Arrefecimento Simulado para Encontrar Raízes de Funções Otimizado por Algoritmo Genético

Cauê Felchar[1], **Iara da Cunha R. da Silva**[2]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR

[2]Departamento Acadêmico de Matemática
Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR

caue_abrir@hotmail.com, iarasilva@utfpr.edu.br

Abstract. *In this paper we present a simulated annealing heuristic optimized by a genetic algorithm for the numerical problem of finding roots of a real function. In the simulated annealing method there is a function whose parameters can be optimized. The use of good parameters in this method results in reduced execution time, improved performance and quality of results. A genetic algorithm was used to find these optimal parameters for the simulated annealing heuristic.*

Resumo. *Neste artigo é apresentado um método heurístico de arrefecimento simulado (AS) otimizado por um algoritmo genético (AG) para o problema numérico de encontrar zeros de funções reais. No método de arrefecimento simulado há uma função cujos parâmetros podem ser otimizados. O uso de bons parâmetros nesse método resulta na redução do tempo de execução, melhora do desempenho e da qualidade dos resultados. Para encontrar esses parâmetros ótimos para o arrefecimento simulado utilizou-se um algoritmo genético.*

1. Introdução

Muitos problemas reais podem ser modelados matematicamente como um sistema de equações não lineares:

$$f_i(x) = 0,\ i = 1, \ldots n,\ x \in \mathbb{R}^n.$$

Neste artigo, a dimensão do problema citado anteriormente será restringida a $n = 1$, ou seja,

$$f(x) = 0,\ x \in \mathbb{R}, \tag{1}$$

deste modo, a solução será x^* tal que $f(x^*)$.

Na literatura [Nocedal and Wright 2006, Franco 2006, Burden and Faires 2011, Ruggiero and da Rocha Lopes 1996, Alfio Quarteroni and Saleri 2000], existem vários métodos determinísticos para solucionar o problema (1), mas a maioria desses métodos apresentam um bom desempenho sob algumas hipóteses, como por exemplo, deve satisfazer ser contínua e suave. Apresentamos, neste artigo, um método um método heurístico, que solucione o problema 1, independentemente de que f seja contínua ou suave. Este método é o arrefecimento simulado, ou *simulated annealing* em inglês, uma técnica probabilística robusta. Para se obter um bom resultado na utilização do arrefecimento simulado deve-se escolher bons parâmetros de entrada, para isso utilizou-se de um algoritmo genético.

Este artigo está organizado da seguinte forma: na Seção 2, é apresentado o método de arrefecimento simulado; na Seção 3, explicamos como foi desenvolvido o método de arrefecimento simulado via algoritmo genético; na Seção 4, é dada uma breve explicação de como os métodos foram implementados; na Seção 5, são apresentados os resultados obtidos.

2. Método de Arrefecimento simulado

Nascido de noções termodinâmicas do comportamento de sistemas físicos, o arrefecimento simulado tomou como base a observação de que para se obter um sistema em seu menor estado de entalpia deve-se primeiramente aumentar a energia do sistema, assim garantindo que todas as moléculas possam ultrapassar a barreira energética das transições de estado [Kirkpatrick et al. 1983]. Podemos pensar no arrefecimento simulado como uma subida de encosta (*hill-climbing*) que tem possibilidade de seguir um caminho desfavorável, sendo que a probabilidade de seguir tal caminho diminui exponencialmente com o fim da quantidade de iterações (3), ou seja, a cada iteração do método pega-se um vizinho do ponto analisado e utiliza-se a função de probabilidade (2) para saber se tal ponto deve ser aceito como ponto atual. A função probabilidade que utilizamos neste trabalho, assemelha-se à distribuição de Gibbs, também conhecida como distribuição canônica [Sen and Stoffa 2013] e é definida como:

$$P(x) = e^{\frac{x_i - x_{i-1}}{T}} \tag{2}$$

onde x_i é a solução da iteração corrente, x_{i-1} é a solução da iteração anterior e T é a temperatura.

A temperatura é definida como:

$$T = \alpha T,\ \alpha \in (0, 1). \tag{3}$$

Na figura (1), podemos observar que quando a temperatura T diminui, a probabilidade (P) de aceitar uma solução não tão boa também diminui. Como nosso problema é encontrar zeros de funções reais, uma solução não boa $\bar{x}$, é uma solução onde $f(\bar{x}) \gg 0$.

