	100	3	135
	10	24	1	433	1	433

Tabela 9. Tempo de execução em segundos.

Método		Problema 1	Problema 2	Problema 3	Problema 4
Épsilon-restrito		5	149	1972	6365
Somas Ponderadas		5	136	1861	5992
VNSM	**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

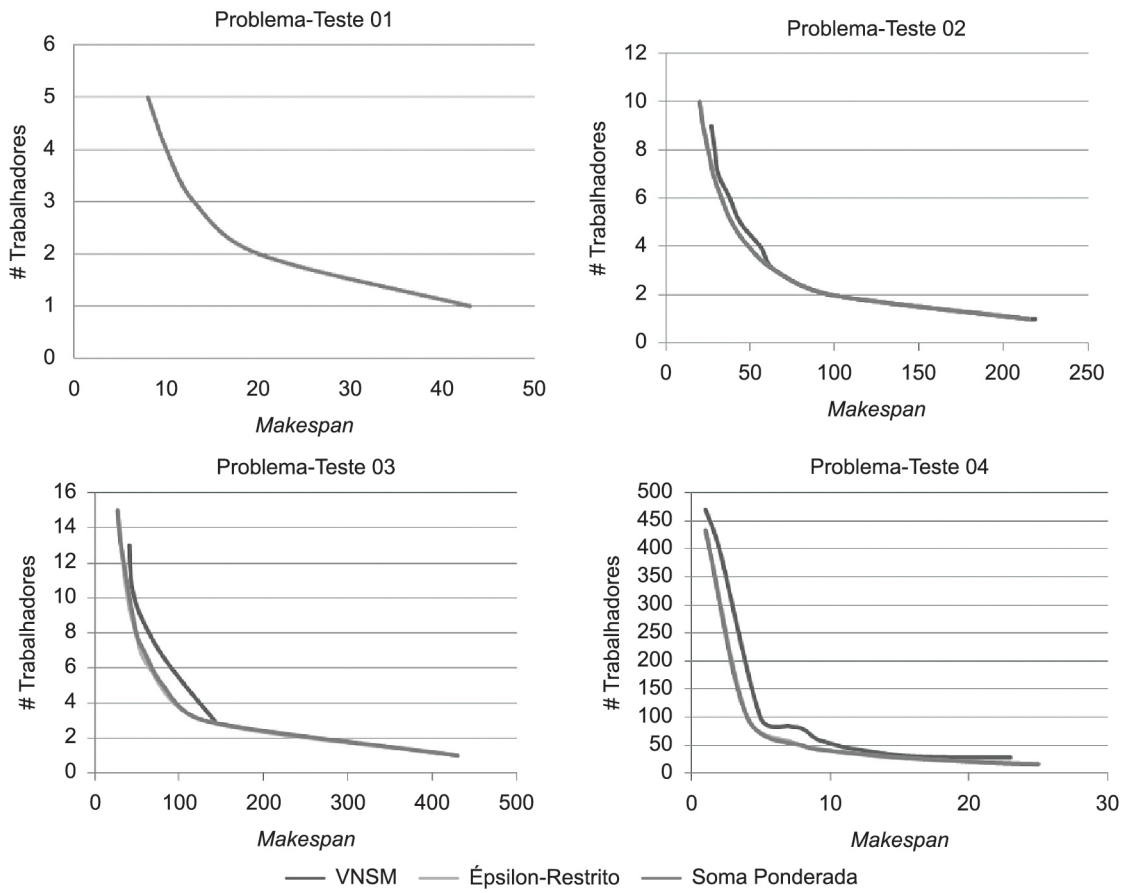

Figura 1. Resultados do VNSM.

como tempo máximo de execução 900 segundos para cada solução gerada. Para ambos os métodos exatos, somente para as soluções 1 e 10 (primeira e última solução), conseguimos resolver o problema e encontrar uma solução em menos de 900 segundos, sendo: para o método *épsilon*-restrito, o tempo de execução para a solução 1 de 41 segundos, e para a solução 2 de 24 segundos; e para o método das somas ponderadas, o tempo de execução para a solução 1 de 39 segundos, e para a solução 2 de 28 segundos.

6.2 Resultados do VNSM

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}). Depois da determinação dos parâmetros, o algoritmo foi submetido a uma bateria de testes com 100 execuções para cada problema teste.

O gráfico da Figura 1 mostra os resultados obtidos depois de 100 execuções do VNSM e dos métodos exatos. 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. Neste gráfico, notamos que, para o problema teste 1, o VNS e os dois métodos exatos (*épsilon*-restrito e soma ponderada) obtiveram o mesmo resultado. Para os outros problemas testes, observamos que os resultados dos métodos exatos foram semelhantes e que o VNS foi capaz de encontrar soluções próximas às dos métodos exatos.

