ESTUDO DE ALGORITMOS PARA O PROBLEMA DE OTIMIZAÇÃO DE VAZÃO DE POÇOS DE PETRÓLEO

João Olavo Baião de Vasconcelos, Maria Claudia Boeres, Lucia Catabriga
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Av. Fernando Ferrari, 514 - Goiabeiras - 29075-910 - Vitória - ES - Brasil
joaoolavo@gmail.com, boeres@inf.ufes.br, luciac@inf.ufes.br
Henrique Araújo Cotrim
Universidade Estadual de Campinas - UNICAMP
Departamento de Engenharia de Petróleo – Faculdade de Engenharia Mecânica
Rua Mendeleyev, 200 - Cidade Universitária "Zeferino Vaz", Barão Geraldo - 13083-860 - Campinas - SP - Brasil
henrique.cotrim@gmail.com

RESUMO

A indústria do petróleo está rotineiramente envolvida numa série de problemas de otimização em variados contextos, dentre eles está a exploração de reservatórios de petróleo, estes interceptados por poços produtores e injetores. Um dos problemas de otimização nesse contexto é descobrir a melhor distribuição de vazões entre os poços que resulta na maior rentabilidade do projeto, respeitando-se os limites de vazão máxima de cada poço e de todo o sistema de produção. Este problema foi aqui denominado *Problema de Otimização de Vazão de Poços de Petróleo* (POVPP). Para buscar a melhor distribuição das vazões entre os poços, são utilizados neste trabalho dois algoritmos de otimização contínua: o DFO e o APPS, em que o primeiro utiliza um modelo quadrático para substituir a função objetivo e o segundo é executado em processos paralelos e assíncronos. São apresentados também dois reservatórios para estudo de caso que dão diversidade aos testes realizados.

PALAVRAS CHAVES. Otimização contínua. Vazão de poços de petróleo. Algoritmos livre de derivadas e paralelos.

ABSTRACT

The oil industry is routinely involved in a series of optimization problems in various contexts, among them is the exploration of oil reservoirs, these intercepted by prodution and injection wells. In this context, one of the optimization problems is to find the best well throughput distribution that increases the project profitability, respecting maximum throughput limits of each well and the whole production system. This problem was here called *Oil Well Throughput Optimization Problem* (OWTOP). To get the best distribution of throughput among wells, this paper uses two algorithms for continuous optimization: DFO and APPS, where the first uses a quadratic model to replace the objective function and the second runs processes in parallel and asynchronous. Two reservoirs are also presented for case studies that give diversity to the tests.

KEYWORDS. Continuous optimization. Oil well throughput. Derivative-free and parallel algorithms.

1 Introdução

A atividade de Engenharia de Petróleo está rotineiramente envolvida numa série de problemas de otimização em variados contextos. Em todas as etapas da cadeia de trabalho, é possível identificar problemas que necessitam de otimização, em maior ou menor escala, com maior ou menor complexidade. Mais especificamente, a busca por projetos otimizados e eficientes na produção e desenvolvimento de reservas de petróleo é um problema desafiador dentro da indústria de Óleo & Gás, uma vez que envolve um número muito elevado de fatores (técnicos, econômicos e estratégicos) e os seus resultados são bastante compensadores. Além disso, os projetos de Exploração e Produção exigem um comprometimento de capital muito elevado, por um longo período e com elevado risco, dentro de cenários de muita incerteza, reforçando a necessidade de projetos otimizados.

Entretanto, há uma extrema dificuldade na resolução de problemas de otimização em petróleo, uma vez que são problemas frequentemente complexos, com elevado grau de não-linearidade, que apresentam alto grau de incertezas e com enorme custo computacional envolvido (Horne, 2002). Nesse contexto, é necessário gerenciar três elementos principais: o *reservatório*, com foco na forma de drenar os hidrocarbonetos contidos no meio poroso; os *poços*, na decisão sobre a melhor locação para se proceder as perfurações; e os *equipamentos de superfície*, onde a infraestrutura e a operação devem ser ajustadas para maximizar a recuperação e minimizar os custos.

O objetivo desse trabalho é estudar o problema dentro da abrangência do elemento reservatório, no que se refere à Engenharia de Reservatórios, com o foco em descobrir a melhor distribuição de *vazões* entre os poços produtores e injetores capaz de resultar em um projeto de maior rentabilidade. Este problema foi aqui denominado *Problema de Otimização de Vazão de Poços de Petróleo* (POVPP) (Oliveira, 2006).

Para tratar da melhor maneira a distribuição das vazões entre os poços, são utilizados neste trabalho dois algoritmos de otimização contínua: o *Derivative Free Optimization* (DFO) (Conn et al., 1997a) e o *Asynchronous Parallel Pattern Search* (APPS) (Hough et al., 2001). Cada algoritmo possui uma característica vantajosa: o DFO utiliza um modelo quadrático da função objetivo para evitar executá-la; já o APPS lida com diversos processos paralelos e assíncronos para realizar a busca por uma solução mais otimizada. Ambas as características auxiliam na redução do tempo de execução dos algoritmos.

