Comparação entre os algoritmos Luus-Jaakola e Particle Swarm Optimization (PSO) em dois problemas de otimização global.

Carlos A. Souza Lima Jr., Jardel S. Costa, Wagner F. Sacco

Universidade do Estado do Rio de Janeiro (UERJ)
IPRJ / PGMC - Instituto Politécnico
Grupo de Termodinâmica e Otimização
Rua Alberto Rangel s/n, Vila Nova, Nova Friburgo
28630-050 Rio de Janeiro, Brasil.
E-mail:{souzalima_ca, jscosta, wfsacco}@iprj.uerj.br

Resumo: *Apresentamos uma comparação entre os algoritmos de otimização global contínua Luus-Jaakola (LJ) e Particle Swarm Optimization (PSO). Inicialmente aplicamos as técnicas a uma função teste conhecida como Exponential Fit de quatro dimensões que é considerada pela literatura e em seguida a um problema de equilíbrio químico que é caracteristicamente um problema de 10 dimensões, mas que pode ser reduzido a um problema de 5 dimensões assim como é tratado. Então verificamos a performance dos algoritmos nos problemas em questão. Os resultados são comparados com a melhor solução conhecida na literatura e o desempenho dos métodos utilizados é analisado no que diz respeito à convergência e consistência nesta aplicação.*

1. Introdução

Problemas de otimização sempre são frequentes em várias áreas do conhecimento na busca de melhores adequações de métodos e/ou processos empregados.

Muitos destes problemas demonstram ter um alto grau de dificuldade para a maioria das técnicas de otimização, o que por anos os deixou sem solução aparente. A partir da segunda metade do século XX técnicas de otimização baseadas em processos aleatórios e guiadas por alguma heurística, conhecidas como técnicas estocásticas, começaram a ser desenvolvidas e a ganhar destaque ao obterem algum êxito na solução de problemas que até então encontravam-se sem solução. A partir de então, diversas outras técnicas de otimização baseadas nos mesmos conceitos, porém com outras características (guiadas por outras heurísticas) foram desenvolvidas e puderam ser implementadas cada vez com maior eficiência para os mesmos problemas dado também pelo avanço da capacidade de processamento dos computadores.

Utilizamos os algoritmos Luus-Jaakola (LJ), que é pouco conhecido, e Particle Swarm Optimization (PSO) na resolução de um problema de ajuste de parâmetros e outro de equilíbrio químico, conhecidos da literatura e de fácil implementação, com o intuito de avaliar comparativamente pela primeira vez o desempenho desses algoritmos na resolução de tais problemas. Trata-se de uma abordagem relevante por capacitar um grupo de pesquisa em termodinâmica no uso de algoritmos sofisticados para resolver problemas de programação global, não linear e contínua.

Na seção 2 é apresentado o problema de equilíbrio químico e na seguinte um problema de ajuste de parâmetros. A seção 4 trata das técnicas de otimização empregadas e a seção 5 mostra os resultados obtidos, finalmente algumas conclusões são expostas na seção 6.

2. O problema do Equilíbrio Químico

Como problema de equilíbrio químico consideramos a abordagem dada por [4] sobre a combustão do propano (C_3H_8) no ar (O_2 e N_2) (1).

$$C_3H_8 + \frac{R}{2}(O_2 + 4N_2) \rightarrow \text{Produtos} \quad (1)$$

Temos então o seguinte sistema não linear de equações cujas incógnitas n_i representam os números de moles do produto *i* formado por mole de propano consumido.

$$\begin{array}{ll}
\textit{Carbono total:} & \textit{Equilíbrio:} \\
f_1 = n_1 + n_4 - 3 = 0 & f_5 = K_5 n_2 n_4 - n_1 n_5 = 0 \\
\textit{Oxigênio total:} & f_6 = K_6 n_2^{1/2} n_4^{1/2} - n_1^{1/2} n_6 N^{1/2} = 0 \\
f_2 = 2n_1 + n_2 + n_4 + n_7 + n_8 + n_9 + 2n_{10} - R = 0 & f_7 = K_7 n_1^{1/2} n_2^{1/2} - n_4^{1/2} n_7 N^{1/2} = 0 \\
\textit{Hidrogênio total:} & f_8 = K_8 n_1 - n_4 n_8 N = 0 \\
f_3 = 2n_2 + 2n_5 + n_6 + n_7 - 8 = 0 & f_9 = K_9 n_1 n_3^{1/2} - n_4 n_9 N^{1/2} = 0 \\
\textit{Nitrogênio total:} & f_{10} = K_{10} n_1^2 - n_4^2 n_{10} N = 0 \\
f_4 = 2n_3 + n_9 - 4R = 0 &
\end{array} \quad (2)$$

onde $N = \frac{p}{\sum_{i=1}^{10} n_i}$, R é um parâmetro físico, K_i é uma constante de equilíbrio e p é a pressão em atmosferas.

