UM MÉTODO HEURÍSTICO APLICADO AO PROBLEMA DE PROGRAMAÇÃO DE SONDAS DE PRODUÇÃO

Miguel Angel Fernández Pérez
miguelfp177@yahoo.com

Fernanda Maria Pereira Raupp
fraupp@puc-rio.br

Departamento de Engenharia Industrial
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro, RJ- Brasil - 22451-900

RESUMO

Neste trabalho, resolvemos o problema de programação de sondas de perfuração e produção de petróleo no ambiente *open shop*, minimizando simultaneamente o *makespan*, a carga de trabalho máxima dos recursos e a carga de trabalho total dos recursos, através da aplicação de um novo método heurístico. O método proposto foi inspirado no método de Newton para problemas de otimização contínua multiobjetivo de Fliege *et al*. (2008). Serão apresentados experimentos numéricos com o método proposto em duas instâncias baseadas em dados reais do problema de escalonamento de tarefas de sondas em poços de petróleo. Ainda, os resultados obtidos foram comparados com os resultados conhecidos de métodos da literatura que resolvem o mesmo problema e consideram apenas um único objetivo.

PALAVRAS CHAVE. Programação de sondas de perfuração e produção de petróleo, Open shop, Otimização multiobjetivo, Não-dominância, Método de Newton multiobjetivo.

ABSTRACT

In this work we solve the problem of scheduling drilling rigs and oil production in the *open shop* environment, minimizing simultaneously the makespan, the maximum workload of the resources and total workload of the resources, through the application of a new heuristic method. The proposed method was inspired by the Newton's method for continuous multi-objective optimization problems of Fliege *et al*. (2008). Numerical experiments with the proposed method are presented for two instances based on real data of the problem of scheduling tasks rigs in oil wells. Also, the obtained results are compared with the known results of methods from the literature that solve the same problem and consider exactly one objective.

KEYWORDS. Scheduling drilling rigs and oil production, Open shop problem, Multi-objective optimization, Non-dominance, Multi-objective Newton's method.

1. Introdução

Um problema de programação ou escalonamento de produção tem como finalidade determinar uma sequência factível de processamento de um conjunto de operações por um conjunto de recursos ao longo de um intervalo de tempo, visando otimizar uma ou mais medidas de desempenho, geralmente associadas ao fator tempo ou ao balanceamento de uso dos recursos. Os problemas de escalonamento são uma classe importante dos problemas de otimização combinatória e em geral pertencem à classe NP-difícil (PINEDO, 2008).

Segundo Natal (2003), diante de toda a complexidade inerente aos processos de perfuração e produção de poços, os tomadores de decisão defrontam-se constantemente com a difícil tarefa de gerenciar recursos, sob a missão de utilizá-los da forma mais eficiente e econômica possível. Torna-se, então, relevante encontrar soluções para a questão da alocação das sondas às fases dos poços a serem perfurados que sejam economicamente interessantes. Esse problema real, encontrado na perfuração de poços de petróleo e gás natural, é também um problema típico em outras indústrias, em que para a execução de uma sequência de atividades existem diversos recursos e procura-se a melhor forma de executá-las do ponto de vista econômico. É importante destacar que outras dimensões de análise podem ser incorporadas nesse problema além dos custos envolvidos, como aspectos relativos à qualidade, flexibilidade, risco e outras.

Borchardt (2002) estudou um problema em um campo de petróleo que possui uma frota de sondas disponíveis, mas que não são o bastante para intervir em todos os poços do campo que reduziram a produção ou estão parados. O aluguel de sondas a custo maior e diário se torna uma opção para contornar tal dificuldade. As demandas por sondas e a produção de cada poço são conhecidas, e o objetivo do problema é programar o atendimento da frota para se obter o maior retorno de óleo produzido no horizonte de tempo, de modo que a programação das atividades também evite acidentes ambientais. Nesse caso, cada poço possui um grau de risco de acidentes, que é levado em conta e descontado em sua produção. Para a solução do problema, a autora propõe uma formulação matemática, porém, não o resolve usando um método exato e sim aplicando algoritmos genéticos com transgenética computacional, que é o foco do trabalho.