O tratamento dado ao POVPP neste trabalho é similar ao de Oliveira (2006). O autor utiliza quatro algoritmos de otimização para lidar com o problema, e dentre eles está o DFO e o *Pattern Search*, que utiliza uma estratégia semelhante ao APPS. Pelo fato desses dois algoritmos terem sido considerado os melhores, eles foram escolhidos para serem utilizados aqui, sendo que, diferentemente do *Pattern Search*, o APPS foi escolhido por ser paralelizável.

Este texto está organizado como a seguir. A Seção 2 introduz o POVPP, mostrando também as restrições às quais o problema está sujeito. Os algoritmos DFO e APPS são vistos na Seção 3. A Seção 4 apresenta dois estudos de caso do problema em questão, os quais são tratados pelos algoritmos citados, e os resultados obtidos são analisados na Seção 5. A partir das análises realizadas, são apresentadas as conclusões obtidas com este trabalho na Seção 6, bem como sugestões para trabalhos futuros.

2 O Problema de Otimização de Vazão de Poços de Petróleo (POVPP)

Os reservatórios de petróleo são em grande parte dos casos estruturas geológicas complexas que apresentam heterogeneidades significativas nas propriedades petrofísicas de fluxo. Para seu desenvolvimento, eles são interceptados por diversos poços, tanto produtores quanto injetores. Tais reservatórios possuem originalmente uma energia dita primária, em função do volume e da natureza dos fluidos, além das pressões e da temperatura do reservatório. O início da produção dissipa essa

energia original e proporciona a perda de produtividade dos poços, que pode ser mitigada pela aplicação de algum método convencional de recuperação secundária (Rosa et al., 2006).

A injeção de água é a prática mais usual de recuperação secundária e é capaz de garantir a manutenção da pressão do reservatório em níveis desejáveis, além de propiciar o deslocamento dos hidrocarbonetos dos poros das rochas reservatório para os poços produtores. Todavia, uma vez iniciada a injeção de água, inevitavelmente haverá produção de água junto com o óleo em níveis crescentes com o tempo, resultando em queda drástica da eficiência de recuperação. A situação ideal é aquela onde as vazões de produção e injeção são definidas tais que a taxa com que o óleo é varrido seja praticamente uniforme no reservatório, de modo que nenhum poço produtor venha a produzir o fluido injetado antes dos demais (Horne, 2002; Zakirov et al., 1996).

A simulação numérica é um dos métodos empregados na engenharia de petróleo para estimar características e prever o comportamento de um reservatório de petróleo. Neste trabalho, é utilizado o simulador comercial de diferenças finitas *CMG Imex*[1]. O simulador recebe como entrada o modelo de um reservatório junto com as vazões a serem utilizadas nos poços. Dependendo da complexidade do reservatório, do número de poços instalados e do tempo de produção do campo, a simulação pode ser bastante custosa em relação ao tempo de execução. Ao final da simulação, é sabido a quantidade de óleo produzido e de água injetada de acordo com os parâmetros recebidos.

De uma maneira geral, a função objetivo utilizada para otimização da explotação de um campo de petróleo está relacionada à produção propriamente dita ou à economia do projeto, associando-se a algum parâmetro econômico. Neste trabalho, é utilizado exclusivamente o valor presente líquido (VPL) do fluxo de caixa da operação do campo como função objetivo do problema a ser maximizada. O VPL obtido depende então dos valores de produção de óleo e de injeção de água retornados pelo simulador por, respectivamente, onerar e desonerar o resultado financeiro da produção.

As *vazões nos poços* em diferentes tempos ao longo da simulação são as variáveis de controle idealizadas para o problema tratado neste trabalho. Por meio do ajuste da abertura dos *chokes* (válvulas) na superfície, as vazões de produção e injeção podem ser obtidas facilmente. Contudo, tais vazões devem respeitar determinados limites máximos, conforme a Eq. (1).

$$q_{p,t} \leq q_p^u, t = \{1, \ldots, n_t\}, p \in P \cup I, \tag{1}$$

onde $q_{p,t}$ é a vazão do poço p no intervalo de tempo t; q_p^u é o limite superior de vazão do poço p, n_t é o número de discretizações do tempo total de operação; P é o conjunto dos índices dos poços produtores; e I é o conjunto dos índices dos poços injetores.

Os sistemas de produção sempre apresentam algum tipo de restrição de capacidade, visto que, no momento em que tais unidades são projetadas e instaladas, as informações de reservatórios são escassas e repletas de incertezas. Assim, o dimensionamento segue critérios que consideram o risco de superdimensionar as instalações frente às incertezas. As restrições usuais de capacidade estão relacionadas à capacidade de tratamento ou processamento dos fluidos produzidos (óleo, água e líquidos totais) e à capacidade de compressão de gás. Dessa maneira, as Eqs. (2) e (3) representam o conjunto de restrições de igualdade do grupo dos poços produtores e injetores, respectivamente.

$$\sum_{p \in P} q_{p,t} = Q_P, t = \{1, \ldots, n_t\}, \tag{2}$$

onde Q_P é a restrição da capacidade do grupo dos poços produtores.

$$\sum_{p \in I} q_{p,t} = Q_I, t = \{1, \ldots, n_t\}, \tag{3}$$

onde Q_I é a restrição da capacidade do grupo dos poços injetores.

[1] http://www.cmgroup.com/software/imex.htm

As restrições de capacidade são tratadas como de limites a serem atingidos (restrições de igualdade), e não como um limite superior – como nas Eqs. (2) e (3), pois tem-se na indústria a percepção de que tal atitude resulta em uma maior produção do campo.

