ALGORITMO DE EVOLUÇÃO COM DIFERENCIAÇÃO NUMÉRICA PARA AJUSTE DE ESTRUTURAS DE SENSORES VIRTUAIS EM UM PROCESSO PETROQUÍMICO

GUSTAVO A. P. DE MORAIS, BRUNO H. G. BARBOSA, HÉLIO R. S. JUNIOR

DEPARTAMENTO DE ENGENHARIA, UNIVERSIDADE FEDERAL DE LAVRAS
LAVRAS, MG, BRASIL
E-MAILS: GUSTAVO_GAPM@HOTMAIL.COM, BRUNOHB@DEG.UFLA.BR, HELIORICARDO27@HOTMAIL.COM

Abstract — In this paper we present a new evolutionary algorithm model for solution of multi modal optimization problems, called Evolution Algorithm with Numerical Differentiation. The proposed method is based on increasing the search radius within the acceptable limits of the domain of the function while the best solutions are being found. Also, the algorithm proposes a dynamic parameterization of evolutionary algorithms, based on the principles of numerical differentiation, ensuring a good balance between local and global search. In this work, the proposed algorithm is compared to other four models already established (Differential Evolution Algorithm, Modified Differential Evolution Algorithm, RST2ANU and MI-LXPM). The comparison is made in two simulated problems, one of maximization and the other one of minimization, and a real problem about parameter estimation of an Artificial Neural Network model for use in virtual sensors of a petrochemical process. The results show that the proposed algorithm has superior performance to other methods in the studied examples.

Keywords — Evolutionary Algorithm, Numerical Differentiation, Functions Optimization.

Resumo — Neste trabalho é apresentado um novo modelo de algoritmo evolucionário para solução de problemas de otimização multi modais, chamado Algoritmo de Evolução com Diferenciação Numérica. O método proposto se baseia em aumentar o raio de busca nos limites aceitáveis do domínio da função à medida que melhores pontos são encontrados. O algoritmo também propõe uma forma dinâmica de parametrização de algoritmos evolucionários, baseada nos princípios de diferenciação numérica, garantindo um bom equilíbrio entre busca local e busca global. Neste trabalho, o algoritmo proposto é comparado com outros quatro modelos já consagrados (Algoritmo de Evolução Diferencial, Algoritmo de Evolução Diferencial Modificado, RST2ANU e MI-LXPM). A comparação é feita em dois problemas simulados, sendo um de máximo global e o outro de mínimo global, e em um problema real de estimação de parâmetros de um modelo baseado em Redes Neurais Artificiais para aplicação em sensores virtuais de um processo petroquímico. Os resultados mostram que o algoritmo proposto possui desempenho superior aos outros métodos nos exemplos estudados.

Palavras-chave — Algoritmo Evolucionário, Diferenciação Numérica, Otimização de Funções.

1 Introdução

Otimização de sistemas lineares e não lineares é, de fato, um desafio para problemas de matemática e engenharia. É comum encontrar situações que podem ser melhoradas ou otimizadas em indústrias, distribuição de água e gás, administração, design, automobilismo, aviação, petróleo, etc. Na tentativa de resolver esses problemas a computação evolucionária vem desenvolvendo trabalhos de grande importância na área de identificação de sistemas e otimização, visto que é difícil encontrar um modelo matemático que represente um processo em intervalos de tempo longos com precisão. Essa dificuldade normalmente surge do fato de que os processos reais são de natureza não-linear e variantes no tempo.

Várias técnicas de computação clássica foram apresentadas visando resolver esses problemas. Nas duas últimas décadas, em particular, foram desenvolvidos vários modelos de algoritmos estocásticos, e eles se mostraram eficientes para otimização de problemas não-convexos (Deep, Singh, Kansal e Mohan, 2009). Em especial, destacam-se técnicas de computação evolucionária, como Algoritmos Genéticos (AGs) e Algoritmos Evolucionários (AEs). Esses algoritmos se baseiam no princípio da seleção natural proposto por Charles Darwin (Goldberg, 1989), com herança genética, mutação e seleção dos indivíduos mais aptos de acordo com os parâmetros de uma função. Os AGs e os AEs utilizam informações de todo o domínio de busca disponível para o problema. O que torna esses métodos não limitados apenas aos valores locais, e sim a todo o espaço de busca.