O algoritmo completo do arrefecimento simulado pode ser visto a seguir:

function $Arrefecimento\_Simulado(funcao, erro, resfriamento, T_i, max_{it})$
$iteracoes \leftarrow 0$
$atual \leftarrow inicio$
$f_a \leftarrow funcao(a)$

while $|f_a| > erro\ and\ i$ **do**
$1 \leftarrow 2$

3. Escolha de parâmetros via algoritmo genético

Para um bom funcionamento do método de arrefecimento simulado é necessário ajustar seus parâmetros de forma que se tenha o comportamento desejado. Para isso, utilizamos um algoritmo genético que consiste em testar valores aleatórios para a função (4):

$$fitness = \frac{|r - r_c| * \Delta_x}{it}, \tag{4}$$

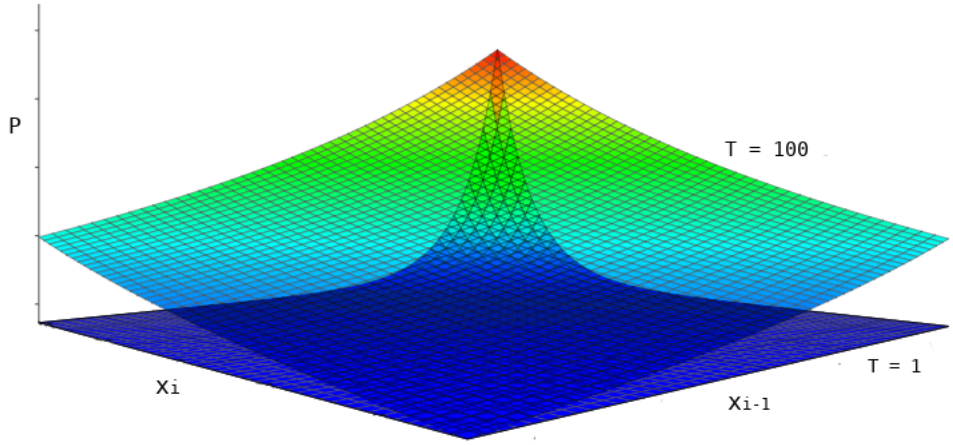


Figura 1. Função Probabilidade considerando $T = 100$ e $T = 1$.

onde r_c é a raiz calculada, r é a raiz exata mais próxima de r_c, Δ_x é a variação entre os extremos vistos pelo método e it é o número de iterações.

A função *fitness* (4), também chamada de função de vitalidade, testa a eficiência dos indivíduos (parâmetros). A heurística de cruzamento desenvolvida neste trabalho consiste em ordenar os indivíduos em relação ao valor do *fitness*. O algoritmo genético implementado necessita das seguintes funções:

- **gera_individuo:** função que gera os indivíduos de uma população;
- **cruzamento:** seleciona os 10% dos indivíduos melhores e piores para que haja uma variação da população.

A função de cruzamento normalmente é feita transformando os dados relevantes ao indivíduo em uma cadeia de bits, e tais bits são tratados como cromossomos em um processo análogo à gametogênese [Goldberg 1989], porém neste trabalho utilizou-se uma implementação diferente. Os melhores indivíduos tem chance de modificar os indivíduos de *fitness* médio, fazendo uma média ponderada entre os valores dos indivíduo, a variação aleatória está na probabilidade do cruzamento ocorrer e nos fatores que ponderam a média entre os indivíduos. Os indivíduos que possuem pior *fitness* são descartados, tendo seus valores gerados novamente pela função de gerar indivíduos.

A rotina com a implementação do algoritmo genético e as funções *gera_individuo* e *cruzamento* podem ser vistas logo a seguir:

function $gera\_individuo(individuo\&)$
$individuo.it_{max} \leftarrow random[0, 50000)$
$individuo.resfriamento \leftarrow random[0, 1)$
$individuo.T_i \leftarrow random[0, 50000)$

```
    individuo.x_0 ← random[0, 50)

function cruzamento(populacao, individuos[])
    j ← 0,
    limite = populacao_size * 0.3
    for i ← limite; i < populacao − limite; i ← i + 1 do
        if random[0, 1) > 0.6 then
            rnd ← random[0, 1)
            individuos[i] ← individuos[i] * rnd + individuos[j] * (1 − rnd)
        j ← j + 1
        if j > limite then
            j ← 0
    for i ← populacao − limit; i < limit; i ← i + 1 do
        gera_individuo(individuos[i])

function algoritmo_genetico(populacao, geracoes)
    individuo = individuos[populacao]
    for i ← 0; i < populacao; i ← i + 1 do
        gera_individuo(individuos[i])
    gen_n ← 0
    while gen_n < geracoes do
        for i ← 0; i < populacao; i ← i + 1 do
            set_fitness(individuos[i])
        Sort(individuos)
        cruzamento(individuos, populacao)
```

4. Implementação do método

O método foi implementado em C++ devido a existência de bons geradores de números aleatórios na biblioteca padrão. Para utilizar o método o usuário deve criar uma função do tipo $y = f(x)$, com entrada e saída *double*:

```
function fun(x)
    return pow(x, −3) * log(x)
```

e passar o endereço dessa função para a chamada da função de arrefecimento simulado.