Através de substituições algébricas em (2) é possível obter-se um sistema de 5 equações (3) nas variáveis y_1, y_2, y_3, y_4 e y_5 que estão relacionadas com n_i.

$$\begin{aligned}
F_1 &= y_1 y_2 + y_1 - 3 y_5 = 0 \\
F_2 &= 2 y_1 y_2 + y_1 + 2 R_{10} y_2^2 + y_2 y_3^2 + R_7 y_2 y_3 + R_9 y_2 y_4 + R_8 y_2 - R y_5 = 0 \\
F_3 &= 2 y_2 y_3^2 + R_7 y_2 y_3 + 2 R_5 y_3^2 + R_6 y_3 - R_8 y_2 - 8 y_5 = 0 \\
F_4 &= R_9 y_2 y_4 + 2 y_4^2 + 4 R y_5 = 0 \\
F_5 &= y_1 y_2 + y_1 + R_{10} y_2^2 + y_2 y_3^2 + R_7 y_2 y_3 + R_9 y_2 y_4 + R_8 y_2 + R_5 y_3^2 + R_6 y_3 + y_4^2 - 1 = 0
\end{aligned} \quad (3)$$

onde R_i é um parâmetro físico relacionado, exatamente como demonstrado em [4].

Para o problema de otimização foi utilizada a função objetivo (4) composta a partir do sistema de equações que descreve o problema de equilíbrio químico. Ao minimizá-la, o seu valor ótimo deverá ser zero se o sistema tiver solução. São consideradas as restrições de domínio aceitáveis para o número de moles e a solução para o problema obtida em [4] é mostrada na Tabela 3.

$$F(y_1, y_2, y_3, y_4, y_5) = F_1^2 + F_2^2 + F_3^2 + F_4^2 + F_5^2 \quad (4)$$

3. O problema de ajuste de parâmetros

Utilizamos uma função conhecida como Exponential Fit [5] como função objetivo, com a intenção de avaliar o desempenho dos métodos de otimização implementados. Tal função foi gerada para simular o ajuste de informações de um problema da medicina que descreve a concentração de uma droga no sangue ao longo do tempo, segue-se a equação (5).

$$f(x) = \sum_{i=1}^{45} \left(y_i - \left(x_3 . e^{x_1 . t_i} + x_4 . e^{x_2 . t_i} \right) \right) \quad (5)$$

onde $t_i = 0{,}02i$ e os termos i e y_i são dados na tabela a seguir:

i	y_i	i	y_i	i	y_i	i	y_i	i	y_i
1	0,090542	10	0,336995	19	0,283987	28	0,183892	37	0,128926
2	0,124569	11	0,348371	20	0,262078	29	0,152285	38	0,119273
3	0,179367	12	0,321337	21	0,281593	30	0,174028	39	0,115997
4	0,195654	13	0,299423	22	0,267531	31	0,150874	40	0,105831
5	0,2699707	14	0,338972	23	0,218926	32	0,126220	41	0,075261
6	0,286027	15	0304763	24	0,225572	33	0,126266	42	0,068387
7	0,289892	16	0,288903	25	0,200594	34	0,106384	43	0,090823
8	0,317475	17	0,300820	26	0,197375	35	0,118923	44	0,085205
9	0,308191	18	0,303974	27	0,182440	36	0,091868	45	0,067203

Onde $minf\left(\vec{X}\right) \approx 5{,}00x10^{-3}$ em $\vec{X} \approx [-4{,}00;-5{,}00;4{,}00;-4{,}00]^T$ ou $\vec{X} \approx [-5{,}00;-4{,}00;-4{,}00;4{,}00]^T$ nos limites das coordenadas $[-100{,}00;100{,}00]^4$.

4. Algoritmos de Otimização

São algoritmos que buscam encontrar uma configuração de variáveis ou informações que sejam a melhor solução de um problema. A cada configuração analisada é designado um "prêmio", mais conhecido como "*fitness*", que quantifica a adequação daquela configuração àquele problema, sendo a *fitness* nos problemas mais simples o valor de uma função real, chamada função objetivo. Espera-se que a solução do problema seja a configuração com a melhor *fitness* encontrada.