Para problemas de otimização, uma ferramenta computacional bastante utilizada é o Algoritmo de Evolução Diferencial (AED). Esse método tem um princípio de aplicação simples e bastante eficiente: poucos parâmetros são controlados, normalmente determinados de forma aleatória, de forma a encontrar o ponto de melhor aptidão, usando os princípios de algoritmos evolucionários (Mohan, 1999).

Uma variação do AED é o Algoritmo de Evolução Diferencial Modificado (AEDM), com a particularidade de fazer com que as posições das partículas sejam ajustadas de forma adaptativa. Isso faz com que os novos conjuntos de dados formados se desenvolvam na melhor direção. Assim, AEDM promove uma maior diversidade da população em relação ao AED mas possui uma convergência mais lenta (Deep, Singh, Kansal e Mohan, 2009).

Outros algoritmos foram desenvolvidos e aprimorados à medida que aumentava a necessidade por melhores resultados com menores custos computacionais, ou seja, menos tempo de operação e processamento. Como o RST2, que é um algoritmo que usa técnicas de aproximação quadrática e sua adaptação para técnicas de buscas aleatórias controladas, RST2ANU (Mohan, 1999). Outro algoritmo que se

mostra eficiente é o algoritmo MI-LXPM (*Mixed Integer - Laplace Crossover and Power Mutation*), que é uma extensão do algoritmo LXPM. Nesse caso, tem-se em especial a utilização do *Crossover* de Laplace e *Power Mutation*, ambos definidos em detalhes por (Deep e Thakur, 2007), para a resolução de problemas de otimização.

Visando desenvolver uma técnica de fácil implementação computacional, com capacidade de tratar problemas de otimização não-convexos, e que tenha capacidade de evolução adaptativa com bom equilíbrio entre busca local e busca global, neste trabalho é apresentado o Algoritmo de Evolução com Diferenciação Numérica. O objetivo não é apenas apresentar mais um algoritmo de otimização, mas também mostrar o benefício do uso da diferenciação numérica dentro do algoritmo evolutivo. Além disso, o algoritmo foi proposto com o intuito de auxiliar na busca por melhores estruturas de modelos dinâmicos a serem empregados em um processo petroquímico, descrito a seguir.

2 Descrição do problema

O trabalho foi iniciado como uma proposta de otimização para os processos de extração de petróleo em alto-mar. Para perfurar poços nessas condições, na maioria das vezes, é necessário o uso de meios artificiais para que o conteúdo do poço flua nos ductos de extração até o reservatório. Vários métodos podem ser usados, mas o método que foi dado atenção no problema foi o chamado *Gás-Lift* Contínuo. Esse é o método mais popular e consiste em reduzir a densidade da coluna de óleo injetando gás sob alta pressão nos ductos, variando, assim, a pressão de fluxo de óleo no fundo dos poços (Thomas, 2001).

A maior dificuldade nessa operação está em determinar o regime ideal de injeção de gás. Isso não é um processo fácil, visto que a relação produção/injeção de gás possui rendimento decrescente devido a compressão e ao custo do gás usado, e também devido a presença de não-hidrocarbonetos no óleo que está sendo elevado (Jadid et al., 2007).

Para auxiliar os operadores em decidir qual o regime ideal de injeção de gás, é utilizado o sensor PDG (*Permanent Downhole Gauge*), que fornece as informações sobre a pressão de fundo de poço. Essas informações são essenciais para determinar o regime ideal de injeção de gás na coluna de óleo. No entanto, o sensor PDG está localizada em uma região inóspita, sob alta pressão e salubridade. Isso faz com que o sensor sofra grande desgaste, diminuindo seu tempo de utilidade. Quando o sensor para de funcionar a equipe é obrigada a trabalhar às cegas, visto que a troca do sensor PDG só pode ser realizada se a operação for parada.

De fato, os sensores ainda estão sujeitos a muitos problemas, tais como atrasos de medidas, desempenho ruim sob certas condições, interferência no processo, preços de compra e/ou manutenção, etc. Para diminuir esses problemas, criou-se os chamados "sistemas inteligentes", que são instrumentos que manipulam computacionalmente as informações medidas para que elas sejam transmitidas da melhor forma possível (Latufo e Garcia, 2008). Dentre esses instrumentos, é destacado o uso de *Soft Sensors*, que são sensores resultantes da união entre sensores inteligentes e técnicas de modelagem e identificação de sistemas.