Dadas então as características do problema a ser tratado, o POVPP é tido como um problema de otimização contínua que visa maximizar uma função objetivo bastante custosa e cujas variáveis estão sujeitas a limites e a restrições lineares de igualdade.

3 Algoritmos utilizados para a resolução do POVPP

Para tratar o POVPP, foram utilizados dois algoritmos cujas implementações estão disponibilizadas para uso: o DFO (Conn et al., 1998) e o APPS (Hough et al., 2001). Ambos são *solvers* para tratar problemas de otimização contínua com restrições lineares e não-lineares cujas funções objetivo são muito custosas. Nesta seção, os algoritmos são apresentados, sendo enfatizado como os elementos do problema de interesse são relacionados com os elementos dos algoritmos.

Em ambos os algoritmos, uma solução do problema é representada por um vetor contendo as vazões a serem aplicadas em cada poço durante cada período de tempo. Para se avaliar o VPL de uma solução, durante a execução das funções objetivo dos algoritmos, é chamado o simulador *CMG Imex*. O resultado informado de produção de óleo e de injeção de água é então analisado para se obter o lucro final.

3.1 Derivative Free Optimization (DFO)

O DFO é um algoritmo de otimização de funções sujeitas a restrições lineares e não lineares e cujas derivadas da função objetivo não estão disponíveis ou são muito difíceis de serem obtidas (Conn et al., 1997a). O algoritmo se baseia na aproximação da função objetivo por um modelo de interpolação polinomial quadrática e usa esse modelo junto com um sistema de região de confiança. Informações sobre a motivação do desenvolvimento do DFO podem ser encontradas em Conn et al. (1997b) e em Powell (1998).

A principal característica do DFO é a aproximação da função objetivo por um modelo quadrático. O modelo é usado para obter valores aproximados da função objetivo, tentando assim prever seu comportamento sem a necessidade de utilizá-la, evitando o alto custo de tempo de execução que a função objetivo possa ter. Tal modelo quadrático é apresentado na Eq. (4).

$$m_k(x_k + s) = f(x_k) + \langle g_k, s\rangle + \frac{1}{2}\langle s, H_k s\rangle \tag{4}$$

onde k é o índice da iteração, x_k (solução), s e g_k são vetores em $\mathbb{R}^n$, H_k é uma matriz hessiana $n \times n$, em que n é a dimensão do problema, e $\langle \cdot, \cdot \rangle$ representa o produto escalar canônico em $\mathbb{R}^n$. Outra característica importante é o sistema de região de confiança, que provê um raio (variável) dentro do qual o modelo quadrático está bem preciso frente à função objetivo. A apresentação do DFO pode ser vista no Algoritmo 1.

A região de confiança β_k está definida na linha 5 do algoritmo: o conjunto de todos os pontos que estão dentro do raio de confiança Δ a partir de x_k. É escolhido, então, o melhor ponto x_k^+ pertencente à região de confiança – de acordo com o modelo quadrático m_k – para ser avaliado. Caso o ponto a ser avaliado seja diferente da melhor solução atual, calcula-se a razão ρ_k (linha 6) para mensurar o quanto a função objetivo será reduzida em relação a quanto o modelo será reduzido com o novo ponto. Em seguida, são executadas avaliações para atualizar o conjunto de interpolação e a solução atual x_k, caso x_k^+ apresente uma melhora na função objetivo (linhas 8 a 11). Por fim, o raio de confiança é aumentado ou diminuído (linha 12) dependendo dos resultados obtidos, parando o algoritmo caso o raio seja muito pequeno. Observa-se que a função objetivo do problema somente é chamada uma ou duas vezes a cada iteração (linha 6). Dado que diversos problemas possuem

Algoritmo 1: DFO

Entrada: Solução inicial x e raio inicial da região de confiança Δ

$k = 0$; $x_k = x$;
Criar conjunto de interpolação Y contendo x_k;
repita
  Construir o modelo $m_k(x)$ usando Y;
  `/* Minimizar o modelo */`
  Encontrar o ponto x_k^+ pertencente à região de confiança, tal que:

$$m_k(x_k^+) = \min_{x \in \beta_k} m_k(x), \beta_k = \{x : ||x - x_k||_\infty \leq \Delta\}$$

  se $x_k^+ \neq x_k$ **então**
    `/* Comparar os valores da função objetivo e do modelo */`

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}$$

  fim se
  `/* Atualizar a interpolação e a solução atual */`
  Adicionar, se possível, x_k^+ ao conjunto de interpolação Y sem remover nenhum outro ponto;
  Se a redução da função objetivo é muito boa quando comparada à redução do modelo ($\rho \to 1$), então incluir x_k^+ em Y, removendo outros pontos caso necessário, e fazer $x_{k+1} = x_k^+$;
  Se a redução não é boa ou nem é atingida, então fazer $x_{k+1} = x_k$;
  Se um ou mais pontos de interpolação estão muito distantes de x_k, removê-los de Y;
  `/* Atualizar a região de confiança */`
  Se a razão ρ_k é boa, aumentar Δ; Se não, diminuir;
  `/* Atualizar a iteração */`
  $k = k + 1$;
até Δ *ser muito pequeno* (condição de parada) ;

funções objetivo muito custosas de serem calculadas, isso dá ao DFO uma grande vantagem no que tange o tempo de execução.