Uma solução possível é frequentemente representada por um vetor $\vec{X} = (x_0, x_1, ..., x_n)$, onde os termos x_i são suas componentes, e o número n é o número de variáveis ou dimensões do problema em questão. Sendo assim *fitness* = $f\left(\vec{X}\right)$, onde $f\left(\vec{X}\right)$ representa uma avaliação que possa quantificar a adequação de $\vec{X}$, e a solução ótima global do problema é dada por $\vec{X}^*$ de modo que $\vec{X}^* = \arg f\left(\vec{X}^*\right)$, onde $f\left(\vec{X}^*\right)$ é a melhor *fitness* encontrada pelo algoritmo de otimização. O algoritmo será de maximização se considerar a melhor *fitness* como sendo o maior número $f\left(\vec{X}\right)$ e será de minimização quando a melhor *fitness* for o menor $f\left(\vec{X}\right)$.

Os algoritmos clássicos de otimização, particularmente os determinísticos, frequentemente necessitam dos valores das derivadas da função objetivo para efetuarem a busca pelo ótimo, ou seja, a função deve ser diferenciável. Há uma classe de algoritmos que não usam as derivadas da função objetivo, são os métodos de busca direta, mais versáteis portanto já que podem enfrentar problemas que não podem ser expressos por uma função diferenciável ou mesmo problemas em que não é possivel representar o contexto como uma função matemática. Dentre

esses existem os métodos estocásticos em que a busca pelo ótimo é guiada por processos aleatórios.

Dois métodos estocásticos de otimização conhecidos como método de Luus-Jaakola e Particle Swarm Optimization (PSO) (otimização por enxame de partículas) implementados em linguagem de programação C++ são utilizados neste trabalho.

Segundo o "*No Free lunch theorem*" enunciado por Wolpert & Macready [7], dado um algoritmo *a* executado *m* vezes em um problema *p*, o desempenho desse algoritmo é definido como $P(dm^y / p, m, a)$, isto é, a probabilidade condicional de encontrar um resultado dm^y. Assim o somatório de tais probabilidades para todos os problemas de otimização é sempre igual para todos os algoritmos. Portanto justifica-se a aplicação de diferentes algoritmos a um mesmo problema, pois um algoritmo altamente especializado em uma classe de problemas poderá apresentar desempenho inferior em outra classe.

4.1 Algoritmo de Luus-Jaakola (LJ)

Proposto por Luus, R. e Jaakola, T. H. I. [3] em 1973 para otimização contínua quando foi largamente utilizado na resolução de problemas da Engenharia Química.

Sua idéia central é: considera-se uma região ampla que englobe os possíveis valores das variáveis, valores factíveis, e geram-se soluções aleatórias enquanto a região de busca torna-se menor ao longo das iterações. A Figura 1 exibe um pseudocódigo.

Escolha um espaço de busca inicial $r^{(0)}$, um número de loops externos n_{out} e um número de loops internos n_{in}.
Escolha um coeficiente de contração ε.
Gere aleatoriamente uma solução inicial X^*
For i = 1 to n_{out}
  For j = 1 to n_{in}
    $X^{(j)} = X^* + R^{(j)} r^{(i-1)}$, onde $R^{(j)}$ é uma matriz diagonal de números aleatórios entre -0.5 e 0.5.
    If Fitness ($X^{(j)}$) < Fitness (X^*)
      $X^* = X^{(j)}$
    End if
  End for
  $r^{(i)} = (1-\varepsilon) r^{(i-1)}$
End for

Figura 1: Luus-Jaakola pseudocódigo

4.2 Particle Swarm Optimization (PSO)

Otimização por enxame de partículas (PSO) é uma técnica de otimização desenvolvida em 1995 por James Kennedy e Russel C. Eberhart em [1] como método de otimização de funções contínuas e não-lineares e tem seus fundamentos em duas principais metodologias: A-life (artificial life) e Swarm theory (teoria de enxames), que simulam o comportamento social dos animais.

Otimização por enxame de partículas (PSO) é um algoritmo de otimização inspirado no comportamento biológico de enxames e aspectos de adaptação social. Os modelos tradicionais de técnicas de otimização têm sua força de concentração na competição (Competição Darwiniana), o algoritmo de PSO escolheu a colaboração como sua estratégia de desenvolvimento.