Dentre as várias propostas apresentadas para corrigir essa deficiência na troca de sensores PDG, foi dado destaque a utilização de *Soft Sensors*, via Redes Neurais Artificiais. No entanto, um grande número de atrasos possíveis poderia ser implementado no modelo neural. Optou-se então por usar Algoritmos Evolucionários para determinar qual conjunto de atrasos promoveria uma melhor resposta do modelo neural. O modelo neural utilizado no trabalho é uma máquina de comitê, formada por 10 RNAs (Redes Neurais Artificiais) do tipo MLP *feedforward* com 10 neurônios na camada escondida. A Figura 1 abaixo representa esse modelo, com os atrasos indicados por variáveis auxiliares (a, b, c e d), que são números inteiros que podem variar entre 2 e 500.

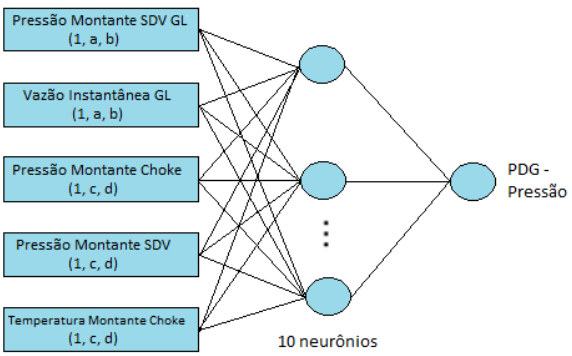

Figura 1. Descrição do modelo neural. Fonte: (Gomes, 2014).

O Algoritmo de Evolução com Diferenciação Numérica foi criado com o objetivo de otimizar esse modelo neural para que o mesmo modelo seja usado em outros poços, com outras características. Cada poço tem um fluxo de petróleo e pressão natural de poço própria. Isso faz com que cada variável contenha uma informação importante para o processo, portanto os valores que são usados para as variáveis de um conjunto de Redes Neurais Artificiais devem ser diferentes para as variáveis que usamos para um segundo conjunto.

Essa técnica foi adaptada sucessivas vezes até que foi desenvolvido o Algoritmo de Evolução com Diferenciação Numérica (AEDN), de fácil utilização, baixo custo computacional e grande eficiência em determinar o conjunto de atrasos em uma faixa intervalo que permitia a criação de um número intratável de combinações.

3 Algoritmo de Evolução com Diferenciação Numérica

Em um contexto geral, o Algoritmo de Evolução com Diferenciação Numérica (AEDN) aumenta o raio de busca por melhores pontos de aptidão à medida que o conjunto de indivíduos se torna homogêneo. Ou seja, quanto menor for a diferença entre os valores de aptidão fornecidos pela função em estudo, maior o raio de busca do algoritmo. Essa ideia tem como essência evitar que os valores convirjam para um ponto que não seja o máximo ou mínimo global da função, evitando assim uma possível convergência prematura do algoritmo.

Outra característica é o fato do algoritmo ter três processamentos paralelos. No primeiro, tem-se o processamento principal, onde ocorre a mutação e a seleção que serão descritas nas seções a seguir. A cada nova iteração, o processamento principal armazena os 5 melhores valores em uma matriz. Esse processo é atualizado a *loop*. O segundo processo paralelo ocorre quando 40% das interações são alcançadas, ele tem como base processar os 5 melhores pontos obtidos no processamento principal a fim de que seja encontrado o melhor ponto nas proximidades desses valores. O terceiro processamento paralelo ocorre quando é atingido 70% das interações, ele repete o mesmo procedimento do processamento anterior, trabalhando com os 5 valores que o processamento principal encontrou nesse momento.

3.1 Diferenciação Numérica

Diferenciação Numérica é uma técnica analítica usada para encontrar a derivada de uma curva sem conhecer a função que a define. Isso é muito útil quando estão sendo trabalhados dados amostrais, tais como a saída de um sensor. Muitas vezes a derivada de um sinal pode deixar mais claro o que está acontecendo com o processo.