A implementação do DFO utilizada está disponível no site do grupo *Computational Infrastructure for Operations Research*[2] (COIN-OR) (Conn et al. (1998)). Para a execução do código, deve-se preparar duas rotinas na linguagem Fortran 77 (Scheinberg (2003)). Na primeira, que será o programa a ser executado, são definidas diversas variáveis de entrada, dentre as quais estão representadas a solução inicial (vazões dos poços a cada período de tempo), dois vetores com os limites das variáveis e as restrições lineares (vazão máxima de cada poço e restrições de igualdade dos grupos de poços injetores e produtores) e os valores inicial e limite mínimo do raio da região de confiança (Δ). Dentro dessa mesma rotina, por fim, é chamada a função principal do algoritmo, a *DFO*, que recebe todos os parâmetros iniciais. A segunda rotina desenvolvida é chamada de *FUN*, que é exatamente a função objetivo do problema. Ela recebe um vetor com uma solução viável, executa a simulação do reservatório com as vazões recebidas, calcula o VPL obtido e retorna o valor da função objetivo para o algoritmo.

3.2 Asynchronous Parallel Pattern Search (APPS)

Os algoritmos de busca por padrões (*pattern search*) são algoritmos de otimização voltados para resolver problemas cujas derivadas da função objetivo não estão disponíveis ou são difíceis de serem obtidas. Eles foram descritos por Hooke & Jeeves (1961) como sendo compostos de uma

[2]https://projects.coin-or.org/Dfo

iteração com duas etapas, a serem descritas a seguir.

A primeira etapa realiza movimentos exploratórios para adquirir conhecimento do comportamento da função objetivo, inferido a partir das melhoras ou pioras que houver, independente dos valores da função em si. Com as informações de melhora ou piora, obtém-se padrões (*patterns*), que indicam a probabilidade de sucesso de um movimento. Já a segunda etapa, chamada de *pattern move*, utiliza as informações adquiridas nos movimentos exploratórios para seguir na direção do padrão obtido, a fim de otimizar a função objetivo. Desse modo, cada *pattern move* é seguido de uma sequência de movimentos exploratórios que continuamente reveem o padrão que está direcionando para o caminho da solução ótima. A ideia intuitiva desse tipo de movimento é a presunção de que movimentos que obtiveram sucesso em otimizar a função objetivo tenderão a um novo sucesso.

O APPS é um algoritmo de busca por padrões para resolver problemas de otimização com restrições lineares e não-lineares (Hough et al., 2001) e permite que a exploração por melhores resultados aconteça em várias frentes paralelas, cada uma de maneira assíncrona com a outra. Desse modo, cada frente de busca não necessita esperar que as demais terminem uma iteração para só então continuar, evitando que o processador fique ocioso (*idle*).

Quando um movimento (processo) realiza a avaliação de um ponto, o algoritmo verifica qual é o melhor resultado obtido até o momento pelos demais processos e compara-o com o que foi encontrado, independente dos outros movimentos estarem ou não na mesma iteração – o APPS não é indexado baseado em iterações, mas sim na noção de passo de tempo (*time step*). Assim, aproveita-se da possibilidade explícita de paralelismo do algoritmo para que o atraso na avaliação de um ponto não resulte na estagnação das demais frentes de busca.

A paralelização concentra-se na estratégia de busca do algoritmo, não na avaliação da função objetivo em si. Desse modo, o APPS é executado utilizando um conjunto *P* de *n* processos. Cada processo $i \in P$ inicia a busca a partir do ponto x_i^0 (solução inicial) e torna-se responsável por explorar uma direção no espaço de busca. Do ponto de vista de um único processo *i*, os passos do APPS estão descritos no Algoritmo 2.

Algoritmo 2: APPS (do ponto de vista do processo i)

Entrada: Solução inicial x_0, vetor direção d_i e tamanho do passo inicial Δ

1 $x_i^0 = x_0$;
2 $\Delta_i^0 = \Delta$;
3 **repita**
4 | Avaliar $f(x_i^t + \Delta_i^t d_i)$;
5 | **se** $f(x_i^t + \Delta_i^t d_i) > f(x_i^t)$ **então**
6 | | Notificar o resultado encontrado para os demais processos;
7 | **fim se**
8 | Receber os resultados encontrados pelos demais processos;
9 | Obter os valores de x_i^{t+1} e Δ_i^{t+1} baseado nos resultados obtidos;
10 **até** *condições de parada atingidas* ;

Dado que o processo i está no passo de tempo t e posicionado no ponto x_i^t, na linha 4 do algoritmo, o processo analisa seu próximo passo, deslocando-se Δ_i^t na direção d_i. Se houve melhora em relação ao ponto que estava, os demais processos são notificados de seu sucesso (linha 6), mas também são considerados os sucessos obtidos pelos demais processos (linha 8). Com tais dados, a próxima solução e tamanho do passo são determinados de acordo com as Eqs. (5) e (6), que estão descritas a seguir.