O trabalho de Vasconcelos (2007) é um projeto da produção em 33 poços de petróleo pertencentes a um mesmo campo em fase de desenvolvimento, o qual tem um horizonte de tempo determinado. O projeto se baseia nas atividades de *perfuração* e *completação*, que necessitam de sondas e barcos para serem realizadas. São dados do problema restrições tecnológicas e número de recursos, são consideradas restrições de precedência das atividades e tempo de intervenção de cada recurso para cada atividade em cada poço. O objetivo do problema é determinar a sequência de atividades tal que a duração total dos projetos (*makespan)* seja mínima. O autor apresentou uma modelagem matemática para o problema, para a qual obteve uma solução ótima usando o método exato *Branch and Bound*. Porém, a crescente dificuldade em resolver problemas com maior horizonte de planejamento através de métodos exatos fez com que o autor optasse pelo desenvolvimento de métodos heurísticos como a Programação por Restrições, GRASP e um Algoritmo Genético para achar soluções do problema, provando ao final de seu trabalho a aplicabilidade destes métodos.

Neste estudo preliminar, um novo método heurístico de solução é proposto para resolver o problema de programação de sondas de perfuração e produção de petróleo no ambiente *open shop* multiobjetivo, considerando a minimização simultaneamente das seguintes medidas de desempenho:

f_1 = tempo de conclusão de todas as tarefas ou *makespan*,
f_2 = carga de trabalho máxima nas máquinas (recursos) e
f_3 = carga de trabalho total das máquinas (recursos).

Os resultados da abordagem proposta serão comparados com os resultados dos métodos usados por Vasconcelos (2007) em instâncias baseadas em dados reais. Apesar de o método proposto que

considera a otimização simultânea de três objetivos a ser comparado com métodos de solução existentes que consideram um único objetivo, nesse estudo se valorizou mais a natureza das instâncias do problema de escalonamento *open shop*, ou seja, a natureza do problema real sendo resolvido.

2. Programação de sondas no ambiente *open shop*

O problema de programação ou de escalonamento no ambiente *open shop* tem como finalidade determinar a sequência de processamento de n tarefas $J_1, J_2, \dots, J_n$ em m máquinas $M_1, M_2, \dots, M_m$. Supõe-se por hipótese que a sequência de processamento de uma tarefa em uma máquina é arbitrária.

Considera-se ainda, para este trabalho, a extensão do problema de escalonamento *open shop* clássico: $O_m/prec, M_j/C_{max}$, onde existe precedência entre as tarefas e máquinas específicas M_j para processar certas tarefas. O processamento da tarefa J_i sobre a máquina M_j é denotado como operação ou atividade (i, j).

Após a atividade de exploração e localização dos poços de petróleo, inicia-se a perfuração desses poços para alcançar os reservatórios de petróleo. Essa perfuração é realizada por sondas que, dependendo da localização dos reservatórios, podem ser terrestres ou marítimas.

Os poços perfurados devem ser preparados para operar de forma segura e econômica durante toda sua vida produtiva ou de injeção. Estas operações destinadas a equipar o poço para produção de óleo ou gás, ou ainda injetar fluidos no reservatório, são denominados *completação*. Ao passo que são usados, os poços tendem a necessitar de intervenções de manutenção, e a este conjunto de atividades denomina-se *workover*. Todas essas atividades são realizadas pelos mesmos recursos da perfuração, sondas de produção, e quando são realizados em poços marítimos também contam com a ajuda de barcos.

2.1. Representação de soluções

Uma solução é representada por dois vetores: o vetor de sequência de operações (atividades) s^* e o vetor alocação de máquinas (recursos) v^*. A seguir, vamos descrever em duas etapas a construção dos vetores s^* e v^*.

Etapa 1: Geração da sequência de operações

O processo começa gerando o vetor de nível prioridade π correspondente ao conjunto de operações. Se J é o número total de operações, então, o vetor π é formado por uma permutação aleatória de $\{1, 2, \dots, J\}$. Usando a procedimento de sequenciamento proposto por Liu *et al*. (2009), podemos obter uma sequência factível s^*.