Para este trabalho, foi considerada a fórmula dos cinco pontos, que é uma função que indica a derivada de um sinal amostral em um dado ponto em função dos dois pontos anteriores e dois pontos posteriores a ele. Assim, pode-se determinar a derivada de todo o conjunto de pontos medidos. Essa função está representada na equação:

$$\frac{\mathrm{dx}(i)}{\mathrm{dt}} = \frac{1}{100}[-2\,\mathrm{x}(i-2) - \mathrm{x}(i-1) + \mathrm{x}(i+1) + 2\,\mathrm{x}(i+2)] \quad (1)$$

Para o caso em estudo, quando é calculada a derivada de *x* na primeira posição, são usados como pontos anteriores a *x(1)* os dois últimos pontos do vetor *x*. Assim como, quando é calculada a derivada de *x* na última posição, os dois pontos posteriores são os dois primeiros pontos do vetor *x*, no caso, *x(1)* e *x(2)*.

3.2 Técnicas de seleção

Algoritmos evolucionários normalmente fazem a seleção dos indivíduos copiando os princípios da seleção dos indivíduos mais aptos da teoria da evolução das espécies. Indivíduos "pais" geram seus descendentes, então mede-se o valor de aptidão entre o indivíduo "pai" e o indivíduo "filho", caso o descendente seja mais apto, permuta-se sua localização com o indivíduo pai, gerando assim uma população mais "adaptada" para o problema em questão.

Para o caso do Algoritmo de Evolução com Diferenciação Numérica, os descendentes não são comparados com seus pais em um primeiro momento. Em vez da comparação ser realizado entre pai e filho, são unidos todos os indivíduos em uma mesma população ao final do processo de mutação e reprodução, tendo assim uma população com o dobro do tamanho da população inicial. Em seguida é medido o valor de aptidão de cada um e são selecionados metade dos indivíduos, formada pelos seus melhores valores.

3.3 Passos computacionais para o algoritmo

1. É criado uma população inicial com valores aleatórios dentro do intervalo de busca. Esses valores são armazenados em uma matriz X, e em uma segunda matriz Y são armazenados os valores de aptidão de cada linha da matriz X.
2. É conferido o critério de parada (se o número de interações chegou ao limite). Caso já tenha atingido esse critério, o algoritmo é finalizado. Caso não tenha atingido, vai para o passo 3.
3. É calculada a derivada de cada linha da matriz Y através da fórmula dos cinco pontos, e esses valores são armazenados na matriz Y'.
4. É calculado o fator α através da equação:

$$\alpha = \Delta Limite * aux * 10 \quad (2)$$

Sendo que aux é o decremento unitário, que ocorre a cada interação, do número inicial de interações até o valor 1, aux = (número de interações):-1:1. E a variação do limite é a diferença entre o ponto máximo e mínimo que o algoritmo pode interagir no limite de busca, ΔLimite = (limite superior) - (limite inferior)

O fator α é usado na equação de mutação, ele contribui para que a matriz Y' não convirja para zero quando os dados da matriz X sejam valores muito próximos, assim, esse processo garante o aumento do raio de busca

à medida que os valores de X são otimizados.

5. É calculada, então, a função de mutação para cada linha da matriz X, utilizando os vizinhos inferior e superior à linha em questão e sua respectiva derivada. Os valores de X após a mutação são armazenados na matriz M. Esse procedimento está descrito na equação 3 abaixo, considerando que a matriz X tenha *i* linhas:

$$\begin{aligned} &M(i,:) = X(i,:) + \\ &X(i-1,:)*[1-Y'(i)] + \\ &X(i+1)*[Y'(i)-1] + \\ &\frac{1}{Y'(i)*\alpha+\alpha}*X(i,:) \end{aligned} \quad (3)$$

6. Os valores da matriz M e X são unidos em uma matriz auxiliar, e são selecionados os *i* indivíduos com melhores valores de aptidão, e esses valores compõem a nova matriz X.

7. São selecionados os 5 melhores valores de X. Esses valores são armazenados em uma matriz auxiliar Z.

8. É verificado se a iteração atual representa 40% do número de interações totais. Se sim, inicia-se o processamento paralelo 2.

O processamento paralelo 2 utiliza os valores armazenados na matriz Z. Inicialmente, é calculado a derivada dos pontos de aptidão da matriz Z e os valores são armazenados na matriz Ya'(i). Em seguida, é realizada a mutação e os valores são armazenados na matriz Ma, seguindo a equação:

$$\begin{aligned} &Ma(i,:) = Z(i,:) + \\ &Z(i-1,:)*[1-Ya'(i)] + \\ &Z(i+1)*[Ya'(i)-1] \end{aligned} \quad (4)$$

Nota-se que essa nova mutação não aumenta o raio de busca do algoritmo, isso é feito apenas no processamento principal. Isso porque a ideia do processamento paralelo 2 é encontrar o ponto de aptidão próximos aos pontos fornecidos pela matriz Z, desenvolvendo uma busca local.