Para efeito de entendimento de como os valores de x_i^t e de Δ_i^t são atualizados, é necessário introduzir alguns conjuntos utilizados (Kolda & Torczon (2003)). T é o conjunto dos passos de tempo de execução e é particionado como $T = S_i \bigcup U_i$, sendo S_i o conjunto de todos os passos de tempo t em que se obteve sucesso no processo i (x_i^t foi atualizado) e U_i o conjunto complementar, em que não houve sucesso. Também é definido o conjunto $C_i \subseteq U_i$ dos passos de tempo em que

ocorre contração no processo i, ou seja, em que Δ_i^t diminui, pois a avaliação da função terminou e nenhum resultado melhor foi encontrado. Além disso, com o objetivo de acompanhar o processo e o tempo que deu origem a um novo ponto otimizado, são apresentadas as seguintes funções:

- $\omega_i(t) =$ o índice do processo que atualizou o processo i no tempo t.
- $\tau_i(t) =$ o passo de tempo de quando se iniciou a avaliação da função que atualizou o processo i no tempo t.
- $\nu_i(t) =$ o passo de tempo de quando se completou a avaliação da função que atualizou o processo i no tempo t.

Dessa maneira, para todo $t \in T$, $t > 0$, x_i^t é formalmente definido de acordo com a Eq. (5).

$$x_i^t = \begin{cases} x_{\omega_i(t)}^{\tau_i(t)} + \Delta_{\omega_i(t)}^{\tau_i(t)} d_{\omega_i(t)}, & \text{se } t \in S_i \\ x_i^{t-1}, & \text{caso contrário} \end{cases} \tag{5}$$

Também fazendo uso das funções anteriores, Δ_i^t é atualizado seguindo a regra da Eq. (6).

$$\Delta_i^t = \begin{cases} \lambda_{\omega_i(t)}^{\nu_i(t)} \Delta_{\omega_i(t)}^{\tau_i(t)}, & \text{se } t \in S_i \\ \theta_i^t \Delta_i^{t-1}, & \text{se } t \in C_i \\ \Delta_i^{t-1}, & \text{caso contrário} \end{cases} \tag{6}$$

Os parâmetros θ_i^t e λ_i^t são dados de entrada do algoritmo. Tipicamente, θ_i^t assume o valor $\frac{1}{2}$ e λ_i^t recebe o valor 2, se a mesma direção otimizar a função objetivo duas ou mais vezes consecutivas, ou 1, caso contrário (Kolda & Torczon, 2003).

O entendimento básico do APPS pode então ser sumarizado como um processo que irá sempre se direcionar para o ponto a partir do qual é sabido que se obteve o melhor resultado até o momento, independente de quem o encontrou, mas seguindo sempre na direção que lhe cabe, caminhando mais rápido ou devagar de acordo com a obtenção ou não de sucesso no último passo de tempo.

A implementação utilizada do APPS foi a APPSPACK[3] (Gray & Kolda, 2006), que recebe os parâmetros de entrada a partir de um arquivo XML. Nesse arquivo, foram especificados o limite superior de vazão de cada poço, as restrições lineares de igualdade para os grupos produtores e injetores de poços, o executável com a função objetivo, a solução inicial selecionada e a tolerância do tamanho do passo Δ_{tol} (critério de parada), que limita o valor do tamanho do passo de um processo. Interessante ressaltar que, somente quando Δ_i^t é menor que Δ_{tol}, é que o processo $i \in P$ fica ocioso no processador, pois o critério de parada foi atingido, e ficará assim até que alguma nova solução otimizada seja encontrada ou até que todos os demais Δ_j^t, $j \in P$, sejam inferiores à tolerância do tamanho do passo.

4 Estudos de caso

Para a análise dos algoritmos DFO e APPS quando aplicados ao POVPP, foram utilizados dois estudos de caso. Cada estudo de caso representa um modelo de reservatório de petróleo, com poços injetores e produtores, além das restrições dos poços e do sistema de produção. Ambos os reservatórios são artificiais e não representam nenhum modelo real existente. O primeiro estudo de caso, denominado *Caso 1*, é um modelo bem simples, com um reservatório pequeno e poucos poços. Já o segundo estudo de caso, denominado *Caso 2*, apresenta um modelo mais complexo, com mais poços e um sistema com três reservatórios isolados entre si. Os dois estudos de caso são melhor detalhados a seguir.

[3] https://software.sandia.gov/appspack/version5.0

4.1 Caso 1

A fim de estudar um caso simples, em que seja possível realizar mais rapidamente testes de parametrização e análises do comportamento dos algoritmos, foi desenvolvido o modelo de reservatório do Caso 1. A ilustração do modelo pode ser observada na Fig. 1, que apresenta três poços produtores (*P1*, *P2* e *P3*) e um poço injetor (*I1*).

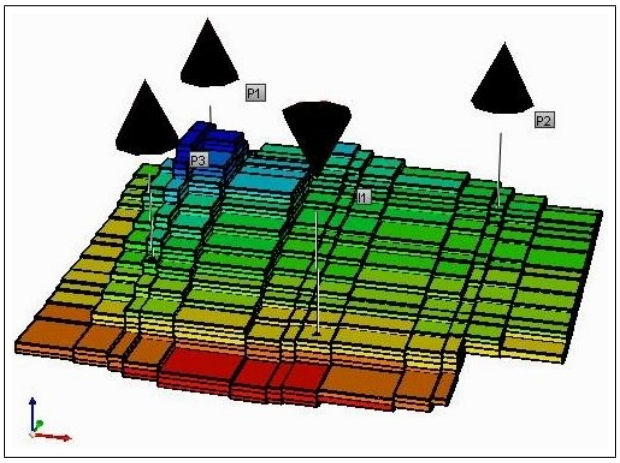

Figura 1: Modelo de reservatório do Caso 1.