Etapa 2: Alocação de máquinas

Agora que a sequência foi fixada, podemos gerar o vetor v^* mediante o procedimento de alocação de máquinas, proposto por Zhang *et al*. (2006). Está alocação é feita escolhendo ao acaso uma máquina para cada operação.

2.2. Geração de vizinhança para máquinas

Se a máquina alocada da operação (i, j) for h, então uma vizinhança pode ser gerada através da atribuição do conjunto de máquinas k capazes de realizar a operação (i, j) tal que os tempos de processamento da operação (i, j) nas máquinas k (p_{ijk}) e h (p_{ijh}) satisfaçam $p_{ijk} \leq p_{ijh}$, tendo em vista que um dos objetivos é minimizar a carga de trabalho total das máquinas.

2.3. Geração de vizinhança para sequências

No algoritmo de busca local, o tipo de vizinhança pode influenciar fortemente o desempenho do algoritmo. Embora a escolha de uma vizinhança, que contém um grande número

de soluções candidatas aumente a probabilidade de encontrar boas soluções, o tempo de computacional para encontrar vizinhos disponíveis também irá aumentar (XIA & WU, 2005). Um método simples para gerar soluções vizinhas é o método de troca entre duas operações de uma sequência, que geram sequências factíveis.

3. Otimização multiobjetivo

Um problema de otimização multiobjetivo com r objetivos pode ser definido da seguinte forma: dado um vetor de variáveis de decisão com dimensão n, $x = \{x_1, \dots, x_n\}$ no espaço de busca X, queremos encontrar um vetor $x^* \in X$ que minimiza simultaneamente as r funções objetivo $f(x^*) = \{f_1(x^*), \dots, f_r(x^*)\}$. Em geral, X é definido por uma série de restrições e limites de especificação para as variáveis de decisão.

Se todas as funções objetivo são de minimização, uma solução viável x diz-se dominar outra solução viável y se e somente se $f_i(x) \leq f_i(y)$ para $i = 1, \dots, r$ e $f_i(x) < f_j(y)$ para pelo menos uma função objetivo j. Uma solução viável é dita *Pareto-ótima*, ou *não-dominada* ou *eficiente*, se não for dominada por nenhuma outra solução viável no espaço de busca X. Uma solução *Pareto-ótima* não pode ser melhorada com relação a qualquer objetivo sem que exista piora para pelo menos algum outro objetivo.

O conjunto das soluções *não-dominadas* em X é chamado de *conjunto Pareto-ótimo*, e a imagem de um determinado conjunto *Pareto-ótimo* no espaço dos valores dos objetivos é chamada de *fronteira de Pareto* (KONAK, COIT & SMITH, 2006).

4. Método Heurístico Proposto

Neste trabalho, propõe-se um novo método heurístico para resolver o problema de escalonamento *open shop* multiobjetivo. O método tem por base o método de Newton multiobjetivo de Fliege *et al*. (2008) que resolve problemas de otimização multiobjetivo não linear com variáveis contínuas. O método de Newton multiobjetivo tem como característica principal o fato de pertencer à classe dos algoritmos ditos de descida, isto é, cada novo iterado melhora simultaneamente todas as funções objetivo. Uma adaptação deste método foi realizada de modo a considerar problemas com variáveis inteiras, e ainda considerar um novo critério de parada, uma vez que o método base utiliza como teste de parada a satisfação da condição necessária de otimalidade de primeira ordem, ou seja, gradiente da função objetivo nulo em um iterado, que agora não pode ser mais usado, já que a noção de gradiente não se aplica em variáveis inteiras.

O algoritmo iterativo aqui proposto visa melhorar primeiramente a alocação de máquinas e depois a sequência de operações, obtendo por fim um conjunto de soluções *não-dominadas* em relação às medidas de desempenho sendo consideradas simultaneamente.

4.1. Estrutura principal

O algoritmo proposto gera uma solução inicial, ou seja, uma sequência de operações e uma alocação de máquinas (s^*, v^*), que seja factível em relação à precedência das operações. O algoritmo segue iterativamente melhorando as soluções inicialmente geradas. Sejam NS o número de soluções geradas inicialmente, S o conjunto de soluções melhoradas de todas as soluções inicialmente geradas, e ND o conjunto de soluções *não-dominadas* do conjunto S. O pseudocódigo da estrutura principal do algoritmo proposto é descrito a seguir.