No PSO as partículas constituem soluções candidatas do sistema a ser otimizado, a cada partícula i em uma dada iteração t do algoritmo, estão associados um vetor posição $X_i(t)$ e um vetor velocidade $V_i(t)$, ambos com n componentes. Sendo o sistema responsável pelas avaliações das partículas durante o processo de busca, determinando topologicamente o comportamento do enxame de partículas. O sistema em questão ao realizar a tarefa de gerar uma avaliação (fitness) para as partículas tem comportamento análogo ao de uma função, e por isso é comumente chamando de função objetivo ou função custo.

As posições das partículas e velocidades são conduzidas por suas próprias experiências tanto quanto a observação de suas vizinhanças bem-sucedidas.

As partículas se movem no espaço de busca segundo as equações (6) e (7)

$$v_{i,j}(t+1) = w.v_{i,j}(t) + c_1.r_1\left(pbest_{i,j}(t) - x_{i,j}(t)\right) + c_2.r_2\left(gbest_j(t) - x_{i,j}(t)\right) \tag{6}$$

$$\begin{aligned} x_{i,j}(t+1) &= x_{i,j}(t) + v_{i,j}(t+1) \\ j &= 1,...,n \qquad i = 1,...,m \end{aligned} \tag{7}$$

onde m é o número de partículas, r_1 e r_2 são números aleatórios entre 0 e 1. Os coeficientes c_1 e c_2 são as dadas constantes acelerações (freqüentemente chamadas de acelerações cognitiva e social) em direção ao pbest e gbest respectivamente [1]. Onde pbest é um vetor que contém as coordenadas da melhor posição já visitada por uma determinada partícula e gbest é um vetor que contém as coordenadas da melhor posição já visitada (entre as posições já visitadas) por todas as partículas. As iterações contabilizam o "passo evolutivo" do enxame de partículas.

No algoritmo PSO, o enxame é inicializado aleatoriamente (posições e velocidades). Então, enquanto o critério de parada não é alcançado, um laço contendo os seguintes passos se segue:
i) Partículas são avaliadas de acordo com o problema da função objetivo, e os valores da fitness são assinalados para cada partícula;
ii) Valores pbest e gbest são atualizados;
iii) A posição e velocidade das partículas são atualizadas de acordo com as equações para velocidade e posição (Eqs. 6 e 7).

```
Inicia PSO
    Inicia enxame de partículas
        Avalia partículas e atualiza gBest
            Inicia iterações
                Move partículas
                Avalia partículas
                Atualiza pbest
                Atualiza gbest
                Enquanto (iterações)
            Fim iterações
Fim PSO
```

Figura 2: PSO pseudocódigo

5. Resultados Obtidos

Foram aplicadas as técnicas de otimização de Luus-Jaakola e Particle Swarm Optimization (PSO) para a função Exponential Fit, num total de 150000 avaliações da função objetivo para cada experimento e considerou-se os parâmentros de configuração exibidos na Tabela 1.

Realizou-se 30 experimentos para o algoritmo de Luus-Jaakola com a melhor configuração encontrada e 30 para o PSO para cada uma das três configurações com 10, 20 e 40 partículas. Os resultados são obtidos na Tabela 2.

Finalmente, aplicamos as técnicas de otimização ao problema de equilíbrio químico reduzido para 5 dimensões [4] utilizando a técnica de otimização de Luus-Jaakola com os melhores parâmetros de configuração encontrados (n_{out}) = 30000; (n_{in}) = 5; (ε) 0,001 e a técnica de otimização por enxame de partículas (PSO) com as configurações exibidas na Tabela 1, porém com população de 20 partículas. Em ambos os casos foram feitos 30 experimentos. Os resultados encontram-se na Tabela 3. Nas Tabelas 2 e 3 os valores de x representam a melhor solução obtida com os 30 experimentos e $f(x)$ o melhor valor, $\overline{f(x)}$ é a média dos 30 melhores valores de função objetivo. Na última linha temos o desvio σ.

Luus-Jaakola (LJ)	PSO
Loop externo (n_{out}) 75000 Loop interno (n_{in}) 2 Coeficiente de contração (ε) 0.0001	Número de partículas 10, 20, 40 Coef. de aprendizado individual (c_1) 2,00 Coef. de aprendizado coletivo (c_2) 3,00 Inércia máxima das partículas $(\omega_{\max})$ 0,8 Inércia mínima das partículas $(\omega_{\min})$ 0,2 Veloc. máxima das partículas $(V_{\max})$ 2,0

Tabela 1: Parâmetros utilizados nos algoritmos Luus-Jaakola e PSO.