A Tab. 1 apresenta os limites de vazão dos poços do Caso 1, em que cada poço produtor pode escoar no máximo 30 m^3/dia de fluido, enquanto que o único poço injetor tem uma vazão máxima de 44 m^3/dia. O sistema de produção limita a vazão do grupo de poços produtores em 40 m^3/dia, enquanto que o injetor pode atuar com sua vazão máxima, já que essa também é o limite de injeção do sistema de produção.

Tabela 1: Limites máximos de vazões dos poços para o Caso 1.

Poço	Vazão máxima (m^3/dia)	Limite do grupo (m^3/dia)
P1	30	
P2	30	40
P3	30	
I1	44	44

As simulações são realizadas em um período de 15 anos, com mudança nas vazões a cada cinco anos. Cada simulação do Caso 1 feita pelo *Imex* leva em torno de 2 segundos. Considerando os quatro poços e os três períodos temporais, tem-se que a otimização do Caso 1 é um problema de 12 variáveis.

4.2 Caso 2

O Caso 2 é caracterizado por ser constituído de três reservatórios independentes, cada um com um conjunto de quatro poços produtores e um poço injetor, dispostos no formato conhecido como *five spot*. Apesar dos reservatórios serem independentes entre si, todos os poços estão instalados no mesmo sistema de produção. A Fig. 2 ilustra este estudo de caso, onde podem ser observados o posicionamento de todos os poços.

As restrições de vazão dos 12 poços produtores e dos três poços injetores podem ser vistas na Tab. 2. O grupo de poços produtores tem um limite máximo de vazão de 4.000 m^3/dia, enquanto que o limite do grupo dos injetores é de 4.500 m^3/dia.

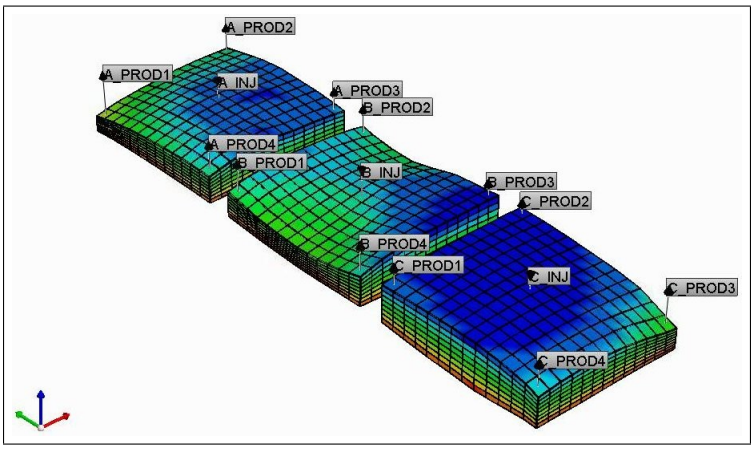

Figura 2: Modelo de reservatório do Caso 2.

Tabela 2: Limites máximos de vazões dos poços para o Caso 2.

Poço	Vazão máxima (m^3/dia)	Limite do grupo (m^3/dia)
A_PROD1	400	4.000
A_PROD2	400	
A_PROD3	400	
A_PROD4	400	
B_PROD1	400	
B_PROD2	400	
B_PROD3	400	
B_PROD4	400	
C_PROD1	400	
C_PROD2	400	
C_PROD3	400	
C_PROD4	400	
A_INJ	2.000	4.500
B_INJ	2.000	
C_INJ	2.000	

O Caso 2 considera a produção do campo por um período de 12 anos, com a troca das vazões dos poços a cada quatro anos. Assim, o problema de otimização do Caso 2 possui um vetor solução com 45 variáveis. As simulações do Caso 2 feitas pelo *Imex* levam em torno de 12 segundos.

5 Resultados computacionais

Nesta seção, são mostrados os resultados numéricos obtidos por cada algoritmo, analisando a otimização da solução inicial do problema, o tempo de execução do algoritmo e a quantidade de vezes que a função objetivo foi chamada.

A execução do APPS foi feita com 16 processos rodando em computadores distintos – e em um único *core* por processador, conectados em uma rede Gigabit, enquanto que o DFO, por ser serial, foi executado em uma única máquina. Os computadores possuem processador *Intel Xeon E5440* de 2,83 GHz e 16 GB de memória RAM.

Como dados de entrada, foram definidas, para cada estudo de caso, três instâncias do POVPP, cada uma com uma solução inicial viável e aleatória, utilizadas para obtenção dos resultados aqui apresentados. A Tab. 3 mostra o VPL das soluções iniciais e finais obtida pelos algoritmos APPS e DFO, em cada uma das instâncias. Os resultados são mostrados em ordem decrescente da evolução

do valor da função objetivo em relação ao da solução inicial e separados por estudo de caso.

Tabela 3: VPLs obtidos pelos algoritmos DFO e APPS, para cada instância selecionada, e a respectiva evolução percentual da função objetivo.