A Tabela 9 mostra o tempo gasto, em segundos, pelos métodos utilizados para a resolução do SPWA. Para os VNSM, 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. Podemos observar que o método exato das somas ponderadas, para todos os problemas, obteve um desempenho melhor ou igual, que o *épsilon*-restrito.

7 Conclusões

Este trabalho apresentou dois modelos exatos de programação matemática e um algoritmo multiobjetivo para a resolução do problema de sequenciamento com alocação de trabalhadores (*scheduling problem with worker allocation* – SPWA).

Os modelos de programação matemática propostos foram baseados em dois métodos clássicos de resolução de problemas multiobjetivos: o método de resolução multiobjetivo ε-restrito, que se baseia na otimização do objetivo mais importante sujeitando-se às restrições dos outros objetivos; e o método iterativo da soma ponderada, que consiste na transformação do problema multiobjetivo em um problema monobjetivo por meio da atribuição de pesos para cada objetivo. O algoritmo multiobjetivo proposto (VNSM) foi baseado na heurística VNS combinada com o algoritmo de busca local VND.

Os resultados mostram que é possível otimizar o número de funcionários e o tempo máximo de execução das tarefas utilizando a abordagem multiobjetivo. Os dois métodos exatos obtiveram resultados semelhantes, sendo que o método da soma ponderada obteve melhor desempenho quanto ao tempo computacional. O VNSM foi capaz de encontrar soluções próximas às dos métodos exatos, provando ser uma boa opção, principalmente para problemas mais complexos.

Ao apresentar várias soluções atendendo a diferentes objetivos, disponibilizam-se, ao gestor, alternativas para sua tomada de decisão. Dessa forma, é possível escolher a solução que melhor se adapta à realidade operacional da empresa.

Agradecimentos

O autor agradece à FAPEMIG e ao IFMG o apoio financeiro, imprescindível para a realização deste projeto de pesquisa.

Referências

ABENSUR, E. O. Um modelo multiobjetivo de otimização aplicado ao processo de orçamento de capital. *Gestão & Produção*, v. 19, n. 4, p. 747-758, 2012. http://dx.doi.org/10.1590/S0104-530X2012000400007

ARROYO, J. E. C. *Heurísticas e metaheurísticas para otimização combinatória multiobjetivo*. 2002. 256 f. Tese (Doutorado)-Programa de Pós-Graduação em Engenharia Elétrica, Universidade Estadual de Campinas - Unicamp, Campinas, 2002.

ARROYO, J. E. C.; OTTONI, R. S.; OLIVEIRA, A. P. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, v. 281, n. 29, p. 5-19, 2011. http://dx.doi.org/10.1016/j.entcs.2011.11.022

COELLO, C. A. C; LAMONT, G. B.; VAN VELDHUIZEN, D. A. *Evolutionary algorithms for solving multi-objective problems*. Boston: Kluwer Academic Publishers, 2002. Relatório Técnico.

FONSECA, C. M.; FLEMING, P. J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 5., 1993, San Mateo, USA. *Proceedings*… p. 416-423.

FONSECA, C. M.; FLEMING, P. J. An overview of evolutionary algorithms in multiobjective optimization.

Evolutionary Computation, v. 3, n. 1, p. 1-16, 1995. http://dx.doi.org/10.1162/evco.1995.3.1.1

GARCIA, J. M. C. et al. Hybrid heuristic and mathematical programming in oil pipelines networks. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, 2004. p. 1479-1486. v. 2.

GEIGER, M. J. Randomized variable neighborhood search for multi objective optimization. In: DESIGN AND EVALUATION OF ADVANCED HYBRID META-HEURISTICS -EU/ME, 2004, Nottingham, UK. *Proceedings*… p. 34-42.

GUIMARÃES, I. F.; PANTUZA, G.; SOUZA, M. J. F. 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. In: SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO - SIMPEP, 14., 2007, Bauru. *Anais*... 11 p. CD-ROM.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In: IEEE CONFERENCE ON EVOLUTIONARY COMPUTATION, IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE, 1., 1994, Piscataway, USA. *Proceedings*… p. 82-87.