São unidas as matrizes Ma e Z, e os 5 melhores valores são escolhidos. Em seguida, a matriz Z recebe esses valores otimizados e o processo se repete na próxima itreração.

9. É verificado se a iteração atual representa 70% do número de interações totais. Se sim, inicia-se o processamento paralelo 3.

O processo ocorre exatamente como acontece no passo 8, com os valores da matriz Z atualizados pelo processamento principal. E os valores mutados são armazenados na matriz Mb.

10. É comparado os melhores pontos nos três processamentos, e é escolhido o melhor valor para o problema. Vai para o passo 2.

4 Comparação entre os algoritmos

Nessa etapa foi utilizado o MATLAB, que é um software que utiliza linguagem de alto-nível, popularmente usado por cientistas e engenheiros para computação numérica, visualização e desenvolvimento de aplicativos e para modelagem de problemas de engenharia

Para verificar a eficiência do algoritmo, foram comparados os resultados obtidos pelo Algoritmo de Evolução com Diferenciação Numérica, com os obtidos pelos Algoritmos de Evolução Diferencial, de Evolução Diferencial Modificado, RST2ANU e MI-LXPM. O objetivo é analisar o desempenho de cada um para o mesmo problema. Para isso foram considerados dois problemas simulados distintos, descritos nas seções *4.1* e *4.2*.

4.1 Problema 1

Para o primeiro problema, o objetivo é encontrar o ponto de máximo global da função descrita na equação:

$$\begin{aligned} &f(x,y) = [f(x,y)-1]*[2\frac{\sin(0{,}1*x)}{3\pi} + \\ &3\cos(0{,}02*x) - 4\sin(0{,}9*x) + \\ &5\sin(0{,}8x) + 2\frac{\sin(0{,}1*y)}{3\pi} + \\ &3\cos(0{,}02*y) - \\ &4\sin(0{,}9*y) + 5\sin(0{,}8y)] \end{aligned} \quad (5)$$

Essa função se mostra muito útil para o este caso, pois ela é composta por vários picos de máximo e mínimos locais. Também foram considerados os limites -50 e 50 nos eixos X e Y para que se tenha um vasto número de pontos e combinações possíveis, exigindo, assim, mais dos algoritmos. O gráfico da função é apresentado na figura 2.

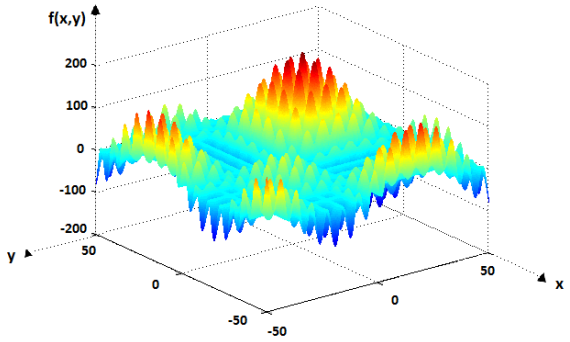

Figura 2. Gráfico da função representado pela equação 1 em 3 dimensões.

4.2 Problema 2

Para o segundo problema, o objetivo é encontrar o mínimo global da função Rastrigin, que assim como o caso anterior, tem duas variáveis independentes no eixo XY, e uma respectiva imagem no eixo Z. E também possui vários mínimos e máximos locais, e um mínimo global, localizado no ponto (0,0). A função de Rastrigin (figura 3) é descrita por:

$$f(x,y) = 20 + x^2 + y^2 - 10(\cos 2\pi x + \cos 2\pi y) \quad (6)$$

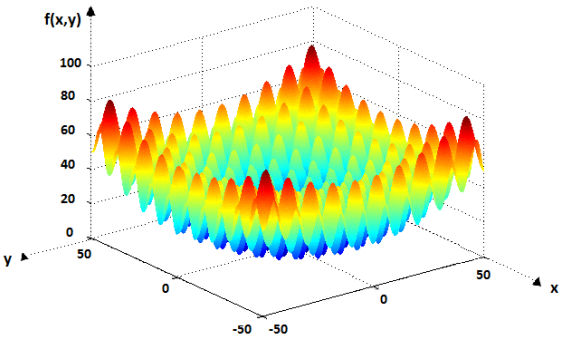

Figura 3. Gráfico da função de Rastrigin em 3 dimensões.