Procedimento: Estrutura principal
Input: NS;
Output: ND;
Início
  $S \leftarrow \emptyset, ND \leftarrow \emptyset$;
  para $i \leftarrow 1$ **até** NS **faça**
    Gere uma solução inicial (s^*, v^*);
    melhoria ← verdadeiro;
    enquanto (melhoria = verdadeiro) **faça**
      $v' \leftarrow v^*$;
      Melhore a alocação de máquinas v^*;
      se $(v' = v^*)$ **então**
        melhoria ← falso;
      senão
        Melhore a sequência de operações s^*;
      fim se;
    fim enquanto;
    $S \leftarrow S \cup \{s^*, v^*\}$;
  fim para;
  $ND \leftarrow$ Soluções não dominadas de S;
Fim

Em seguida, são detalhados os procedimentos do algoritmo proposto.

4.2. Melhore a alocação de máquinas

Considerando s^* fixa, o procedimento parte de uma alocação inicial de máquinas $v_0 = v^*$, para $k = 0$. Seu objetivo é melhorar em cada iteração pelo menos uma função objetivo sem piorar a outra função. Para isso, em cada iteração k, gera-se a vizinhança $M(v_k)$ da alocação v_k e, dependendo do valor de θ, escolhe-se o melhor vizinho de $M(v_k)$, de modo que se minimize pelo menos um objetivo. O melhor vizinho de $M(v_k)$ será v_{k+1} e k passa a ser $k + 1$. O procedimento continuará enquanto for possível reduzir pelo menos uma função objetivo (caso em que descida=verdadeiro). Ao término do procedimento a melhor alocação encontrada v_k será atribuída a v^*. A vizinhança $M(v_k)$ é gerada pelo método descrito na seção 2.2.

Neste procedimento, v denota cada vizinho de $v_k \in M(v_k)$ na iteração k; f_1, f_2 e f_3 são as funções objetivo a minimizar; θ mede a variação de f_1, f_2 e f_3 entre a solução corrente (s^*, v_k) e as soluções vizinhas (s^*, v) dado por:

$$\theta = \min_{v \in M(v_k)} \max_{j=1,2,3} (f_j(s^*, v) - f_j(s^*, v_k));$$

Se θ for negativo, significa que é possível reduzir simultaneamente f_1, f_2 e f_3. Se θ for positivo, então, pelo menos uma função objetivo piora. Se θ for zero, então, é possível melhorar no máximo dois objetivos ou simplesmente nenhum objetivo melhora.

O esquema geral do procedimento para melhorar a alocação de máquinas, é apresentado a seguir.

Procedimento: Melhore a alocação de máquinas
Input: s^*, v^*;
Output: v^*;
Início
  $v_0 \leftarrow v^*, k \leftarrow 0$, descida ← verdadeiro;
  enquanto (descida = verdadeiro) **faça**
    determine θ;
    se $(\theta < 0)$ **então**
      $v_{k+1} \leftarrow$ o vizinho com a menor variação máxima de f_1, f_2 e f_3;
    senão se $(\theta = 0)$ **então**
      se (menor valor da variação mínima não positiva de f_1, f_2 e $f_3 = 0$) **então** descida ← falso;
      senão
        Teste as direções de busca f_1, f_2, f_3, f_1 e f_2, f_2e f_3, f_1e f_3;
        Selecione a direção de busca que melhore no máximo dois objetivos;
        $v_{k+1} \leftarrow$ o vizinho com a menor variação máxima não positiva dentre f_1 ou $f_2$ou f_3 ou f_1 e $f_2$ou f_2e $f_3$ou f_1e f_3;
    senão descida ← falso;
    fim se;
    se (descida = verdadeiro) **então** $k \leftarrow k + 1$;
    fim se;
  fim enquanto;
  $v^* \leftarrow v_k$;
Fim

Se o menor valor da variação mínima não positiva de f_1, f_2 e f_3 for zero, não é possível continuar melhorando nenhum objetivo, fazendo com que o procedimento termine. No caso contrário, tem-se a situação em que é possível melhorar um ou dois objetivos sem piorar os demais objetivos.