Algoritmo	Instância	VPL da solução inicial ($)	VPL da solução final ($)	Evolução da função objetivo (%)
CASO 1				
APPS	CASO1_RND1	60.321.293,6	65.300.940,6	8,3
APPS	CASO1_RND2	61.276.905,6	65.421.167,1	6,8
DFO	CASO1_RND2	61.276.905,6	65.360.580,0	6,7
APPS	CASO1_RND3	64.036.790,8	65.362.126,0	2,1
DFO	CASO1_RND1	60.321.293,6	612.17.570,0	1,5
DFO	CASO1_RND3	64.036.790,8	64.036.790,0	0,0
CASO 2				
DFO	CASO2_RND3	456.537.999,7	922.174.300,0	102,0
DFO	CASO2_RND2	721.153.372,9	971.030.800,0	34,6
DFO	CASO2_RND1	653.861.891,7	733.838.700,0	12,2
APPS	CASO2_RND3	456.537.999,7	500.775.009,0	9,7
APPS	CASO2_RND1	653.861.891,7	658.683.686,0	0,7
APPS	CASO2_RND2	721.153.372,9	722.331.360,0	0,2

É notável que, nas instâncias do Caso 2, o DFO apresenta resultados muito superiores que os obtidos pelo APPS. Como exemplo, o DFO conseguiu dobrar o VPL da instância *CASO2_RND3*, enquanto que o APPS não chegou a aumentar em 10%. Entretanto, a situação se inverte no Caso 1: os resultados do APPS são todos superiores em relação aos do DFO, ressaltando-se ainda que, na instância *CASO1_RND3*, o DFO não conseguiu obter uma solução muito diferente que a inicial.

Para se avaliar o tempo de execução de cada algoritmo, pode-se observar a Fig. 3. Nela são mostrados, para cada uma das três instâncias de cada estudo de caso, o tempo em segundos que os algoritmos APPS e DFO levaram para encontrar a solução final.

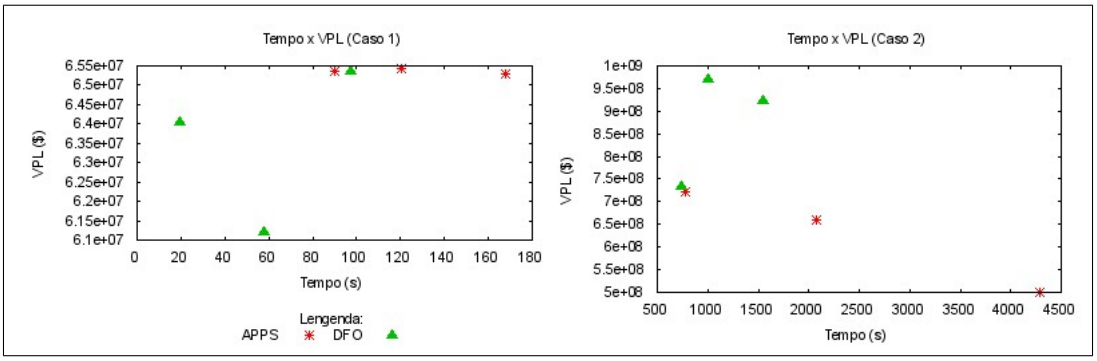

Figura 3: Tempo de execução de cada algoritmo para as instâncias dos casos 1 (esquerda) e 2 (direita) em relação ao VPL da solução final encontrada.

Nas instâncias do Caso 1, observa-se, como visto na Tab. 3, que o APPS apresenta os melhores resultados. Contudo, o tempo para obtenção das soluções finais pelo DFO é menor. Já no Caso 2, o DFO apresenta as melhores soluções nos menores tempos. Ressalta-se aqui o fato do DFO ter obtido tais resultados utilizando apenas um processador, enquanto que a execução paralela do APPS fez uso de 16 processadores.

Por fim, é também interessante analisar a quantidade de vezes que a função objetivo é chamada por cada algoritmo. Como o POVPP tem como característica um alto custo da avaliação da função objetivo, o número de vezes que a função objetivo necessita ser chamada implica diretamente no

tempo de execução do algoritmo. A Fig. 4 apresenta uma comparação entre o número de vezes que o APPS executa a função objetivo em relação ao DFO.

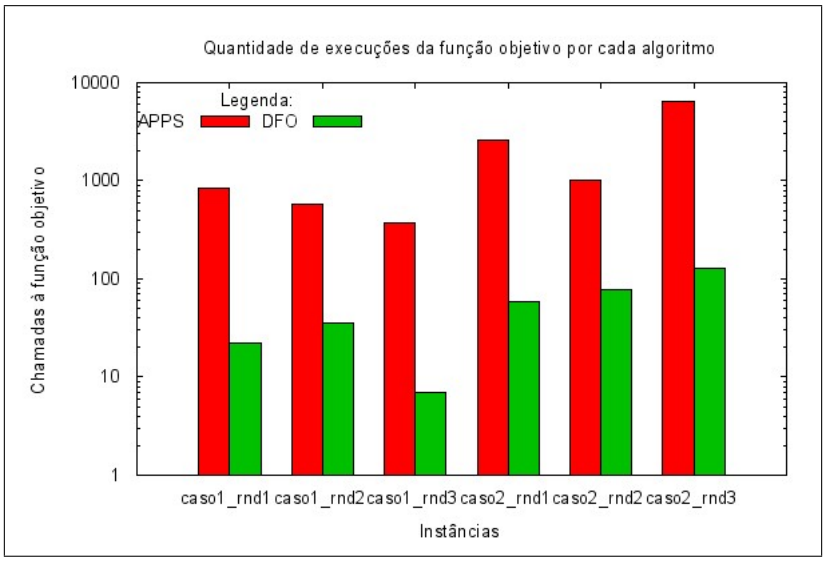

Figura 4: Número de vezes que cada algoritmo executa a função objetivo para as três instâncias de cada estudo de caso (escala logarítmica).