5 Resultados

5.1 Problema simulados

Para os dois problemas simulados, o algoritmo foi iniciado como um vetor composto por 20 linhas (indivíduos) e 2 colunas, cada linha representando um ponto no plano XY, com quatro casas decimais. E cada linha da matriz possui um correspondente valor de aptidão no eixo Z igual a imagem da função no ponto (x,y). Assim, com poucas gerações, população pequena e em duas situações opostas, o primeiro problema com o objetivo de encontrar o máximo global, enquanto o segundo problema com o objetivo de encontrar o mínimo global, foi observado qual algoritmo tem uma melhor convergência.

Cada algoritmo foi testado 100 vezes. Para cada teste, foi utilizado valores diferentes e aleatórios, composto por uma matriz inicial X com 20 linhas e 2 colunas, cada linha composta por um ponto no plano XY ϵ [-50,50].

Para comparar os modelos, foi utilizado o software MATLAB, versão R2011b.

A figura 6, apresenta a comparação entre os métos para o problema 1. e a figura 7 apresenta a comparação para o problema simulado 2. Como pode ser observado, o algoritmo proposto, AEDN, apresentou melhores resultados que todos os outros nos dois problemas estudados, tornando-o apto para aplicação no problema real.

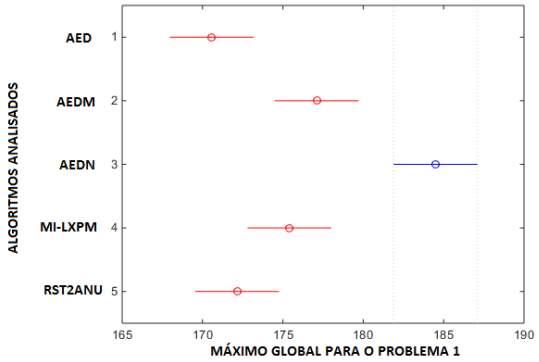

Figura 6. Teste de Tukey (95% de confiança) para os 100 resultados obtidos no problema 1 por cada algoritmo.

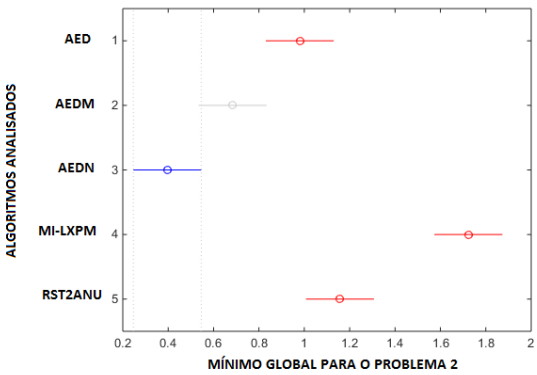

Figura 7. Teste de Tukey (95% de confiança) para os 100 resultados obtidos no problema 2 por cada algoritmo.

5.2 Problema real

Encontrar o conjunto de atrasos do modelo neural que simulasse a pressão do sensor PDG, de fato, representa um grande desafio, visto que havia um grande volume de dados e muitas combinações de atrasos possíveis. Testar todas as entradas possíveis às RNAs era uma atividade inviável, pois exigia muito tempo e custo computacional. Assim, criou-se o AEDN, com o objetivo de ser um modelo que tornasse esse processo mais simples, dinâmico e eficiente.

Como exemplo, é mostrado na figura 8 o valor de pressão PDG medido (em azul) de um poço A junto ao valor de pressão predito pelo modelo neural (em vermelho) cujos atrasos foram ajustados pelo AEDN, sendo o erro quadrático médio (EQM) da

simulação em dados de validação igual a 0,5494 $(kgf/cm^2)^2$. Os atrasos encontrados para o modelo neural desse poço foram [130, 26, 485, 128].

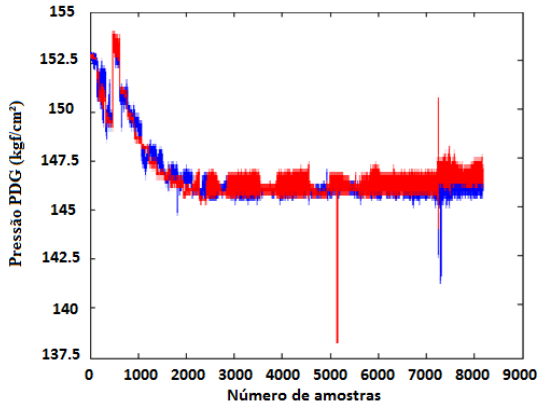

Figura 8. Comparação entre o modelo real (em azul) e o modelo simulado (em vermelho).