Compo-nente	Solução[5]	LJ	PSO		
	x*		10, 20 e 40 partículas		
x_1	-4,00	-4,013500	-4,999997	-4,000025	-4,000047
x_2	-5,00	-4,981890	-4,000001	-4,999967	-4,999937
x_3	4,00	4,128590	-4,000013	4,000230	4,000435
x_4	-4,00	-4,128510	-4,000013	-4,000229	-4,000434
$f(x)$	$5{,}00 \times 10^{-3}$	$4{,}99998 \times 10^{-3}$	$4{,}99997 \times 10^{-3}$	$4{,}99997 \times 10^{-3}$	$4{,}99997 \times 10^{-3}$
$\overline{f(x)}$	-	$7{,}28977 \times 10^{-3}$	$5{,}00017 \times 10^{-3}$	$4{,}99999 \times 10^{-3}$	$5{,}00003 \times 10^{-3}$
σ	-	$1{,}18870 \times 10^{-2}$	$3{,}66751 \times 10^{-7}$	$7{,}12732 \times 10^{-8}$	$1{,}80324 \times 10^{-7}$

Tabela 2: Resultados obtidos com LJ e PSO para a Exponential Fit (30 experimentos).

Compo-nente	Solução [4]	LJ	PSO
x_1	0,3114102 x 10^{-2}	0,33748 x 10^{-2}	0,3114102 x 10^{-2}
x_2	0,3459792 x 10^{2}	0,31931 x 10^{2}	0,3459792 x 10^{2}
x_3	0,6504177 x 10^{-1}	0,677006 x 10^{-1}	0,6504178 x 10^{-1}
x_4	0,8593780	0.85949	0,8593780
x_5	0,3695185 x 10^{-1}	0,369586 x 10^{-1}	0,3695186 x 10^{-1}
$f(x)$	0	1,84548 x 10^{-7}	2,22796 x 10^{-26}
$\overline{f(x)}$	-	1,26366 x 10^{-3}	5,54412 x 10^{-16}
σ	-	1,75874 x 10^{-3}	2,84885 x 10^{-11}

Tabela 3: Resultados obtidos com LJ e PSO para o Equilíbrio Quimico (30 experimentos).

6. Conclusões

Os resultados da Tabela 2 exibem a capacidade das técnicas de otimização utilizadas de "buscar" os valores ótimos para a função teste em um problema de otimização e indicam um melhor desempenho do PSO em relação ao LJ.

As tabelas 2 e 3 mostram que o algoritmo de Luus-Jaakola atingiu no processo de otimização pontos próximos das soluções esperadas enquanto o PSO as atingiu com uma melhor aproximação dos resultados obtidos anteriormente [4] e [5], como mostra os desvio padrão dos resultados obtidos. Como pode ser visto em [6], o LJ pode tornar-se deficiente em problemas com grande número de dimensões e mesmo com a redução matemática do problema de equilíbrio químico de 10 para 5 dimensões tal algoritmo apresentou desempenho não satisfatório em comparação com o PSO, porém os resultados obtidos encontram-se dentro de uma margem aceitável para o problema.

Agradecimentos:
Os autores Jardel S. Costa e Carlos A. Souza Lima Jr agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior (CAPES) pelo suporte financeiro.

Referências

1. KENNEDY, J.; EBERHART, R.C. , “Swarm Intelligence”, Academic Press., 2001.
2. LAUREANO, M. ;“Programando em C”, Ed. Brasport livros e Multimidia Ltda,2005.
3. LUUS, R. e JAAKOLA, T.H.I.;“Optimization by Direct Search and Systematic Reduction of the Size of Search Region”, AIChE Journal, 19, (1973), 760-766.
4. MEINTJES, K. e MORGAN, A. P.;"Chemical Equilibrium Systems as Numerical Test Problems", ACM Transactions on Mathematical Software, Vol. 16, No. 2, 1990.
5. NIELSEN, H. B.; "UCTP Test Problems for unconstrained optimization", Technical Report, Department of Mathematical Modelling, University of Denmark (2000).
6. RIOS COELHO, A. C.; OLIVEIRA, L. N. H. G.; SACCO, W. F. ; DOMINGOS, R. P. , “Comparison of the Luus-Jaakola Algorithm and Particle Swarm Optimization Applied to a Multimodal Test Function”, X Encontro de Modelagem Computacional, Nova Friburgo – RJ, 2007.
7. WOLPERT, D. H.; MACREADY, W. G., “No Free Lunch Theorems for Optimization”, IEEE Trans. Evol. Conq., Vol 1, pp. 67-82, 1997.