A Fig. 4 pode ser utilizada para corroborar os resultados vistos na Fig. 3: como o APPS executa muito mais a função objetivo que o DFO, o tempo final do primeiro algoritmo tende a ser maior, ainda que tenha sido utilizado processamento paralelo.

6 Conclusões e Trabalhos Futuros

O POVPP é um grande desafio do ponto de vista matemático, físico e de engenharia. Sua relevância é vista através dos resultados apresentados, em que se conseguiu ganhos expressivos no VPL. No entanto, é importante aqui enfatizar que não se pode afirmar que tais ganhos serão auferidos em cenários reais da indústria do petróleo.

Analisando os resultados, os algoritmos DFO e APPS apresentaram características diferentes quando tratando as instâncias dos dois estudos de caso. Enquanto que o APPS encontrou melhores soluções no Caso 1, no Caso 2 a situação se inverteu, com o DFO obtendo os melhores resultados.

Analisando-se os tempos de execução dos algoritmos, o DFO encontrou os resultados em tempo menor nos dois estudos de caso. Tal resultado deve-se à modelagem da função objetivo que o DFO incorpora em seu código. Isso permite com que, enquanto o APPS executa a função objetivo por diversas vezes em cada iteração, o DFO utilize a função modelo para analisar as soluções candidatas, somente chamando a função objetivo uma ou duas vez por iteração.

Observando também que o Caso 1 possui 12 variáveis e que o Caso 2 possui 45, pode-se concluir que o desempenho do APPS decai com um número maior de variáveis no problema, enquanto que o DFO obteve os melhores percentuais de melhora no VPL quando lidando com mais variáveis. Além disso, o DFO fez uso de somente um computador, enquanto que os resultados do APPS foram obtidos utilizando 16 máquinas.

Após essas análises, pode-se concluir que o desempenho do algoritmo DFO no POVPP foi superior ao do APPS para os casos estudados.

Este trabalho pode ser complementado analisando outros algoritmos de otimização (*e.g., Algoritmo Genético*) que tratem restrições lineares. O acréscimo de mais estudos de caso – com diferentes quantidades de variáveis – também pode ser válido para se ter uma comparação mais

aprofundada entre os algoritmos. Pode-se pensar ainda na utilização de soluções iniciais melhor estruturadas para o problema – por exemplo, considerando soluções em que haja necessariamente a reposição de volumes produzidos, visto que tal abordagem é premissa adotada rotineiramente por engenheiros de reservatórios, principalmente no Caso 2, onde há compartimentos porosos isolados. Por fim, a análise do *speed up* dos algoritmos paralelos utilizados é bastante interessante para avaliar a quantidade de nós necessários para se obter resultados do tempo de execução semelhantes ao DFO.

Referências

Conn, A. R., Scheinberg, K., & Toint, P. L. (1997a). On the convergence of derivative-free methods for uncontrained optimization. *Approximation Theory and Optimization: Tributes do M. J. D. Powell*, pages 83–108.

Conn, A. R., Scheinberg, K., & Toint, P. L. (1997b). Recent progress in unconstrained nonlinear optimization without derivatives. *Math. Program*, 79:397–414.

Conn, A. R., Scheinberg, K., & Toint, P. L. (1998). A derivative free optimization algorithm in practice. *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*.

Gray, G. A. & Kolda, T. G. (2006). Algorithm 856: APPSPACK 4.0: asynchronous parallel pattern search for derivative-free optimization. *ACM Trans. Math. Softw.*, 32(3):485–507.

Hooke, R. & Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229.

Horne, R. N. (2002). Optimization application in oil and gas recovery. In *Handbook of Applied Optimization*, New York. Oxford University Press.

Hough, P. D., Kolda, T. G., & Torczon, V. J. (2001). Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1):134–156.

Kolda, T. G. & Torczon, V. J. (2003). Understanding asynchronous parallel pattern search. In *High Performance Algorithms and Software for Nonlinear Optimization*, pages 316–335. G. Di Pillo and Murli, Editors.

Oliveira, D. F. B. (2006). Técnicas de otimização da produção para reservatórios de petróleo – abordagens sem uso de derivadas para alocação dinâmica das vazões de produção e injeção.

Powell, M. J. D. (1998). Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336.

Rosa, A. J., Carvalho, R. S., & Xavier, J. A. D. (2006). *Engenharia de Reservatórios de Petróleo*. Editora Inerciência, Rio de Janeiro.

Scheinberg, K. (2003). Manual for fortran software package dfo v2.0. Technical report.

Zakirov, I. S., Aanonsen, S. I., Zakirov, E. S., & Palatnik, B. (1996). Optimizing reservoir performance by automatic allocation of well rates. In *5th European Conference on the Mathematics of Oil Recovery*.