Utilizando os atrasos obtidos em um outro poço, para os dados do poço A, atrasos [42, 136, 5, 22], conforme utilizado no trabalho (Barbosa et al, 2015), o erro médio quadrático foi de 0,7397 $(kgf/cm^2)^2$. O que mostra que o uso de um algoritmo evolucionário para estimação dos atrasos em cada poço é justificável. E o AEDN obteve os atrasos que forneciam os melhores valores de erro, sendo superior ao AED, AEDM, RST2ANU e MI-LXPM para essa aplicação. Outra vantagem do AEDN é que o tempo que o algoritmo gasta para encontrar os atrasos do modelo é muito próximo dos outros algoritmos, como o Algoritmo de Evolução Diferencial e o de Evolução Diferencial Modificado. Todos eles levam em média de 30 a 35 minutos para concluir a tarefa em um computador. Enquanto que o MI-LXPM e o RST2ANU levam em média de 60 a 70.

6 Conclusão

Um dos grandes problemas quando é usado algoritmos evolucionários é determinar quais parâmetros serão usados no algoritmo, visto que esses parâmetros mudam a eficiência do algoritmo dependendo do problema em teste. A grande vantagem do AEDN é que a parametrização é atualizada para cada novo processo de mutação em função dos valores encontrados através da diferenciação numérica.

Após essa vantagem ser percebida, foi possível concluir dos resultados que o Algoritmo de Evolução com Diferenciação Numérica teve uma melhor convergência para dois problemas simulados do que outros quatro algoritmos testados (Algoritmo de Evolução Diferencial, Algoritmo de Evolução Diferencial Modificado, MI-LXPM e RST2ANU). Ao aplicar o AEDN no problema de otimização de estrutura de soft-sensor, foi constatado que é possível sua utilização em problemas com grandes espaços de busca, encontrando resultados satisfatórios. E também o algoritmo se mostrou o melhor método para o ajuste de sensores virtuais no processo de extração de petróleo descrito neste trabalho.

Agradecimentos

Os autores agradecem à agência brasileira FAPEMIG pelo apoio financeiro dado a este trabalho.

Referências Bibliográficas

Barbosa, B.H.G.; Gomes, L. P. ; Teixeira, A. F. ; Aguirre, L. A. . Downhole Pressure Estimation Using Committee Machines and Neural Networks. In: IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, 2015, Florianópolis. Proceedings of the 2nd IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, 2015. v. 1.

Deep, k., Krishna, P. S., Kansal, M. L. and Mohan, C (2009). A real coded genetic algorithm for solving integer and mixed optimization problems. Applied Mathematics and Computation, No. 212; pp. 505-518.

Deep, k. e Thakur, M (2007). A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, No. 188; pp. 895-912.

Deep, k. e Thakur, M (2007). A new mutation operator for real coded genetic algorithms. Applied Mathematics and Computation, No. 193; pp. 353-361.

Goldberg, D. E (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Boston: Addison-Wesley Longman Publishing Co.

Gomes, L. P. Utilização de Inteligência Computacional e Técnicas Avançadas de Identificação de Sistemas em Processos de Extração de Petróleo em Alto-Mar, 2014. 109 p. Monografia (Conclusão de curso de Engenharia de Controle e Automação) – Universidade Federal de Lavras, Lavras, 2014.

Jadid, M. B. et al. The Presssure's On: Innovations in Gas Lift. Oilfield Review, [s.i.], Winter 2007. 44-52.

Latufo, F. A. e Garcia, A. Sensores Virtuais ou Soft Sensors: Uma Introdução. 7th Brazilian Conference on Dynamics, Control and Applications, Presidente Prudente, Maio 07-09, 2008.

Mohan, C (1999). Controlled Random Search Technique Incorporating the Simulated Annealing Concept for Solving Integer and Mixed Integer Global Optimization Problems. Applied Mathematics and Computation, No. 14; pp. 103-132.

THOMAS, J. E. Fundamentos de engenharia do petróleo. Rio de Janeiro: Interciência, 2001.