IIMA, H. Preposition of selection in a genetic algorithm for a job shop rescheduling problem. In: INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTICRITERION OPTIMIZATION, 3., 2005, Guanajuato, Mexico, 2005. *Proceedings*…

IIMA, H.; SANNOMIYA, N. Module type genetic algorithm for modified scheduling problems with worker allocation. In: AMERICAN CONTROL CONFERENCE, 2001, Arlington, VA. *Proceedings*…

ISHIBUSHI, H.; MURATA, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, v. 28, n. 3, p. 392-403, 1998.

LEI, D.; WU, Z. Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling. *International Journal of Advanced Manufacturing Tecnology*, v. 30, n. 1-2, p. 112-117, 2005. http://dx.doi.org/10.1007/s00170-005-0029-6

LI, J.; PAN, Q.; LIANG, Y. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, v. 59, p. 647-662, 2010. http://dx.doi.org/10.1016/j.cie.2010.07.014

MELLO, M. H.; FERREIRA, J. C. E. Avaliação de presença de recursos alternativos em plano de processos para melhorar o desempenho de sistemas manufatura. *Produção Online*, v. 14, n. 2, p. 648-678, 2014. http://dx.doi.org/10.14488/1676-1901.v14i2.1467

MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. *Computers and Operations Research*, v. 24, n. 11, p. 1097-1100, 1997. http://dx.doi.org/10.1016/S0305-0548(97)00031-2

OTTONI, R. S.; ARROYO, J. E. C.; SANTOS, A. G. Algoritmo vns multi-objetivo para um problema de programação de tarefas em uma máquina com janelas de entrega. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL - SBPO, XLIII., 2011, Ubatuba, SP. *Anais*...

OSAWA, A.; IDA, K. Scheduling problem with worker allocation using genetic algorithm. In: JAPAN-AUSTRALIA WORKSHOP ON INTELLIGENT AND EVOLUTIONARY SYSTEMS, 2005. p. 1-8.

OSAWA, A.; IDA, K. A solution method of scheduling problem with worker allocation by a genetic algorithm. *IEEJ Transactions on Electronics, Information and Systems*, v. 127, n. 5, p. 755-761, 2007. http://dx.doi.org/10.1541/ieejeiss.127.755

QUIAN, B. et al. Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. *International Journal of Advanced Manufacturing Tecnology*, v. 35, n. 9-10, p. 1014-1027, 2006. http://dx.doi.org/10.1007/s00170-006-0787-9

RITZEL, B. J.; EHEART, J. W.; RANJITHAN, S. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. Water Resources Research, v. 30, n. 5, p. 1589-1603, 1994. http://dx.doi.org/10.1029/93WR03511

RUIZ, S.; CASTRILLÓN, O. D.; SARACHE, W. A. Una metodología multiobjetivo para optimizar un ambiente job shop. *Información Tecnológica*, v. 23, n. 1, p. 35-46, 2012. http://dx.doi.org/10.4067/S0718-07642012000100005

SCHAFFER, J. Multiple objective optimization with vector evaluated genetic algorithms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 1., 1985. *Proceedings*… p. 93-100. v. 1.

SILVA, A. L.; RENTES, A. F. Um modelo de projeto de layout para ambientes job shop com alta variedade de peças baseado nos conceitos da produção enxuta. *Gestão & Produção*, v. 19, n. 3, p. 531-541, 2012. http://dx.doi.org/10.1590/S0104-530X2012000300007

SRIVIVAS, N.; DEB, K. Multiobjective optimization using non dominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, n. 3, p. 221-248, 1995.

SURESH, R. K.; MOHANASUNDARAM, K. M. Pareto archived simulated annealing for job shop scheduling with multiple objectives. International *Journal of Advanced Manufacturing Technology*, v. 29, n. 1-2, p. 184-196, 2004. http://dx.doi.org/10.1007/s00170-004-2492-x

TAN, S.; WENG, W.; FUJIMURA, S. Scheduling of worker allocation in the manual labor environment with genetic algorithm. In: INTERNATIONAL MULTICONFERENCE OF ENGINEERS AND COMPUTER SCIENTISTS – IMECS, 2009, Hong Kong. *Proceedings*… v. 1.

TAVARES NETO, R. F.; GODINHO FILHO, M. Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida. *Gestão & Produção*, v. 20, n. 1, p. 76-86, 2013. http://dx.doi.org/10.1590/S0104-530X2013000100006

XIA, W.; WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, v. 48, n. 2, p. 409-425, 2005. http://dx.doi.org/10.1016/j.cie.2005.01.018

ZITZLER, E. *Evolutionary algorithms for multiobjective optimization*: methods and applications. 1999. 122 f. Tese (Doutorado)-Federal Institute of Technology Zurich, Zurich, Swiss, 1999.