4.3 Melhore a sequência de operações – procedimento auxiliar

Este procedimento resolve um problema auxiliar biobjetivo, onde o primeiro objetivo f_1 é equivalente ao objetivo original do problema de minimização e o segundo objetivo f_4 é um objetivo auxiliar não conflitante com f_1. Sua ideia é minimizar primeiramente f_1 e depois minimizar f_4 sem piorar f_1.

Considerando v^* fixa, o procedimento parte de uma sequência inicial de operações $s_0 = s^*$, em $k = 0$. Em cada iteração k, gera-se a vizinhança $N(s_k)$ da sequência s_k e, dependendo do valor de θ, escolhe-se o melhor vizinho de $N(s_k)$, de modo que minimize f_1 (neste caso f_4 pode piorar). Quando não for possível melhorar f_1, escolhe-se o melhor vizinho de $N(s_k)$ que minimize f_4 sem piorar f_1. O melhor vizinho de $N(s_k)$ será s_{k+1} e k passa a ser $k + 1$. O procedimento continuará enquanto for possível reduzir f_1 ou reduzir f_4 sem piorar f_1 (caso em que descida=verdadeiro). Finalmente, ao término do procedimento a melhor sequência s_k será atribuída a s^*. A vizinhança $N(s_k)$ é gerada pelo mesmo método descrito na seção 2.3.

Neste procedimento, s denota cada vizinho de $s_k \in N(s_k)$ na iteração k; θ mede a variação de f_1 entre a solução corrente (s_k, v^*) e as soluções vizinhas (s, v^*) dado por:

$$\theta = \min_{s \in N(s_k)} (f_1(s, v^*) - f_1(s_k, v^*))$$

Se θ for negativo, então, é possível reduzir f_1. Se θ for positivo, então f_1 piora. Se θ for zero, então, é possível melhorar f_4 sem piorar f_1 ou simplesmente nenhum objetivo melhora.

O esquema geral do procedimento é descrito a seguir.

Procedimento: Melhore a sequência de operações
Input: s^*, v^*;
Output: s^*;
Início
$s_0 \leftarrow s^*$, $k \leftarrow 0$, descida ← verdadeiro;
enquanto (descida = verdadeiro) **faça**
determine θ;
se $(\theta < 0)$ **então**
$s_{k+1} \leftarrow$ o vizinho com a menor variação de f_1;
senão se $(\theta = 0)$ **então**
se (menor valor da variação não positiva de $f_4 = 0$) **então** descida ← falso;
senão $s_{k+1} \leftarrow$ o vizinho com a menor variação não positiva de f_4;
fim se;
senão descida ← falso;
fim se;
se (descida = verdadeiro) **então** $k \leftarrow k + 1$;
fim se;
fim enquanto;
$s^* \leftarrow s_k$;
Fim

Se o menor valor da variação não positiva de f_4 for zero, não é possível continuar melhorando f_4. No caso contrário, tem-se a situação em que é possível melhorar f_4 sem piorar f_1.

5. Experimentos numéricos

Nesta seção são apresentados resultados preliminares com o algoritmo heurístico proposto para resolver o problema de programação de sondas *open shop* multiobjetivo com instâncias baseadas em dados reias. Os experimentos numéricos foram realizados em MATLAB 6 e executados em um computador com processador de 2,4 GHz e 2 GB de memória RAM.

O algoritmo proposto para resolver o problema *open shop* usou os procedimentos descritos na seção anterior, com as funções objetivo $f_1 =$ *makespan*, $f_2 =$ carga de trabalho máxima e $f_3 =$ carga de trabalho total, assim como o objetivo auxiliar $f_4 =$ tempo total de máquina parada.

As duas instâncias aqui consideradas foram tomadas de Vasconcelos (2007), a primeira instância tem 137 atividades e 4 recursos e a segunda tem 23 atividades e 4 recursos. O método proposto foi aplicado considerando 10 soluções iniciais e executado 10 vezes para cada instância.