A função *Arrefecimento_Simulado* retorna um ponteiro para estrutura do tipo *f_out*, que contém os valores: *x*, *y*, *max_x*, *min_x*, *doubles* que, respectivamente, representam o melhor valor encontrado pelo anelamento, o valor de tal ponto na função, o maior valor de *x* avaliado pelo método, e o menor valor de *x* avaliado pelo método. O parâmetro *f_out* também possui um inteiro *it*, que contém o número de iterações feitas pelo método, e, caso a *flag string_out* esteja como verdadeira, *s* conterá todos os pontos avaliados pelo método, na forma $f(x), x; \backslash n$

5. Resultados Computacionais

O método arrefecimento simulado foi aplicado nas seguintes funções reais:

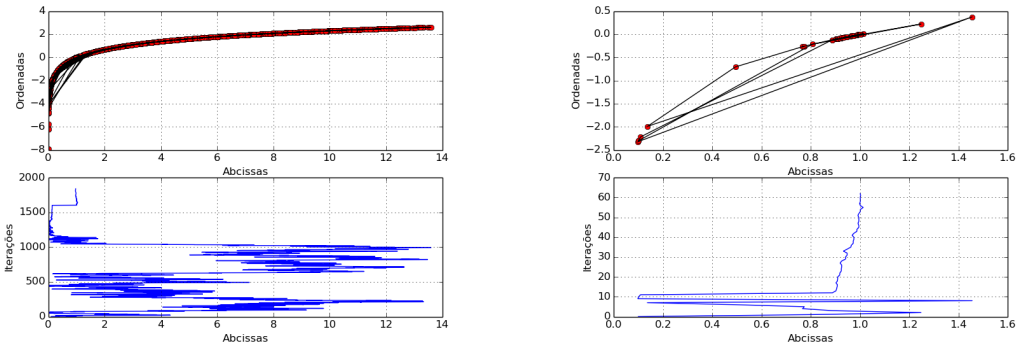
- $f(x) = ln(x)$

- $f(x) = sen^2(x) + cos(x)$
- $\frac{e^{tan(x)}}{1+x^2} sen\left(\sqrt{1 - ln^2(x)}\right)$

Implementamos duas versões do método arrefecimento simulado, uma sem otimizar os parâmetros e a outra otimizada por algoritmo genético. As funções reais utilizadas nos testes foram aplicada nos métodos determinísticos: Newton-Raphson e secante para efeito comparativo. O método de Newton-Raphson, para ter uma convergência quadrática [Alfio Quarteroni and Saleri 2000] precisa de um ponto inicial para iniciar o método próximo da raiz da função. O método da secante tem uma convergência superlinear [Ruggiero and da Rocha Lopes 1996] e precisamos de dois pontos iniciais. Ambos os métodos tem a garantia de convergência para funções contínuas e diferenciáveis [Burden and Faires 2011].

Os gráficos foram gerados utilizando cada ponto escolhido pelo algoritmo, sendo plotados em gráficos $x \times f(x)$ e $x\times$ número de iterações, para que possamos observar as escolhas do algoritmo a cada iteração.

Na figura a seguir, temos o funcionamento do método na função $f(x) = ln(x)$. Podemos observar que a versão do algoritmo após a escolha dos parâmetros pelo algoritmo genético foi bem mais eficiente.



Antes do GA Depois do GA

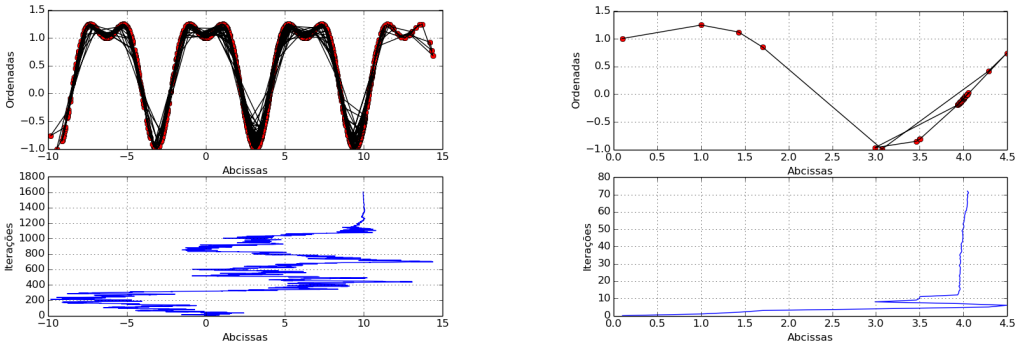
Figura 2. Comportamento do SA para a função $f(x) = ln(x)$.