Na Tabela 1 e na Tabela 2 são apresentados os resultados obtidos com o método proposto juntamente com os resultados conhecidos de métodos estudados por Vasconcelos (2007) para as duas instâncias: programação por restrições, GRASP, Algoritmo Genético, Branch and bound e MSproject.

Os resultados dos experimentos numéricos indicam que o método proposto conseguiu obter uma boa aproximação das soluções conhecidas, considerando a minimização de três objetivos simultaneamente e partindo de 10 soluções iniciais, quando comparado com os métodos existentes que consideram um único objetivo.

É importante mencionar que experimentos computacionais com o algoritmo proposto em instâncias não relacionadas ao problema de programação de sondas de produção apresentam como resultados soluções ótimas dispersas ao longo da fronteira de Pareto. Nesse problema particular, considerando instâncias baseadas em dados reais, em que o número de atividades é muito maior que o número de recursos, não é possível gerar a fronteira de Pareto aproximadamente quando mais de um objetivo são considerados.

Tabela 1. Resultados obtidos para a instância com 137 atividades e 4 recursos

Método	Makespan	Carga de trabalho máxima	Carga de trabalho total
Programação por Restrições	1088 dias	-	-
GRASP	1092 dias	-	-
Algoritmo Genético	1075 dias	-	-
Branch and Bound	1083 dias	-	-
MSProject	1299 dias	-	-
Método proposto	1077 dias	1070	2399

Tabela 2. Resultados obtidos para a instância com 23 atividades e 4 recursos

Método	Makespan	Carga de trabalho máxima	Carga de trabalho total
Algoritmo Genético	173 dias	-	-
Branch and Bound	173 dias	-	-
Método proposto	174 dias	168	361

6. Conclusões

Neste trabalho, resolvemos o problema de escalonamento *open shop* multiobjetivo com um novo método inspirado no método de Newton multiobjetivo para otimização contínua.

Experimentos numéricos com o método proposto foram realizados em duas instâncias conhecidas para encontrar aproximadamente soluções *não-dominadas* ou eficientes. Bons resultados foram obtidos para essas instâncias, considerando apenas 10 soluções iniciais. De fato, o método proposto conseguiu soluções satisfatórias nestas instâncias com a vantagem de utilizar apenas um único parâmetro, ou seja, o número de soluções iniciais.

Referências

Borchardt, M., *Algoritmos evolucionários na solução do problema da programação de sondas de produção*, Dissertação de mestrado do curso de Sistemas e Computação, UFRN, Natal, 2002.

Fliege, J., Drummond, L. M. G. e Svaiter, B. (2008), Newton's method for multiobjective optimization, *Optimization Online*.

Konak, A., Coit, D. e Smith, A. (2006), Multi-objective optimization using genetic algorithms: a tutorial, *Reliability Engineering & System Safety*, 91, 992-1007.

Liu, D., Yan, P. e Yu, J. (2009), Development of a multiobjective GA for advanced planning and scheduling problem, *The International Journal of Advanced Manufacturing Technology*, 42, 974-992.

Natal, A. C., *Aplicação de programação matemática na racionalização do uso de sondas de perfuração e completação de poços de petróleo off-shore*, Projeto de fim de curso - Engenharia de Produção, UFRJ, Rio de Janeiro, 2003.

Pinedo, M., *Scheduling theory algorithms and system*, New York, Prentice Hall, 2008.

Vanconcellos, R. V. J. C., *Um algoritmo genético para o problema de scheduling de projetos com restrições de recursos – uma solução com gerenciamento de risco*, Tese de doutorado do curso de Engenharia de Produção, UFRJ, Rio de Janeiro, 2007.

Xia, W. e Wu, Z. (2005), An effective hybrid optimization approach for muti-objective flexible job-shop scheduling problem, *Computers & Industrial Engineering*, 48, 409-425.

Zhang, H., Gen, M. e Seo, Y. (2006), An effective coding approach for multiobjective integrated resource selection and operation sequences problem, *Journal of intelligent manufacturing*, 17, 385-397.