Tabela 1. Solução final para a função $f(x) = ln(x)$.

	x	y	iterações
Antes do AG	0.999903	$-9.70308e-05$	1843
Pós do AG	1.00017	0.000166289	64

Para a função $f(x) = ln(x)$, o método convergiu com seis iterações para o método Newton-Raphson e não houve convergência utilizando o método da secante.

Os resultados a seguir, referem-se a função contínua $f(x) = sen^2(x) + cos(x)$, com infinitas raízes. Para essa função, os métodos determinísticos convergiram: o método de Newton-Raphson convergiu para a raiz $x = -14,80$ com oito iterações e a secante convergiu para $x = 2,23$ com 5 iterações.



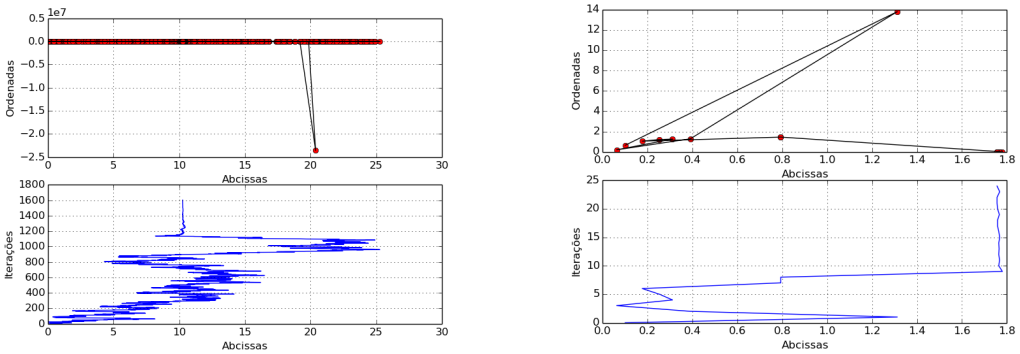
Antes do GA Depois do GA

Figura 3. Comportamento do SA para a função $f(x) = sen^2(x) + cos(x)$.

Tabela 2. Solução final para a função $f(x) = sen^2(x) + cos(x)$.

	x	y	iterações
Antes do AG	8.51982	0.000702022	1604
Após o AG	4.0467	0.000969026	74

Os resultados a seguir referem-se à função descontínua $\frac{e^{tan(x)}}{1+x^2} sen\left(\sqrt{1 - ln^2(x)}\right)$. Para essa função, os métodos Newton-Raphson e da secante não convergiram.



Antes do GA Depois do GA

Figura 4. Comportamento do SA para a função $\frac{e^{tan(x)}}{1+x^2} sen\left(\sqrt{1 - ln^2(x)}\right)$.

Tabela 3. Solução final para a função $\frac{e^{tan(x)}}{1+x^2} sen\left(\sqrt{1 - ln^2(x)}\right)$.

	x	y	iterações
Antes do AG	1.57204	0	1604
Após o AG	1.74458	0.000756225	26

6. Conclusão

O método de arrefecimento simulado é um bom método exploratório, apesar do comportamento mais aleatório no começo, o mesmo convergiu para o melhor valor encontrado ao final de seu funcionamento até para os casos de funções não contínuas e não suaves. Podemos observar através dos resultados que ao otimizarmos os parâmetros do método de arrefecimento simulado por algoritmo genético, houve uma redução média de aproximadamente 96,77% do número de iterações. A vantagem do método de arrefecimento simulado é que além de retornar resultados satisfatórios, não precisas de pontos inicias e hipótese sobre a função que interviram na convergência.

Referências

Alfio Quarteroni, R. S. and Saleri, F. (2000). *Numerical mathematics*. Springer, New York, 1 edition.

Burden, R. L. and Faires, J. D. (2011). *Numerical Analysis*. Cengage Learning, 9 edition.

Franco, N. B. (2006). *Cálculo numérico*, volume 1. São Paulo: Pearson Prentice Hall, 1 edition.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2 edition.

Ruggiero, M. A. and da Rocha Lopes, V. L. (1996). *Cálculo numérico: aspectos teóricos e computacionais*. Makron Books do Brasil.

Sen, M. K. and Stoffa, P. L. (2013). *Global Optimization Methods in Geophysical Inversion*. Cambridge University Press.